Handbook of Combinatorial Optimization

Supplement Volume B

Ding-Zhu Du and Panos M. Pardalos (Eds.)

Springer

HANDBOOK OF COMBINATORIAL OPTIMIZATION

Supplement Volume B

HANDBOOK OF COMBINATORIAL OPTIMIZATION

Supplement Volume B

Edited by

DING-ZHU DU
University of Minnesota, Minneapolis, MN

PANOS M. PARDALOS
University of Florida, Gainesville, FL

Springer

eBook ISBN: 0-387-23830-1
Print ISBN: 0-387-23829-8

Created in the United States of America

Visit Springer's eBookstore at: http://ebooks.springerlink.com
and the Springer Global Website Online at: http://www.springeronline.com

Preface

Combinatorial (or discrete) optimization is one of the most active fields in the interface of operations research, computer science, and applied mathematics. Combinatorial optimization problems arise in various applications, including communications network design, VLSI design, machine vision, airline crew scheduling, corporate planning, computer-aided design and manufacturing, database query design, cellular telephone frequency assignment, constraint directed reasoning, and computational biology. Furthermore, combinatorial optimization problems occur in many diverse areas such as linear and integer programming, graph theory, artificial intelligence, and number theory. All these problems, when formulated mathematically as the minimization or maximization of a certain function defined on some domain, have a commonality of discreteness.

Historically, combinatorial optimization starts with linear programming. Linear programming has an entire range of important applications including production planning and distribution, personnel assignment, finance, allocation of economic resources, circuit simulation, and control systems. Leonid Kantorovich and Tjalling Koopmans received the Nobel Prize (1975) for their work on the optimal allocation of resources. Two important discoveries, the ellipsoid method (1979) and interior point approaches (1984) both provide polynomial time algorithms for linear programming. These algorithms have had a profound effect in combinatorial optimization. Many polynomial-time solvable combinatorial optimization problems are special cases of linear programming (e.g. matching and maximum flow). In addition, linear programming relaxations are often the basis for many approximation algorithms for solving NP-hard problems (e.g. dual heuristics).

Two other developments with a great effect on combinatorial optimization are the design of efficient integer programming software and the availability of parallel computers. In the last decade, the use of integer programming models has changed and increased dramatically. Two decades ago, only problems with up to 100 integer variables could be solved in a computer. Today we can solve problems to optimality with thousands of integer variables. Furthermore, we can compute provably good approximate solutions to problems with millions of integer variables. These advances have been made possible by developments in hardware, software and algorithm design.

The Handbooks of Combinatorial Optimization deal with several algorithmic approaches for discrete problems as well as with many combinatorial problems. We have tried to bring together almost every aspect of this enormous field with emphasis on recent developments. Each chapter in the Handbooks is essentially expository in nature, but of scholarly treatment.

The Handbooks of Combinatorial Optimization are addressed not only to researchers in discrete optimization, but to all scientists in various disciplines who use combinatorial optimization methods to model and solve problems. We are certain that experts in the field as well as nonspecialist readers will find the material of the Handbooks stimulating and helpful.

We would like to take this opportunity to thank the authors, the anonymous referees, and the publisher for helping us produce these volumes of the Handbooks of Combinatorial Optimization with state-of-the-art chapters. We would also like to thank Mr. Arnold Mayaka for making Author Index and Subject Index for this volume.

Ding-Zhu Du and Panos M. Pardalos

Contents

HANDBOOK OF COMBINATORIAL OPTIMIZATION(VOL 5)
D.-Z. Du and P.M. Pardalos (Eds.) pp. 1-53

Data Correcting Algorithms in Combinatorial Optimization

Diptesh Ghosh
P&QM Area, Indian Institute of Management
Vastrapur, Ahmedabad 380015, Gujarat, India
E-mail: diptesh@iimahd.ernet.in

Boris Goldengorin
Faculty of Economic Sciences
University of Groningen, 9700AV Groningen, The Netherlands
E-mail: b.goldengorin@eco.rug.nl

Gerard Sierksma
Faculty of Economic Sciences
University of Groningen, 9700AV Groningen, The Netherlands
E-mail: g.sierksma@eco.rug.nl

Contents

1 Introduction

Polynomially solvable special cases have long been studied in the literature on combinatorial optimization problems (see, for example, Gilmore *et al.* [19]). Apart from being mathematical curiosities, they often provide important insights for serious problem-solving. In fact, the concluding paragraph of Gilmore *et al.* [19] states the following, regarding polynomially solvable special cases for the traveling salesperson problem.

> " ... We believe, however, that in the long run the greatest importance of these special cases will be for approximation algorithms. Much remains to be done in this area."

This chapter describes a step in the direction of incorporating polynomially solvable special cases into approximation algorithms. We review *data correcting algorithms* — approximation algorithms that make use of polynomially solvable special cases to arrive at high-quality solutions. The basic insight that leads to these algorithms is the fact that it is often easy to compute a bound on the difference between the costs of optimal solutions to two instances of a problem, even though it may be hard to compute optimal solutions for the two instances. These algorithms were first reported in the Russian literature (see Goldengorin [9, 10, 11, 12, 13]).

The approximation in data correcting algorithms is in terms of an *accuracy parameter,* which is an upper bound on the difference between the objective value of an optimal solution to the instance and that of a solution

returned by the data correcting algorithm. Note that this is *not* expressed as a fraction of the optimal objective value for this instance as in common ε-optimal algorithms but as actual deviations from the cost of optimal solutions.

Although we suggest the use of data correcting algorithms to solve NP-hard combinatorial optimization problems, they form a general problem solving tool and can be used for functions defined on a continuous domain as well. We will in fact, motivate the algorithm in the next section using a function defined on a continuous domain, and having a finite range. We then show in Section 3, how this approach can be adapted for combinatorial optimization problems. In the next three sections we describe actual implementation of data correcting algorithms to three problems, the asymmetric traveling salesperson problem, the maximization of a general submodular function, and the simple plant location problem.

2 Data Correcting for Real-Valued Functions

Consider a real-valued function $f : D \to \Re$, where D is the domain on which the function is defined. We assume that f is not analytically tractable over D, but is computable in polynomial time for any $x \in D$, and concern ourselves with the problem of finding α-minimal solutions to the function f over D, i.e. the problem of finding a member of $\{x | x \in D, f(x) \leq f(x^*) + \alpha\}$, where $x^* \in \arg\min_D\{f(x)\}$, and α is the pre-defined *accuracy* parameter. The discussion here is for a minimization problem; the maximization problem can be dealt with in a similar manner.

Let us assume a partition $\{D_1, \ldots, D_p\}$ of the domain D. Let us further assume that for each of the sub-domains D_i of D, $i = 1, \ldots, p$, we are able to find functions $g_i : D_i \to \Re$, which are easy to minimize over D_i, and such that

$$|f(x) - g_i(x)| \leq \frac{\alpha}{2} \quad \forall x \in D_i. \tag{1}$$

We call such easily minimizable functions *regular*.

Theorem 2.1 demonstrates an important relationship between the regular functions g_i and the original function f. It states that the function value of f at the best among the minima of all the g_i's over their respective domains is close to the minimum function value of f over the domain.

Theorem 2.1 *Let* $x_i^\alpha \in \arg\min_{x \in D_i}\{g_i(x)\}$, $x^\alpha \in \arg\min_i\{f(x_i^\alpha)\}$, *and let* $x^\star \in \arg\min_{x \in D}\{f(x)\}$. *Then*

$$f(x^\alpha) \leq f(x^\star) + \alpha.$$

Proof: Let $x_i^\star \in \arg\min_{x \in D_i}\{f(x)\}$. Then for $i = 1, \ldots, p$, $f(x_i^\alpha) - \frac{\alpha}{2} \leq g_i(x_i^\alpha) \leq g_i(x_i^\star) \leq f(x_i^\star) + \frac{\alpha}{2}$,i.e. $f(x_i^\alpha) \leq f(x_i^\star) + \alpha$. Thus $\min_i\{f(x_i^\alpha)\} \leq \min_i\{f(x_i^\star)\} + \alpha$, which proves the result. ■

Notice that $x^\star$ and x^α do not need to be in the same sub-domain of D.

Theorem 2.1 forms the basis of the data correcting algorithm to find an approximate minimum of a function f over a certain domain D. The procedure consists of three steps: the first in which the domain D of the function is partitioned into several sub-domains; the second in which f is approximated in each of the sub-domains by regular functions following the condition in expression (1) and a minimum point of the regular function is obtained; and a third step, in which the minimum points computed in the second step are considered and the best among them is chosen as the output. This procedure can be further strengthened by using lower bounds to check if a given sub-domain can possibly lead to a solution better than any found thus far. The approximation of f by regular functions g_i is called *data correcting,* since an easy way of obtaining the regular functions is by altering the data that describe f. A pseudocode of the algorithm, which we call DC-G, is provided below.

Procedure DC-G(f, D, α)
Output: $x^\alpha \in D$ such that $f(x^\alpha) \leq \min\{f(x) | x \in D\} + \alpha$.

Code:

```
1. begin
2.       create a partition {D_1, ..., D_n} of D;
3.       for each sub-domain D_i
4.       begin
5.             f_i := a lower bound to f(x), x ∈ D_i;
6.             if f_i ≥ bestvalue
7.                   continue;
8.             construct a regular function g_i(x) obeying (1);
9.             x_i^α ∈ arg min_{x∈D_i}{g_i(x)};
10.      end;
11.      bestvalue := ∞;
12.      if f(x_i^α) < bestvalue;
```

```
13.       begin
14.             x^α := x_i^α;
15.             bestvalue := f(x^α);
16.       end;
17.       return x^α;
18. end.
```

Lines 5 through 7 in the code carry out the bounding process, and lines 8 and 9 implement the process of computing the minima of regular functions over each sub-domain. These steps are enclosed in a loop, so that at the end of line 10, all the minima of the regular functions are at hand. The code in lines 11 through 16 obtain the best among the minima obtained before. By Theorem 2.1, the solution chosen by the code in lines 11 through 16 is an α-minimum of f, and therefore, this solution is returned by the algorithm in line 17.

We will now illustrate the data correcting algorithm through an example. The example that we choose is one of a real-valued function of one variable, since these are some of the simplest functions to visualize.

Consider the problem of finding an α-minimum of the function f shown in Figure 1. The function is assumed to be well-defined, though analytically intractable on the domain D.

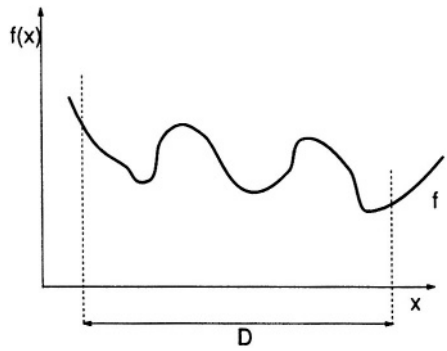


Figure 1: A general function f

The data correcting approach can be used to solve the problem above,

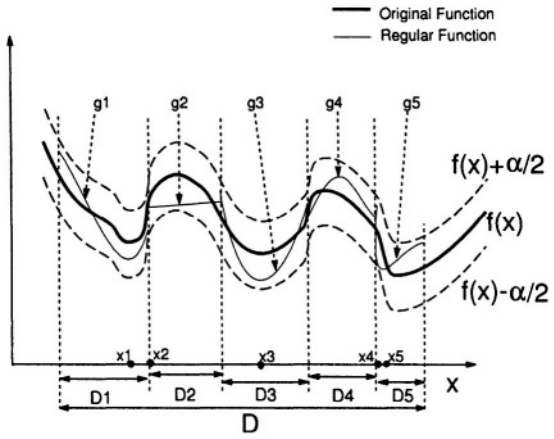


Figure 2: Illustrating the Data Correcting approach on f

i.e. of finding a solution $x^{\alpha} \in D$ such that $f(x^{\alpha}) \leq \min\{f(x) | x \in D\} + \alpha$.

Consider the partition $\{D1, D2, D3, D4, D5\}$ of D shown in Figure 2. Let us suppose that we have a regular function $g1(x)$ with $|g1(x) - f(x)| \leq \frac{\alpha}{2}$, $\forall x \in D1$. Assume also, that $x1$ is a minimum point of $g1(x)$ in $D1$. Since this is the best solution that we have so far, we store $x1$ as an α-minimal solution to $f(x)$ in the domain $D1$. We then consider the next interval in the partition, $D2$. We first obtain a lower bound on the minimum value of $f(x)$ on $D2$. If this bound is larger than $f(x1)$, we ignore this domain and examine domain $D3$. Let this not be the case in our example. So we construct a regular function $g2(x)$ with $|g2(x) - f(x)| \leq \frac{\alpha}{2}$, $\forall x \in D2$, and find $x2$, its minimum point over $D2$. Since $f(x2) \geq f(x1)$ (see Figure 2), we retain $x1$ as our α-optimal solution over $D1 \cup D2$. Proceeding in this manner, we examine $f(x)$ in $D3$ through $D5$, compute regular functions $g3(x)$ through $g5(x)$ for these domains, and compute $x3$ through $x5$. In this example, $x3$ replaces $x1$ as our α-minimal solution after consideration of $D3$, and remains so until the end. At the end of the algorithm, $x3$ is returned as a value of x^{α}.

There are four points worth noting at this stage. The first is that we need to examine all the sub-domains in the original domain before we re-

turn a near-optimal solution using this approach. The reason for this is very clear. The correctness of the algorithm depends on the result in Theorem 2.1, and this theorem only concerns the *best* among the minima of each of the sub-domains. For instance, in the previous example, if we stop as soon as we obtain the first α-optimal solution x_1 we would be mistaken, since Theorem 2.1 applies to $x1$ only over $D1 \cup D2$. The second point is that there is no guarantee that the near-optimal solution returned by DC-G will be in the neighborhood of a true optimal solution. There is in fact, nothing preventing the near-optimal solution existing in a sub-domain different from the sub-domain of an optimal solution, as is evident from the previous example. The true minimum of f lies in the domain $D5$, but DC-G returns $x3$, which is in $D3$. The third point is that the regular functions $g_i(x)$ approximating $f(x)$ do not need to have the same functional form. For instance in Example 1, $g1(x)$ is quadratic, while $g2(x)$ is linear. Finally, for the proof of Theorem 2.1, it is sufficient for $\{D_1, \ldots, D_n\}$ to be a cover of D (as opposed to a partition as required in the pseudocode of DC-G).

3 Data Correcting for Combinatorial Optimization Problems

The data correcting methodology described in the previous section can be incorporated into an implicit enumeration scheme (like branch and bound) and used to obtain near-optimal solutions to NP-hard combinatorial optimization problems. In this section we describe how this incorporation is achieved for a general combinatorial optimization problem.

We define a combinatorial optimization problem P as a collection of instances I. An instance I consists of a ground set $G = \{e_1, e_2, \ldots, e_n\}$ of n elements, a cost vector $C_I = (c_1^I, c_2^I, \ldots, c_n^I)$ corresponding to the elements in G, a set $S \subseteq 2^G$ of feasible solutions, and a cost function $f_I : S \to \Re$. The objective is to obtain a solution, i.e. a member of S that minimizes the cost function. For example, for an asymmetric traveling salesperson problem (ATSP) instance on a digraph $G = (V, A)$, with a distance matrix $D = [d_{ij}]$, we have $G = A$, $c_{ij}^I = d_{ij}$, S is the set of all Hamiltonian cycles in G, and $f_I(s) = \sum_{(i,j)\in s} d_{ij}$ for each $s \in S$.

Implicit enumeration algorithms for combinatorial problems include two main strategies, namely branching and fathoming. Branching involves partitioning the set of feasible solutions S into smaller subsets. This is done under the assumption that optimizing the cost function over a more restricted so-

lution space is easier than optimizing it over the whole space. Fathoming involves one of two processes. First, we could compute lower bounds to the value that the cost function can attain over a particular member of the partition. If this bound is not better than the best solution found thus far, the corresponding subset in the partition is ignored in the search for an optimal solution. The second method of fathoming is by computing the optimum value of the cost function over the particular subset of the solution space (if that can be easily computed for the particular subset). We see therefore that two of the main requirements of the data correcting algorithm presented in the previous section, i.e. partitioning and bounding, are automatically taken care of for combinatorial optimization problems by implicit enumeration. The only other requirement that we need to consider is that of obtaining regular functions approximating f_I over subsets of the solution space.

Notice that the cost function $f_I(s)$ is a function of the cost vector C. So if the values of the entries in C are changed, $f_I(s)$ undergoes a change as well. Therefore, cost functions corresponding to polynomially solvable special cases can be used as "regular functions" for combinatorial optimization problems. Also note that for the same reason, the accuracy parameter can be compared with a suitably defined distance measure between two cost vectors, (or equivalently, two instances). Consider a subproblem in the tree obtained by normal implicit enumeration. The problem instance that is being evaluated at that subproblem is a restricted version of the original problem instance, i.e., it evaluates the cost function of the original problem instance for a subset S_k of the original solution space S. If we alter the data of the problem instance in a way that the altered data corresponds to a polynomially solvable special case, while guaranteeing that the cost of an optimal solution to the altered problem in S_k is not more that an acceptable amount higher than the cost of an optimal solution to original instance in S_k, then the altered cost function can be considered to be a regular approximation of the cost function of the original instance in S_k.

For combinatorial optimization problems, let us define a *proximity measure* $\rho(I_1, I_2)$ between two problem instances I_1 and I_2, as an upper bound for the difference between $f_{I_1}(s_1^\star)$ and $f_{I_1}(s_2^\star)$, where $s_1^\star$ and $s_2^\star$ are optimal solutions to I_1 and I_2 respectively. The following lemma shows that the *Hamming distance* between the cost vectors of the two instances is a proximity measure when the cost function is of the sum type or the max type.

Lemma 3.1 *If the cost function of an instance I of a combinatorial optimization problem is of the* sum *type, (i.e. $f_I(s) = \sum_{e_k \in s} c_k^I$) or the* max *type, (i.e. $f_I(s) = \max_{e_k \in s} c_k^I$) then the measure*

$$\rho(I_1, I_2) = \sum_{e_i \in G} |c_i^{I_1} - c_i^{I_2}| \tag{2}$$

between two instances I_1 and I_2 of the problem is an upper bound to the difference between $f_{I_1}(s_1^\star)$ and $f_{I_1}(s_2^\star)$, where $s_1^\star$ and $s_2^\star$ are optimal solutions to I_1 and I_2, respectively.

Proof: We will prove the result for sum type cost functions. The proof for max type cost functions is similar.

For sum type cost functions, it is sufficient to prove the result when the cost vectors C_{I_1} and C_{I_2} differ in only one position. Let $c_k^{I_1} = c_k^{I_2}$ for $k = 1, 2, \ldots, j-1, j+1, \ldots, n$, and $c_j^{I_1} \neq c_j^{I_2}$. Consider any solution $s \in S$. There are two cases to consider:

- $e_j \in s$: In this case, $|f_{I_1}(s) - f_{I_2}(s)| = \sum_{e_k \in S} |c_k^{I_1} - c_k^{I_2}| = |c_j^{I_1} - c_j^{I_2}|$.
- $e_j \notin s$: In this case it is clear that $f_{I_1}(s) = f_{I_2}(s)$.

Therefore, $|f_{I_1}(s) - f_{I_2}(s)| \leq \sum_{e_i \in G} |c_i^{I_1} - c_i^{I_2}| = \rho(I_1, I_2)$ for any solution $s \in S$, which automatically implies that $\rho(I_1, I_2)$ as defined in the statement of Lemma 3.1 is an upper bound for the difference between $f_{I_1}(s_1^\star)$ and $f_{I_1}(s_2^\star)$, where $s_1^\star$ and $s_2^\star$ are optimal solutions to I_1 and I_2, respectively. ■

At this point, it is important to point out the difference between a fathoming step and a data correcting step. The bounds used in fathoming steps consistently overestimate the objective function of optimal solutions in maximization problems and underestimate it for minimization problems. The amount of over- or underestimation is not bounded. In data correcting steps however, the "regular function" may overestimate or underestimate the objective function, regardless of the objective of the problem. However, there is always a bound on the deviation of the "regular function" from the objective function of the problem.

One way of implementing the data correcting for a NP-hard problem instance I is the following. We first execute a data correcting step. We construct a polynomially solvable relaxation I_L of the original instance, and obtain an optimal solution x_L to I_L. Note that x_L need not be feasible to I. We next construct the best solution x to I that we can, starting from

x_L. (If such a solution is not possible, we conclude that the instance does not admit a feasible solution.) We also construct an instance I_C of the problem, which will have x as an optimal solution. The proximity measure $\rho(I, I_C)$ then is an upper bound to the difference between the costs of x and of an optimal solution to I. I_C is called a *correction* of the instance I. If the proximity measure is not more that the allowable accuracy, then we can output x as an adequate solution to I. If this is not the case, then we partition the feasible solution space of I (these are formed by adding constraints to the formulation of I) and apply the data correction step to each of these subproblems.

The similarity in the procedural aspects of the data correcting step described above (and illustrated in the example) to fathoming rules used in branch and bound implementations makes it convenient to incorporate data correcting in the framework of implicit enumeration. We present the pseudocode of a recursive version of branch and bound incorporating data correcting below. The initial input to this procedure is the data for the original instance I, the feasible solution set S, any solution $s \in S$, and the accuracy parameter α. Notice that the data correcting step discussed earlier in this section is implemented in lines 6 through 10 in the pseudocode.

Algorithm DC(I, S, α)
Output: $x^\alpha \in S$ such that $f_I(x^\alpha) \leq \min\{f_I(x) | x \in S\} + \alpha$.

Code:

```
1. begin
2.       s := a solution in S;
3.       lb := a lower bound on the value of f_I(x) over S;
4.       if f_I(s) = lb return s;
5.       compute an optimal solution s_L to a polynomially solvable
6.       relaxation I_L to I;
7.       if possible then construct a solution s to I starting from s_L;
8.       else return "infeasible";          (* No solution s can be constructed *)
9.       construct an instance I_C that has s as an optimal solution;
10.       if ρ(I_C, I) ≤ α
11.             return s;
12.       else begin
13.             partition S into subsets S_1 through S_n;
14.             for i := 1 to n                                    (* Branch *)
15.                   s_i := DC(I, S_i, α);
```

```
16.              return the best solution from among s_1 through s_n;
17.        end;
18. end.
```

The algorithm described above is a prototype. We have not specified how the lower bound is to be computed, or which solution to choose in the feasible region, or how to partition the domain into sub-domains. These are details that vary from problem to problem, and are an important part in the engineering aspects of the algorithm. Note that this is just one of many possible ways of implementing a data correcting algorithm.

We can now describe our implementation of data correcting on specific combinatorial optimization problems. The next section deals with asymmetric traveling salesperson problems. The implementation of the data correcting algorithm for this problem closely follows the pseudocode above. Sections 5 and 6 deal with the maximization of general submodular functions and the simple plant location problem. Our implementations of data correcting for these two problems are slightly different, in which the data correction is done in an implicit manner.

4 The Asymmetric Traveling Salesperson Problem

In an asymmetric traveling salesperson problem (ATSP) instance we are given a weighted digraph $G = (V, A)$ and a $|V| \times |V|$ distance matrix $D = [d_{ij}]$ and our objective is output a least cost Hamiltonian cycle in this graph. This is one of the most studied problems in combinatorial optimization, see Lawler *et al.* [26] and Gutin and Punnen [20] for a detailed introduction.

4.1 The Data Correcting Algorithm

The data correcting algorithm (DC) presented in the previous section can be easily mapped to the ATSP. Lemma 3.1 takes the following form for ATSP instances.

Lemma 4.1 *Consider two ATSP instances I_1 and I_2 defined on digraphs $G_1 = (V_1, A_1)$ and $G_2 = (V_2, A_2)$ respectively, with $|V_1| = |V_2|$. Let $D_1 = [d^1_{ij}]$ and $D_2 = [d^2_{ij}]$ be the distance matrices associated with I_1 and I_2.*

Further let T_1 *and* T_2 *be optimal solutions to* I_1 *and* I_2, *and let* L_1 *and* L_2 *respectively represent the lengths of* T_1 *and* T_2 *in instance* I_1. *Then*

$$L_2 - L_1 \leq \sum_{(i,j)\in A} |d_{ij}^1 - d_{ij}^2|.$$

Before presenting a pseudocode for the algorithm, let us illustrate the data correcting step for the ATSP with an example. Consider the 6-city ATSP instance with the distance matrix $D = [d_{ij}]$ shown below. (This corresponds to I in the discussion in the previous section.)

D	1	2	3	4	5	6
1	-	10	16	19	25	22
2	19	-	10	13	13	10
3	10	28	-	22	16	13
4	19	25	13	-	10	19
5	16	22	19	13	-	11
6	13	22	15	13	10	-

If we allow subtours in solutions to the ATSP, we get the assignment problem relaxation. Solving the assignment problem on D, using the Hungarian method, we get the following reduced distance matrix $D^H = [d_{ij}^H]$. (The assignment problem with the distance matrix D corresponds to I_L.)

D^H	1	2	3	4	5	6
1	-	0	6	7	16	12
2	9	-	0	1	4	0
3	0	18	-	10	7	3
4	8	14	2	-	0	8
5	5	11	8	0	-	0
6	2	11	4	0	0	-

This leads to a solution with two cycles (1231) and (4564) (corresponding to x_L). Using patching techniques (see for example Lawler *et al.* [26]), we obtain a solution (1245631) (corresponding to x). Notice that (1245631) would be an optimal solution to the assignment problem if d_{24}^H and d_{63}^H had been set to zero in D^H, and that would have been the situation, if d_{24} and d_{63} were initially reduced by 4 and 1 respectively, i.e. if the distance matrix in the original ATSP instance was D^P defined below. (This corresponds to I_C.)

D^P	1	2	3	4	5	6
1	-	10	16	19	25	22
2	19	-	10	9	13	10
3	10	28	-	22	16	13
4	19	25	13	-	10	19
5	16	22	19	13	-	11
6	13	22	14	13	10	-

Therefore D^P is the distance matrix of the correction of the instance with distance matrix D. The proximity measure $\rho(D, D^P) = \sum_{i=1}^{6} \sum_{j=1}^{6} |d_{ij} - d_{ij}^P| = |d_{24} - d_{24}^P| + |d_{63} - d_{63}^P| = 4 + 1 = 5$.

A proximity measure is an upper bound to the difference between the costs of two solutions for a problem instance, so the stronger the bound, the better would be the performance of any enumeration algorithm dependent on such bounds. It is possible to obtain stronger performance measures for ATSP instances, for example

$$\rho_1(D, D^P) = \min \Big\{ \sum_{i=1}^{n} \max_{1 \le j \le n} |d_{ij} - d_{ij}^P|, \sum_{j=1}^{n} \max_{1 \le i \le n} |d_{ij} - d_{ij}^P| \Big\} \qquad (3)$$

is a better proximity measure than the one defined in (2).

Given the data correcting step, the DC algorithm presented in the previous section can be modified to solve ATSP instances. The pseudocode for a recursive version of this algorithm is given below.

Algorithm DCA-ATSP(G, α)
Output: A tour x^α such that the difference between the cost of x^α and the cost of an optimal tour is not more than α

Code:

```
1. begin
2.       s := an arbitrary tour in G;
3.       lb := a lower bound on the cost of an optimal tour;
4.       if f_I(s) = lb return s;
5.       s_L := an optimal solution to the assignment problem on G;
6.       construct a solution s from s_L through patching;
7.       using s_L, compute the proximity measure ρ;
8.       if ρ ≤ α
9.             return s;
```

```
10.     else begin                                          (* Branch *)
11.         branch according to a pre-decided branching rule;
12.         for each subproblem i generated
13.             s_i := the solution output by DCA-ATSP on subproblem i;
14.         return the best solution from among all s_i's;
15.     end;
16. end.
```

Note that a good lower bound can be incorporated into DCA-ATSP to make it more efficient.

We next illustrate the DCA-ATSP algorithm above on an instance of the ATSP. Consider the 8-city ATSP instance with the distance matrix $D = [d_{ij}]$ shown below. (This example was taken from Balas and Toth [1], p. 381).

D	1	2	3	4	5	6	7	8
1	-	2	11	10	8	7	6	5
2	6	-	1	8	8	4	6	7
3	5	12	-	11	8	12	3	11
4	11	9	10	-	1	9	8	10
5	11	11	9	4	-	2	10	9
6	12	8	5	2	11	-	11	9
7	10	11	12	10	9	12	-	3
8	7	10	10	10	6	3	1	-

We use

- the proximity measure ρ (see Expression (2)) for data correction,
- the assignment algorithm to compute lower bounds for subproblems,
- a patching algorithm to create feasible solutions, and compute proximity measures, and
- the patched solution derived from the assignment solution as a feasible solution in the domain.

The branching rule used in this example is as follows. At each subproblem, we construct the assignment problem solution and then patch it. We also correct the original matrix to a new matrix that would output the patched solution if the assignment problem is solved on it. We next identify

the arc corresponding to the entry in the new matrix that had to be corrected by the maximum amount. The tail of this arc is identified, and we branch on all the arcs in the subtour containing that vertex. For example, in this problem, the assignment solution is (1231)(4564)(787), which is patched to (123786451) and the cost of patching is 9. If we correct the problem data, we will see that the entry d_{51} (corresponding to arc (5,1)) contributes the maximum amount (7) to the patching. Hence we branch on each arc in the cycle (4564), and construct three subproblems, the first with the additional constraint that arc (4,5) be excluded from the solution, the second with the additional constraint that arc (5,6) be excluded from the solution, and the third with the additional constraint that arc (4,5) be excluded from the solution.

The polynomially solvable special case that we consider is the set of all ATSP instances for which the assignment procedure gives rise to a cyclic permutation.

Using the branching rule described above, depth-first branch and bound generates the enumeration tree of Figure 3. The nodes are labeled according to the order in which the problems at the corresponding nodes were evaluated.

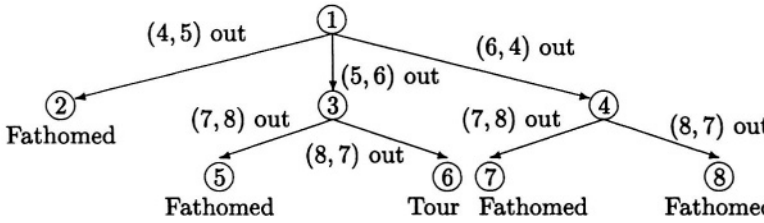
Since the cost of patching equals the value of ρ, we can now evaluate the performance of data correcting on this example. If the allowable accuracy parameter α is set to 0, then the enumeration tree constructed by DC will be the one shown in Figure 3 and evaluates 8 subproblems. However if the value of α is set to 1, then enumeration stops after node 4, since the lower bound obtained is 25 which is one less than the solution we have at hand.

The previous example shows that the data correcting algorithm can be a very attractive alternative to branch and bound algorithms. In the next subsection we report experiences of the performance of the data correcting algorithm on ATSP instances from the TSPLIB [30].

4.2 Computational Experience with ATSP Instances

TSPLIB has twenty seven ATSP instances, out of which we have chosen twelve for our experiments. These twelve can be solved to optimality within five hours using an ordinary branch and bound algorithm. Eight of these belong to the '*ftv*' class of instances, while four belong to the *'rbg'* class. We implemented DCA-ATSP in C and ran it on a Intel Pentium based computer running at 666MHz with 128MB RAM.

The results of our experiments are presented graphically in Figures 4



Subproblem at node	Upper bound	Lower bound	Assignment solution	Patched tour	Cost of patching	Revised bound
1	∞	17	(1231)(4564)(787)	(123786451)	9	26
2	26	29	Fathomed by bounds		–	26
3	26	25	(12631)(454)(787)	(126453781)	6	26
4	26	25	(12631)(454)(787)	(126453781)	6	26
5	26	31	Fathomed by bounds		–	26
6	26	26	(123786451)	Patching not required		26
7	26	31	Fathomed by bounds		–	26
8	26	27	Fathomed by bounds		–	26

Figure 3: Branch and bound tree for the instance in the example.

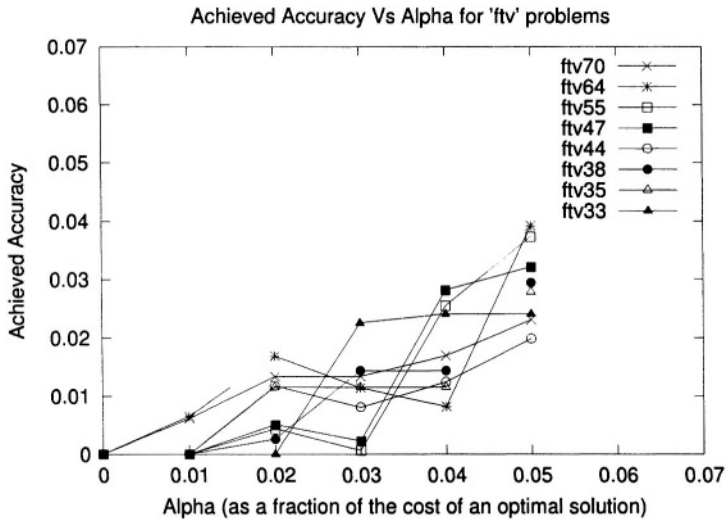


Figure 4: Accuracy achieved versus α for *ftv* instances

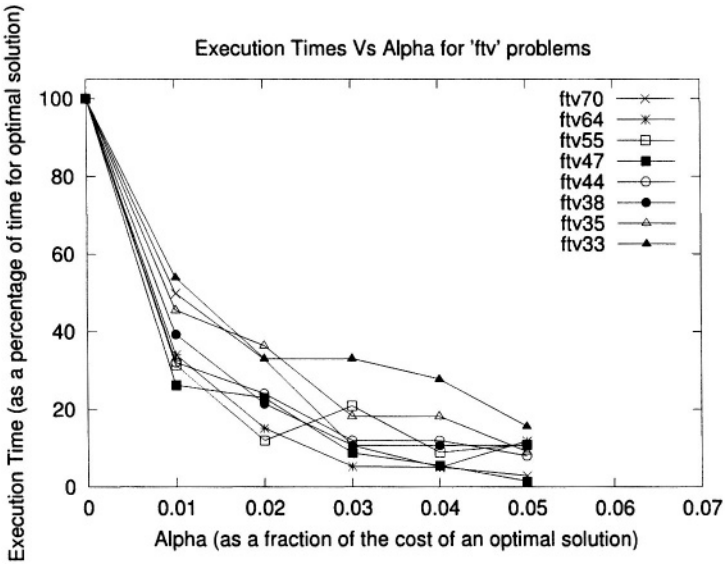


Figure 5: Variation of execution times versus α for *ftv* instances

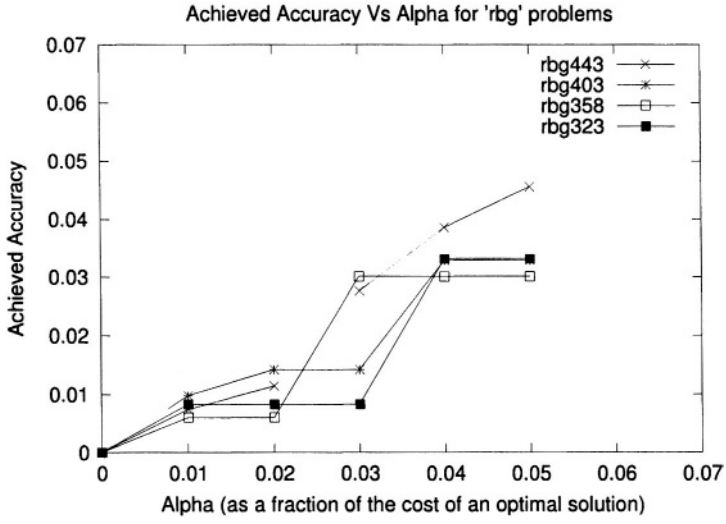


Figure 6: Accuracy achieved versus α for *rbg* instances

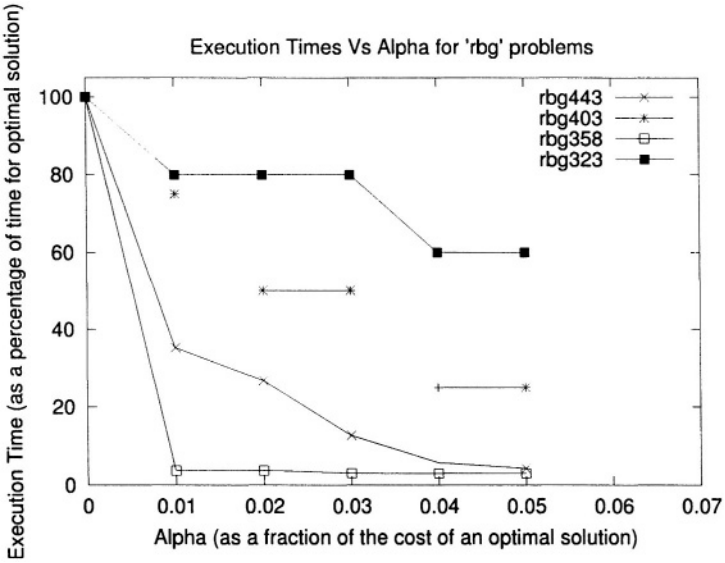


Figure 7: Variation of execution times versus α for *rbg* instances

through 7. In computing accuracies, (Figures 4 and 6) we have plotted the accuracy and deviation of the solution output by the data correcting algorithm from the optimal (called 'achieved accuracy' in the figures) as a fraction of the cost of an optimal solution to the instance. We observed that for each of the twelve instances that we studied, the achieved accuracy is consistently less than 80% of the pre-specified accuracy.

There was a wide variation in the CPU time required to solve the different instances. For instance, *ftv*70 required 17206 seconds to solve to optimality, while *rbg*323 required just 5 seconds. Thus, in order to maintain uniformity while demonstrating the variation in execution times with respect to changes in α values, we represented the execution times for each instance for each α value as a percentage of the execution time required to solve that instance to optimality. Notice that for all the *ftv* instances when α was 5% of the cost of the optimal solution, the execution time reduced to 20% of that required to solve the respective instance to optimality. The reduction in execution times for *rbg* instance was equally steep, with the exception of *rbg*323 which was in any case an easy instance to solve.

In summary, it is quite clear that data correcting is an effective methodology for solving ATSP instances. There are usually steep reductions in

execution times even when the allowed accuracy is very small. This makes the method very useful for solving real world problems where a near-optimal solution is often acceptable provided the execution times are not too long.

5 Maximization of General Submodular Functions

Let $N = \{1, 2, \ldots, n\}$ and 2^N denote the set of all subsets of N. A function $z : 2^N \to \Re$ is called *submodular* if for each $I, J \in 2^N$, $z(I)+z(J) \geq z(I \cup J)+ z(I \cap J)$. The solution process of many classical combinatorial optimization problems, like the generalized transportation problem, the quadratic cost partition (QCP) problem with nonnegative edge weights, and set covering, can be formulated as the maximization of a submodular function (MSF), i.e. the problem:

$$\max\{z(I) | T \subseteq N\}.$$

Although the general problem of the maximization of a submodular function is known to be NP-hard (see Lovasz [28]), there has been a sustained research effort aimed at developing practical procedures for solving medium and large-scale problems in this class. In the remainder of this section we suggest two data correcting algorithms for solving the problem. Note that Lemma 3.1 assumes the following form for this problem.

Lemma 5.1 *Consider two submodular functions* $z_1(\overline{x}) = \sum_{i=1}^{n} \sum_{j=i}^{n} c_{ij}^1 \prod_{k=i}^{j} x_k$ *and* $z_2(\overline{x}) = \sum_{i=1}^{n} \sum_{j=i}^{n} c_{ij}^2 \prod_{k=i}^{j} x_k$. *Let* $\overline{x_1^*}$ *and* $\overline{x_2^*}$ *be the maximum points of* $z_1(\overline{x})$ *and* $z_2(\overline{x})$ *respectively. Then*

$$z_1(\overline{x_1^*}) - z_1(\overline{x_2^*}) \leq \sum_{i=1}^{n} \sum_{j=1}^{n} |c_{ij}^1 - c_{ij}^2|.$$

The algorithm described in Section 5.1 have been published in Goldengorin *et al.* [14] while that described in Section 5.2 have been published in Goldengorin and Ghosh [18]. For each of the two algorithms in we first describe a class of polynomially solvable instances for submodular function maximization problems. We then describe the data correcting algorithms that uses this class of polynomially solvable instances to solve a general submodular function maximization problem. The classes of polynomially solvable instances are algorithmically defined, i.e. they are classes of instances that are solved to optimality using a pre-specified polynomial algorithm.

5.1 A Simple Data Correcting Algorithm

The class of polynomially solvable instances that we describe here is defined using a polynomial time algorithm called the Preliminary Preservation (PP) algorithm. Normally these algorithms terminate with a subgraph of the Hasse diagram of the original instance which is guaranteed to contain the maximum. However, for instances where PP returns a subgraph with a single node, that node *is* the maximum, and the instance is said to have been solved in polynomial time. Instances such as these make up the class of polynomially solvable instances that we consider here.

Let z be a real-valued function defined on the power set 2^N of $N = \{1, 2, \ldots, n\}$; $n \geq 1$. For each $S, T \in 2^N$ with $S \subseteq T$, we define

$$[S,T] = \{I \in 2^N \mid S \subseteq I \subseteq T\}.$$

Note that $[\emptyset, N] = 2^N$. Any *interval* $[S,T]$ is a subinterval of $[\emptyset, N]$ if $\emptyset \subseteq S \subseteq T \subseteq N$. We denote this using the notation $[S,T] \subseteq [\emptyset, N]$. In this section an interval is always a subinterval of $[\emptyset, N]$. It is assumed that z attains a finite maximum value on $[\emptyset, N]$ which is denoted by $z^*[\emptyset, N]$, and $z^*[S,T] = \max\{z(I) | I \in [S,T]\}$ for any $[S,T] \subseteq [\emptyset, N]$. We also define $d_k^+(I) = z(I+k) - z(I)$ and $d_k^-(I) = z(I-k) - z(I)$.

The following theorem and corollaries from Goldengorin *et al.* [14] act as a basis for the Preliminary Preservation (PP) algorithm described therein.

Theorem 5.2 *Let z be a submodular function on $[S,T] \subseteq [\emptyset, N]$ and let $k \in T \backslash S$. Then the following assertions hold.*

1. $z^*[S+k,T] - z^*[S,T-k] \leq z(S+k) - z(S)) = d_k^+(S)$.
2. $z^*[S,T-k] - z^*[S+k,T] \leq z(T-k) - z(T) = d_k^-(T)$.

Corollary 5.3 *(Preservation rules of order zero). Let z be a submodular function on $[S,T] \subseteq [\emptyset, N]$, and let $k \in T \backslash S$. Then the following assertions hold.*

1. *First Preservation Rule: If $d_k^+(S) \leq 0$, then $z^*[S,T] = z^*[S,T-k] \geq z^*[S+k,T]$.*
2. *Second Preservation Rule: If $d_k^-(T) \leq 0$, then $z^*[S,T] = z^*[S+k,T] \geq z^*[S,T-k]$.*

The PP algorithm accepts an interval $[S,T]$, $S \subseteq T$ and tries to apply Corollary 5.3 repeatedly. It returns an interval $[X, Y]$, $S \subseteq X \subseteq Y \subseteq T$, such that $z^*[S,T] = z^*[X,Y]$. The pseudocode for this algorithm is given below.

Algorithm PP$([S,T])$
Output: A subinterval of $[S,T]$ containing the maximum of z over $[S,T]$.

Code:

```
1. begin
2.        if T = S return [S,S];
3.        while T ≠ S do begin
4.              d+max := max{d+k | k ∈ T \ S};
5.              d−max := max{d−k | k ∈ T \ S};
6.              if d+max ≤ 0 then begin
7.                    k+max := arg min{k ∈ T \ S | d+k = d+max};
8.                          T := T − k+max;
9.              end
10.              else if d−max ≤ 0 then begin
11.                    k−max := arg min{k ∈ T \ S | d−k = d−max};
12.                    S := S + k−max;
13.             end
14.             else return [S,T];
15.       end;
16. end.
```

The PP algorithm is called repeatedly by the DCA-MSF to generate a solution to the MSF instance within the prescribed accuracy level α. The pseudocode for DCA-MSF is given below. As in the case of ATSP, a good problem-specific upper bound will improve the performance of the algorithm.

Algorithm DCA-MSF$([S,T],\alpha)$
Output: $x^\alpha \in [S,T]$ such that $z(x^\alpha) \geq z^*[S,T] - \alpha$.

Code:

```
1. begin
2.        [S,T] := PP([S,T]);
```

```
3.      if T = S return S;
4.      d+max := max{d+k | k ∈ T \ S};
5.      d−max := max{d−k | k ∈ T \ S};
6.      if d+max ≤ d−max then begin
7.            if d+max ≤ α then begin
8.                  k+max := arg min{k ∈ T \ S | d+k = d+max};
9.                  return DCA-MSF([S, T − k+max], α − d+max) (* Correction *)
10.           end;
11.           else begin                                        (* Branch *)
12.                 x1 := DCA-MSF([S + k+max, T], α);
13.                 x2 := DCA-MSF([S, T − k+max], α);
14.                 if z(x1) ≥ z(x2) return x1
15.                 else return x2;
16.           end;
17.     end
18.     else begin
19.           if d−max ≤ α then begin
20.                 k−max := arg min{k ∈ T \ S | d−k = d−max};
21.                 return DCA-MSF([S + k−max, T], α − d−max) (* Correction*)
22.           end;
23.           else begin                                        (* Branch *)
24.                 x1 := DCA-MSF([S + k−max, T], α);
25.                 x2 := DCA-MSF([S, T − k−max], α);
26.                 if z(x1) ≥ z(x2) return x1
27.                 else return x2;
28.           end;
29.     end;
30. end.
```

5.2 A Data Correcting Algorithm based on Multi-Level Search

The preservation rules mentioned in Corollary 5.3 look at a level which is exactly one level deeper in the Hasse diagram than the levels of S and T. However, instead of looking one level deep we may look r levels deep in order to determine whether we can include or exclude an element. Let

$$M_r^+[S,T] \quad = \quad \{I \in [S,T] \mid |I \backslash S| \leq r\},$$

$$M_r^-[S,T] = \{I \in [S,T] | |T \backslash I| \leq r\}.$$

The set $M_r^+[S,T]$ is a collection of all sets representing solutions containing more elements than S, and which are no more than r levels deeper than S in the Hasse diagram. Similarly, the set $M_r^-[S,T]$ is a collection of all sets representing solutions containing less elements than T, and which are no more than r levels deeper than T in the Hasse diagram. Let us further define the collections of sets

$$\begin{aligned} N_r^+[S,T] &= M_r^+[S,T] \backslash M_{r-1}^+[S,T], \\ N_r^-[S,T] &= M_r^-[S,T] \backslash M_{r-1}^-[S,T]. \end{aligned}$$

The sets $N_r^+[S,T]$ and $N_r^-[S,T]$ are the collection of sets which are located exactly r levels above S and below T in the Hasse diagram, respectively.

Further, let $v_r^+[S,T] = \max\{z(I) | I \in M_r^+[S,T]\}$, $v_r^-[S,T] = \max\{z(I) | I \in M_r^-[S,T]\}$, $w_{rk}^+[S,T] = \max\{d_i^+(I) | I \in N_r^+[S+k,T]\}$ and $w_{rk}^-[S,T] = \max\{d_i^-(I) | I \in N_r^-[S,T-k]\}$.

Theorem 5.4 *Let z be a submodular function on $[S,T] \subseteq [\emptyset, N]$ with $k \in T \backslash S$ and let r be a positive integer. Then the following assertions hold.*

1. *If $|N_r^+[S+k,T]| > 0$, then $z^*[S+k,T] - \max\{z^*[S,T-k], v_r^+[S,T]\} \leq \max\{w_{rk}^+[S,T], 0\}$.*
2. *If $|N_r^-[S,T-k]| > 0$, then $z^*[S,T-k] - \max\{z^*[S+k,T], v_r^-[S,T]\} \leq \max\{w_{rk}^-[S,T], 0\}$.*

Proof: We prove only part 1 since the proof of the part 2 is similar. We may represent the partition of interval $[S, T]$ as follows:

$$[S,T] = M_r^+[S,T] \cup \bigcup_{I \in N_r^+[S,T]} [I,T].$$

Using this representation on the interval $[S+k,T]$, we have $z^*[S+k,T] = \max\{v_r^+[S+k,T], \max\{z^*[I+k,T] | I \in N_r^+[S,T]\}\}$. Let $I(k) \in \arg\max\{z^*[I+k,T] | I \in N_r^+[S,T]\}$.

There are twocases to consider: $z^*[I(k)+k,T] \geq v_r^+[S+k,T]$, and $z^*[I(k)+k,T] < v_r^+[S+k,T]$.

In the first case $z^*[S+k,T] = z^*[I(k)+k,T]$. For $I(k) \in N_r^+[S,T]$ we can apply Theorem 5.2(a) on the interval $[I(k),T]$ to obtain $z^*[I(k)+k,T] - z^*[I(k),T-k] \leq d_k^+(I(k))$, so that in this case $z^*[S+k,T] - z^*[I(k),T-k] \leq d_k^+(I(k))$. Note that for $[I(k),T-k] \subseteq [S,T-k]$ we have $z^*[S,T-k] \geq z^*[I(k),T-k]$, which implies that $z^*[S+k,T] - z^*[S,T-k] \leq d_k^+(I(k))$. Adding two maximum operations we get

$$z^*[S+k,T] - \max\{z^*[S,T-k], v_r^+[S+k,T]\} \leq \max\{d_k^+(I(k)), 0\}.$$

Since $w_{rk}^+[S,T]$ is the maximum of $d_k^+(I)$ for $I \in N_r^+[S+k,T]$, we have the required result.

In the second case $z^*[S+k,T] = v_r^+[S+k,T]$ which implies that $z^*[S+k,T] - v_r^+[S+k,T]\} = 0$ or $z^*[S+k,T] - \max\{z^*[S,T-k], v_r^+[S+k,T]\} \leq 0$. Adding a maximum operation with $w_{rk}^+[S,T]$ completes the proof. ■

Corollary 5.5 *(Preservation rules of order r). Let z be a submodular function on $[S,T] \subseteq [\emptyset,N]$ and let $k \in T \backslash S$. Then the following assertions hold.*

1. *First Preservation Rule of Order r: If $w_{rk}^+[S,T] \leq 0$, then $z^*[S,T] = \max\{z^*[S,T-k], v_r^+[S+k,T]\} \geq z^*[S+k,T]$.*
2. *Second Preservation Rule of Order r: If $w_{rk}^-[S,T] \leq 0$, then $z^*[S,T] = \max\{z^*[S+k,T], v_r^-[S,T-k]\} \geq z^*[S,T-k]$*

Notice that when we apply Corollary 5.3 to an interval, we get a reduced interval, however, when we apply Corollary 5.5, we get a value v_r in addition to a reduced interval.

It can be proved by induction that the portion of the Hasse diagram eliminated by preservation rules of order $r-1$ while searching for a maximum of the submodular function will certainly be eliminated by preservation rules of order r. In this sense, preservation rules of order r are not weaker than preservation rules of order $r-1$. (A detailed proof for the result that preservation rules of order 1 are not weaker than preservation rules of order 0, refer to Goldengorin [17]).

In order to apply Corollary 5.5, we need functions that compute the value of $w_{rk}^+[S,T]$, $w_{rk}^-[S,T]$ $v_r^+[S+k,T]$, and $v_r^-[S,T-k]$. To that end, we define two recursive functions, *PPArplus* to compute $w_{rk}^+[S,T]$ and $v_r^+[S+k,T]$, and *PPArminus* to compute $w_{rk}^-[S,T]$ and $v_r^-[S,T-k]$. The pseudocode for *PPArplus* is shown below. Its output is a 3-tuple, containing, in order,

$w^+_{rk}[S,T]$ and $v^+_r[S+k,T]$, and a solution in $M^+_r[S+k,T]$ whose objective function value is $v^+_r[S+k,T]$. The pseudocode for *PPArminus* can be constructed in a similar manner.

---

function PPArplus$([S,T],r,k)$
1. begin
2. $w := -\infty$;
3. $v := -\infty$;
4. $vset := \emptyset$;
5. $(w,v,vset) :=$ IntPPArPlus$([S+k,T],r,w,v,vset)$;
6. return $(w,v,vset)$;
7. end;

function IntPPArplus$([X,Y],r,w,v,vset)$
1. begin
2. for each $t \in Y \setminus X$ do begin
3. if $z(X+t) > v$ then begin
4. $v := z(X+t)$;
5. $vset := (X+t)$;
6. end;
7. if $d^+_t(X+t) > w$ then $w := d^+_t(X+t)$;
8. if $d^+_t(X+t) > 0$ and $r > 1$ then
9. $(w,v,vset) :=$ IntPPArPlus$([X+t,Y],r-1,w,v,vset)$;
10. end;
11. return $(w,v,vset)$;
12. end;

Note that *PPArplus* and *PPArminus* are both $\mathcal{O}(n\binom{n}{r})$, i.e. polynomial for a fixed value of r. However, in general, they are not polynomial in r.

We now use *PPArplus* and *PPArminus* to describe the Preliminary Preservation Algorithm of order r (PPAr(r)). Given a submodular function z on $[X,Y] \subseteq [\emptyset,N]$, PPAr outputs a subinterval *[S, T]* of *[X, Y]* and a set B such that $z^*[X,Y] = \max\{z^*[S,T], z(B)\}$ and $\min\{w^+_{rk}[S,T], w^-_{rk}[S,T]\} > 0$ for all $k \in T \setminus S$. At iteration i of the algorithm when the search has been restricted to $[S_i,T_i]$, starts by applying the PP algorithm (from Goldengorin *et al.* [14]) to this interval and reducing it to $[S'_i,T'_i]$. If $|T'_i \setminus S'_i| > 0$, an element $k \in T'_i \setminus S'_i$ is chosen, and the algorithm tries to apply Corollary 5.5.1 to

decide whether it belongs to the set that maximizes $z(\cdot)$ over $[S_i, T_i]$ or not. If it does, then the search is restricted to the interval $[S_i' + k, T_i']$. Otherwise, the search tries to apply Corollary 5.5.2 to decide whether the interval can be reduced to $[S_i', T_i' - k]$.

Algorithm PPAr$([S,T], r)$
Output: $x^\alpha \in [S,T]$ such that $z(x^\alpha) \geq z^*[S,T] - \alpha$.

Code:

```
1. begin
2.       X := S, Y := T; B := arg max{z(S), z(T)};
3.       while Y ≠ X do begin
4.             [S_i, T_i] := PP([X, Y]);
5.             d+ := max{d_k+(S)|k ∈ T\S};
6.             d- := max{d_k-(T)|k ∈ T\S};
7.             if d+ > d- then begin
8.                   k := arg max{d_t+(S)|t ∈ T\S};
9.                   (w, v, vset) := PPArplus([S_i, T_i], r, k);
10.                  if v > z(B) then B := vset;
11.                  if w ≤ 0 then Y := T_i - k;
12.                  else return ([S_i, T_i], B);
13.            else begin
14.                  k := arg max{d_t-(S)|t ∈ T\S};
15.                  (w, v, vset) := PPArminus([S_i, T_i], r, k);
16.                  if v > z(B) then B := vset;
17.                  if w ≤ 0 then X := S_i + k;
18.                  else return ([S_i, T_i], {w_ri+[S_i, T_i]}, {w_ri-[S_i, T_i]}, B);
19.            end;
20.      end;
21. end.
```

It is clear that if $r = |T \setminus S|$, PPAr will always find an optimal solution to our problem. However, PPAr is not a polynomial in r, and so PPAr with a large r is not practically useful.

We can embed PPAr in a branch and bound framework to describe DCA-MSFr, a data correcting algorithm based on PPAr. It is similar to the DCA-MSF proposed in Goldengorin *et al.* [14]. For DCA-MSFr we are

given a submodular function z to be maximized over an interval $[S, T]$, and an accuracy parameter α, and we need to find a solution such that the difference between the objective function values of the solution output by DCA-MSFr and the optimal solution will not exceed α.

Notice that for a submodular function z, PPAr with a fixed r may terminate with $T \neq S$ and $\min\{w^+_{ri}[S,T], w^-_{ri}[S,T] \mid i \in T \backslash S\} = \omega > 0$. The basic idea behind DCA-MSFr is that if this situation occurs, then the data of the current problem is corrected in such a way that ω is non-positive for the corrected function and PPAr can continue. Moreover, each correction of z needs to be carried out in such a way that the corrected function remains submodular. The attempted correction is carried out implicitly, in a manner similar to the one in Goldengorin *et al.* [14] but using Corollary 5.5 instead of Corollary 5.3. Thus, for example, if $w^+_{rj}[S,T] = \omega \leq \alpha$, then PPAr is allowed to continue, but the accuracy parameter reduced to $\alpha - \omega$.

If such a correction is not possible, i.e. if ω exceeds the accuracy parameter, then we branch on a variable $k \in \arg\max\{d^+_i(S), d^-_i(T) | i \in T \setminus S\}$ to partition the interval $[S, T]$ into two intervals $[S + k, T]$ and $[S, T - k]$. This branching rule was proposed in Goldengorin [10]. An upper bound for the value of z for each of the two intervals is then computed to see if either of the two can be pruned. We use an upper bound due to Khachaturov [23] described as follows. Let $d^+(S,T) = \{d^+_i(S) | d^+_i(S) > 0, i \in T \backslash S\}$ and $d^-(S,T) = \{d^-_i(T) | d^-_i(T) > 0, i \in T \backslash S\}$. Further let $d^+[i]$ (respectively $d^-[i]$) denote the ith largest element of $d^+(S,T)$ (respectively $d^-(S,T)$). Then *ub* described below is an upper bound to $z^*[S,T]$.

$$ub[S,T] = \max\{\min_{i=1,\ldots,|T\backslash S|}\{z(S) + \sum_{j=1}^{i} d^+[j], z(T) + \sum_{j=1}^{i} d^-[j]\}\}.$$

The following pseudocode describes DCA-MSFr formally.

Algorithm DCA-MSFr$([S,T], \alpha, r)$

1. begin
2. $\quad best\_set := \arg\max\{z(S), z(T)\}$;
3. $\quad best := z(best\_set)$;
4. $\quad (best\_set, best) :=$ IntDCA-MSFr$([S,T], \alpha, r, best_set, best)$;
5. $\quad$ return $best\_set$;
6. end.

function IntDCA-MSFr$([S,T], \alpha, r, best\_set, best)$

```
1. begin
2.       ([S,T],{w_rk^+},{w_rk^-},B] :=PPAr([S,T],r);
3.       if z(B) > best then begin
4.             best_set := B;
5.             best := z(B);
6.       end;
7.       if S = T return (best_set, best);
8.       ω^+ := max{w_rk^+[S,T]|k ∈ T \ S};
9.       choose k^+ from min{k|w_rk^+[S,T] = ω^+, k ∈ T \ S};
10.      ω^- := max{w_rk^-[S,T]|k ∈ T \ S};
11.      choose k^- from min{k|w_rk^-[S,T] = ω^-, k ∈ T \ S};
12.      if ω^+ ≤ α then                                   (* Correction *)
13.            IntDCA-MSFr([S + k^+, T], α − ω^+, r, best_set, best);
14.      else if ω^- ≤ α then                              (* Correction *)
15.            IntDCA-MSFr([S, T − k^-], α − ω^-, r, best_set, best);
16.      else begin                  (* Branch [S,T] → [S + k, T], [S, T − k] *)
17.            choose k from arg max{d_i^+(S), d_i^-(T)|i ∈ T \ S};
18.            if ub[S + k, T] > best then begin           (* Bound *)
19.                  (bs_1, b_1) :=IntDCA-MSFr([S + k, T], α, r, best_set, best);
20.                  if b_1 > best then begin
21.                        best_set := bs_1;
22.                        best := b_1;
23.                  end;
24.            end;
25.            if ub[S, T − k] > best then begin           (* Bound *)
26.                  (bs_2, b_2) :=IntDCA-MSFr([S, T − k], α, r, best_set, best);
27.                  if b_2 > best then begin
28.                        best_set := bs_2;
29.                        best := b_2;
30.                  end;
31.            end;
32.      end;
33. end;
```

5.3 Computational Experience with Quadratic Cost Partition Instances

In this section we report our computational experience with DCA-MSFr. We choose the quadratic cost partition problem as a test bed. The quadratic cost partition (QCP) problem can be described as follows (see e.g., Lee *et al.* [27]). Given nonnegative real numbers q_{ij} and real numbers p_i with $i, j \in N = \{1, 2, \ldots, n\}$, the QCP is the problem of finding a subset $S \subseteq N$ such that the function $z(S) = \sum_{j \in S} p_i - \frac{1}{2} \sum_{i,j \in S} q_{ij}$ will be maximized. The density d of a QCP instance is the ratio of the number of finite q_{ij} values to $n(n-1)/2$, and is expressed as a percentage. It is proved in Theorem 2.2 of Lee *et al.* [27] that $z(\cdot)$ is submodular.

In Goldengorin *et al.* [14] computational experiments with QCP have been restricted to instances of size not more than 80, because instances of that size have been considered in Lee *et al.* [27]. For these instances, it was shown that the average calculation times grow exponentially when the number of vertices increases and reduce exponentially with increasing density.

Herein we report the performance of DCA-MSFr on QCP instances of varying size and densities. The maximum time that we allow for an instance is 10 CPU minutes on an personal computer running on a 300MHz Pentium processor with 64 MB memory. The algorithms have been implemented in Delphi 3.

The instances we test our algorithms on are statistically similar to the instances in Lee *et al.* [27]. Instances of size n and density $d\%$ are generated as follows. A graph with n nodes and $\frac{d}{100} \times \frac{n(n-1)}{2}$ random edges is generated. The edges are assigned costs from a $\mathcal{U}[1, 100]$ distribution. n edges connect each node to itself, and these edges are assigned costs from a $\mathcal{U}[0, 100]$ distribution. The distance matrix of this graph forms a QCP instance.

We first report the effect of varying the value of r on the performance of DCA-MSFr(r). It is intuitive that DCA-MSFr(r) will require more execution times when the value of r increases. Our computation experience with 10 QCP instances of size 100 and different densities is shown in Figures 8-10. Figure 8 shows the number of subproblems generated when r is increased from a value of 0 (i.e. DCA-MSF) to 5. As is intuitive, the number of subproblems reduce with increasing r for all density values. Figure 9 shows the execution times of DCA-MSFr(r) with varying d and r values. Recall that when the value of r increases, the time required at each subproblem

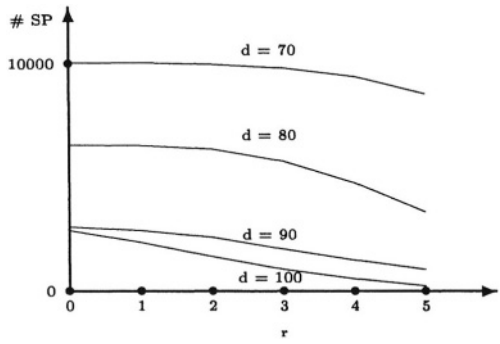


Figure 8: Average number of subproblems generated against r for QCP instances with $n = 100$ and varying d values

increases, since PPAr requires more computations for larger r values. The decrease in the number of subproblems approximately balance the increase in the time at each subproblem for r values in the range 0 through 4. When $r = 5$, the computation times for DCA-MSFr(r) increase significantly for all densities. From Figure 9 it seems that for dense graphs, r values of 3 or 4 are most favorable. This effect also holds for larger instances — Figure 10 shows the execution times for instances with $n = 200$ and $d = 100$.

We next report the results of our experiments to solve large sized QCP instances with DCA-MSFr(r). Using results obtained from the previous part of our study, we choose to use DCA-MSFr(3) as our algorithm of choice. We consider instances of the QCP with size n ranging from 100 to 500 and densities varying between 10% and 100%. We try to solve these instances exactly ($\alpha_0 = 0\%$), and with a prescribed accuracy $\alpha_0 = 5\%$ within 10 minutes. We report in Tables 1 and 2 the average execution times in seconds for exact and approximate solutions with DCA-MSFr(3) and DCA-MSF. The entries marked '*' could not be solved within 10 minutes. From the table, we note that the execution times increase exponentially with increasing problem size and decreasing problem densities. Therefore QCP instances with 500 vertices and densities between 90% and 100% are the largest instances which can be solved by the DCA-MSFr(3) within 10 minutes on a standard personal computer. We also see that on an average DCA-MSFr(3) takes roughly 11% of the time taken by DCA-MSF for the exact solutions, and

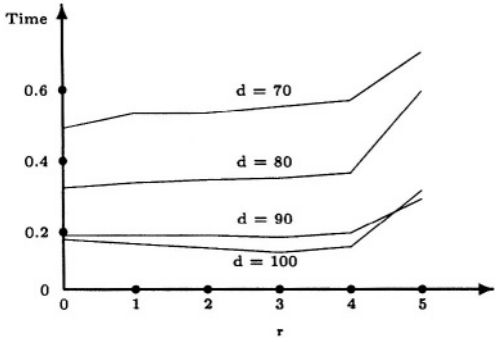


Figure 9: Average execution time (in seconds) against r for QCP instances with $n = 100$ and varying d values

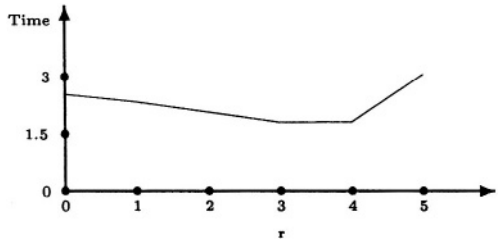


Figure 10: Average execution time (in seconds) against r for QCP instances with $n = 200$ and $d = 100$

roughly 13% of the time taken by DCA-MSF for the approximate solutions. The reduction in time is more pronounced for problems with higher size and higher densities.

Table 1: Average execution times on QCP instances when $\alpha = 0\%$

		Problem size (n)				
Density(%)	Algorithm	100	200	300	400	500
100	DCA-MSFr(3)	0.098	2.63	18.316	85.827	229.408
	DCA-MSF	0.831	64.372	340.681	1894.811	*
90	DCA-MSFr(3)	0.138	3.824	37.931	173.063	624.925
	DCA-MSF	1.027	78.614	794.037	3505.892	*
80	DCA-MSFr(3)	0.28	9.506	98.69	679.914	*
	DCA-MSF	1.784	217.898	2681.973	*	*
70	DCA-MSFr(3)	0.393	17.643	413.585	*	*
	DCA-MSF	2.498	631.492	*	*	*
60	DCA-MSFr(3)	0.731	86.33	*	*	*
	DCA-MSF	3.509	1414.103	*	*	*
50	DCA-MSFr(3)	1.752	345.723	*	*	*
	DCA-MSF	9.382	*	*	*	*
40	DCA-MSFr(3)	3.457	*	*	*	*
	DCA-MSF	17.245	*	*	*	*
30	DCA-MSFr(3)	11.032	*	*	*	*
	DCA-MSF	48.013	*	*	*	*
20	DCA-MSFr(3)	47.162	*	*	*	*
	DCA-MSF	195.82	*	*	*	*
10	DCA-MSFr(3)	70.081	*	*	*	*
	DCA-MSF	446.293	*	*	*	*

6 The Simple Plant Location Problem

The Simple Plant Location Problem (SPLP) takes a set $I = \{1, 2, \ldots, m\}$ of sites in which plants can be located, a set $J = \{1, 2, \ldots, n\}$ of clients, each having a unit demand, a vector $F = (f_i)$ of fixed costs for setting up plants at sites $i \in I$, and a matrix $C = [c_{ij}]$ of transportation costs from $i \in I$ to $j \in J$ as input. It computes a set P^*, $\emptyset \subset P^* \subseteq I$, at which plants can be located so that the total cost of satisfying all client demands is minimal. The costs involved in meeting the client demands include the fixed costs of setting up plants, and the transportation cost of supplying clients from the plants that are set up. A detailed introduction to this problem has appeared in Cornuejols *et al.* [6], which also classifies the problem as NP-hard. The

Table 2: Average execution times on QCP instances when $\alpha = 5\%$

		Problem size (n)				
Density(%)	Algorithm	100	200	300	400	500
100	DCA-MSFr(3)	0.094	2.444	17.179	85.096	222.883
	DCA-MSF	0.752	38.351	229.396	1162.396	*
90	DCA-MSFr(3)	0.118	3.607	34.972	166.996	608.755
	DCA-MSF	0.916	49.926	583.754	1996.544	*
80	DCA-MSFr(3)	0.228	8.186	89.685	580.789	*
	DCA-MSF	1.108	162.455	1875.603	3604.715	*
70	DCA-MSFr(3)	0.304	15.693	364.48	*	*
	DCA-MSF	1.593	376.629	3165.384	*	*
60	DCA-MSFr(3)	0.517	72.931	*	*	*
	DCA-MSF	2.874	895.426	*	*	*
50	DCA-MSFr(3)	1.298	267.445	*	*	*
	DCA-MSF	5.931	1937.673	*	*	*
40	DCA-MSFr(3)	2.179	*	*	*	*
	DCA-MSF	10.327	*	*	*	*
30	DCA-MSFr(3)	5.88	*	*	*	*
	DCA-MSF	22.209	*	*	*	*
20	DCA-MSFr(3)	17.477	*	*	*	*
	DCA-MSF	74.841	*	*	*	*
10	DCA-MSFr(3)	12.196	*	*	*	*
	DCA-MSF	95.122	*	*	*	*

objective function of the SPLP is supermodular, but we do not use the results of the previous section explicitly in this section.

In applying data correcting to the SPLP, we work with a pseudo-Boolean formulation of the problem. We show how data correcting can be used to preprocess SPLP instances efficiently, and then to solve the problem.

6.1 A Pseudo-Boolean Formulation of the SPLP

The pseudo-Boolean approach to solving the SPLP (Hammer [21], Beresnev [4]) is a penalty-based approach that relies on the fact that any instance of the SPLP has an optimal solution in which each client is supplied by exactly one plant. This implies, that in an optimal solution, each client will be served fully by the plant located closest to it. Therefore, it is sufficient to determine the sites where plants are to be located, and then use a minimum cost assignment of clients to plants.

An instance of the SPLP can be described by a m-vector $F = (f_i)$,

and a $m \times n$ matrix $C = [c_{ij}]$; $m, n \geq 1$. We will use the $m \times (n+1)$ *augmented matrix* $[F|C]$ as a shorthand for describing an instance of the SPLP. The total cost $f_{[F|C]}(P)$ associated with a subset P of I consists of two components, namely the fixed costs $\sum_{i \in P} f_i$ and the transportation costs $\sum_{j \in J} \min\{c_{ij} | i \in P\}$; i.e.

$$f_{[F|C]}(P) = \sum_{i \in P} f_i + \sum_{j \in J} \min\{c_{ij} | i \in P\},$$

and the SPLP is the problem of finding

$$P^* \in \arg\min\{f_{[F|C]}(P) | \emptyset \subset P \subseteq I\}. \tag{4}$$

In the remainder of this subsection we describe the pseudo-Boolean formulation of the SPLP due to Hammer [21].

A $m \times n$ *ordering matrix* $\Pi = [\pi_{ij}]$ is a matrix each of whose columns $\Pi_j = (\pi_{1j}, \ldots, \pi_{mj})^T$ define a permutation of $1, \ldots, m$. Given a transportation matrix C, the set of all ordering matrices Π such that $c_{\pi_{1j}j} \leq c_{\pi_{2j}j} \leq \cdots \leq c_{\pi_{mj}j}$ for $j = 1, \ldots, n$, is denoted by *perm(C)*.

Defining for each $i = 1, \ldots, m$

$$y_i = \begin{cases} 0 & \text{if } i \in P \\ 1 & \text{otherwise,} \end{cases} \tag{5}$$

we can indicate any solution P by a vector $\overline{y} = (y_1, y_2, \ldots, y_m)$. The fixed cost component of the total cost can be written as

$$F_F(\overline{y}) = \sum_{i=1}^{m} f_i(1 - y_i). \tag{6}$$

Given a transportation cost matrix C, and an ordering matrix $\Pi \in perm(C)$, we can denote differences between the transportation costs for each $j \in J$ as

$$\begin{aligned} \Delta c[0, j] &= c_{\pi_{1j}j}, \quad \text{and} \\ \Delta c[l, j] &= c_{\pi_{(l+1)j}j} - c_{\pi_{lj}j}, \quad l = 1, \ldots, m-1. \end{aligned}$$

Note that $\Delta c[l, j] \geq 0$, even if the transportation cost matrix C contains negative entries. The transportation costs of supplying any client $j \in J$ from any open plant can be expressed in terms of the $\Delta c[\cdot, j]$ values. It is

clear that we have to spend at least $\Delta c[0,j]$ in order to satisfy j's demand, since this is the cheapest cost of satisfying j. If no plant is located at the site closest to j, i.e. $y_{\pi_{1j}} = 1$, we try to satisfy the demand from the next closest site. In that case, we spend an additional $\Delta c[1,j]$. Continuing in this manner, the transportation cost of supplying $j \in J$ is

$$\begin{aligned}\min\{c_{ij} | i \in P\} &= \Delta c[0,j] + \Delta c[1,j] \cdot y_{\pi_{1j}} + \Delta c[2,j] \cdot y_{\pi_{1j}} \cdot y_{\pi_{2j}} \\ &\quad + \cdots + \Delta c[m-1,j] \cdot y_{\pi_{1j}} \cdots y_{\pi_{(m-1)j}} \\ &= \Delta c[0,j] + \sum_{k=1}^{m-1} \Delta c[k,j] \cdot \prod_{r=1}^{k} y_{\pi_{rj}},\end{aligned}$$

so that the transportation cost component of the cost of a solution $\overline{y}$ corresponding to an ordering matrix $\Pi \in perm(C)$ is

$$T_{C,\Pi}(\overline{y}) = \sum_{j=1}^{n} \left\{ \Delta c[0,j] + \sum_{k=1}^{m-1} \Delta c[k,j] \cdot \prod_{r=1}^{k} y_{\pi_{rj}} \right\}. \tag{7}$$

Combining (6) and (7), the total cost of a solution $\overline{y}$ to the instance $[F|C]$ corresponding to an ordering matrix $\Pi \in perm(C)$ is given by the pseudo-Boolean polynomial

$$\begin{aligned}f_{[F|C],\Pi}(\overline{y}) &= F_F(\overline{y}) + T_{C,\Pi}(\overline{y}) \\ &= \sum_{i=1}^{m} f_i(1 - y_i) + \\ &\quad \sum_{j=1}^{n} \left\{ \Delta c[0,j] + \sum_{k=1}^{m-1} \Delta c[k,j] \cdot \prod_{r=1}^{k} y_{\pi_{rj}} \right\}.\end{aligned} \tag{8}$$

It can be shown (see Goldengorin *et al.* [15]) that the total cost function $f_{[F|C],\Pi}(\cdot)$ is identical for all $\Pi \in perm(C)$. We call this pseudo-Boolean polynomial the *Hammer function* $H_{[F|C]}(\overline{y})$ corresponding to the SPLP instance $[F|C]$ and $\Pi \in perm(C)$. In other words

$$H_{[F|C]}(\overline{y}) = f_{[F|C],\Pi}(\overline{y}) \text{ where } \Pi \in perm(C). \tag{9}$$

We can formulate (4) in terms of Hammer functions as

$$\overline{y^*} \in \arg\min\{H_{[F|C]}(\overline{y}) | \overline{y} \in \{0,1\}^m, \overline{y} \neq \overline{1}\}. \tag{10}$$

As an example, consider the SPLP instance:

$$[F|C] = \left[\begin{array}{c|cccc} 9 & 7 & 12 & 22 & 13 \\ 4 & 8 & 9 & 18 & 17 \\ 3 & 16 & 17 & 10 & 27 \\ 6 & 9 & 13 & 10 & 11 \end{array}\right]. \tag{11}$$

Two possible ordering matrices corresponding to C are

$$\Pi_1 = \begin{bmatrix} 1 & 2 & 3 & 4 \\ 2 & 1 & 4 & 1 \\ 4 & 4 & 2 & 2 \\ 3 & 3 & 1 & 3 \end{bmatrix} \text{ and } \Pi_2 = \begin{bmatrix} 1 & 2 & 4 & 4 \\ 2 & 1 & 3 & 1 \\ 4 & 4 & 2 & 2 \\ 3 & 3 & 1 & 3 \end{bmatrix}. \tag{12}$$

The Hammer function is $H_{[F|C]}(\overline{y}) = \{9(1-y_1)+4(1-y_2)+3(1-y_3)+6(1-y_4)\} + \{7+1y_1+1y_1y_2+7y_1y_2y_4\} + \{9+3y_2+1y_1y_2+4y_1y_2y_4\} + \{10+0y_3+8y_3y_4+4y_2y_3y_4\} + \{11+2y_4+4y_1y_4+10y_1y_2y_4\} = 59-8y_1-y_2-3y_3-4y_4+2y_1y_2+4y_1y_4+8y_3y_4+21y_1y_2y_4+4y_2y_3y_4$.

6.2 Preprocessing SPLP instances

The first preprocessing rules for the SPLP involving both fixed costs and transportation costs appeared in Khumawala [24]. In terms of Hammer functions, these rules are stated in the following theorem. We assume (without loss of generality) that we cannot partition I into sets I_1 and I_2, and J into sets J_1 and J_2, such that the transportation costs from sites in I_1 to clients in J_2, and from sites in I_2 to clients in J_1 are not finite. We assume too, that the site indices are arranged in non-increasing order of $f_i + \sum_{j \in J} c_{ij}$ values.

Theorem 6.1 *Let $H_{[F|C]}(\overline{y})$ be the Hammer function corresponding to the SPLP instance $[F|C]$ in which like terms have been aggregated. For each site index k, let a_k be the coefficient of the linear term corresponding to y_k and let be the sum of the coefficients of all non-linear terms containing y_k. Then the following assertion holds.*

RO: *If $a_k \geq 0$, then there is an optimal solution $\overline{y^*}$ in which $y_k^* = 0$, else*

RC: *If $a_k + t_k \leq 0$, then there is an optimal solution $\overline{y^*}$ in which $y_k^* = 1$, provided that $y_i^* \neq 1$ for some $i \neq k$.*

Notice that RO and RC primarily try to either open or close sites. If it succeeds, it also changes the Hammer function for the instance, reducing the number of non-linear terms therein. In the remaining portion of this subsection, we describe a completely new reduction procedure (RP), whose primary aim is to reduce the coefficients of terms in the Hammer function, and if we can reduce it to zero, to eliminate the term from the Hammer function. This procedure is based on fathoming rules of branch and bound algorithms and data correcting principles.

Let us assume that we have an upper bound (UB) on the cost of an optimal solution for the given SPLP instance. This can be obtained by running a heuristic on the problem data. Now consider a non-linear term $s \cdot \prod_{r=1}^{k} y_{\pi_{rj}}$ in the Hammer function. This term will contribute to the cost of a solution, only if plants are *not* located in any of the sites $\pi_{1j}, \ldots, \pi_{kj}$. Let LB be a lower bound on the cost of solutions in which facilities are not located in sites $\pi_{1j}, \ldots, \pi_{kj}$. If $LB \leq UB$, then we cannot make any judgement about this term. On the other hand, if $LB > UB$, then we know that there cannot be an optimal solution with $y_{\pi_{1j}} = \ldots = y_{\pi_{kj}} = 1$. In this case, if we reduce the coefficient s by $LB - UB - \varepsilon$, ($\varepsilon > 0$, small), then the new Hammer function and the original one have identical sets of optimal solutions. If after the reduction, s is non-positive, then the term can be removed from the Hammer function. Such changes in the Hammer function alter the values of t_k, and can possibly allow us to use Khumawala's rules to close certain sites. Once some sites are closed, some of the linear terms in the Hammer function change into constant terms, and some of the quadratic terms change into linear ones. These changes cause changes in both the a_k and the t_k values, and can make further application of Khumawala's rules possible, thus preprocessing some other sites, and making further changes in the Hammer function. A pseudocode of the reduction procedure (RP) is provided below.

Procedure RP$(H_{[F|C]}(\overline{y}))$
Output: A preprocessed instance of the SPLP, i.e. an equivalent instance of reduced size, and decisions to either locate or not locate plants in some of the sites.

Code:

```
1. begin
2.       repeat
3.             compute an upper bound UB for the instance;
```

4. for each nonlinear term $s \cdot \prod_{r=1}^{\kappa} y_{\pi_{rj}}$ in $H_{[F|C]}(\overline{y})$ do
5. begin
6. compute lower bound LB on the cost of solutions in
7. which plants are not located in sites $\pi_{1j}, \ldots, \pi_{kj}$;
8. if $LB > UB$ then
9. reduce the coefficient of the term by
10. $\max\{s, LB - UB - \varepsilon\}$;
11. apply Khumawala's rules until no further preprocessing is possible;
12. recompute the Hammer function $H_{[F|C]}(\overline{y})$;
13. until no further preprocessing of sites was achieved in the current iteration;
14. end;

Let us consider the application of all preprocessing rules to the example with the Hammer function $H_{[F|C]}(\overline{y}) = 59 - 8y_1 - y_2 - 3y_3 - 4y_4 + 2y_1y_2 + 4y_1y_4 + 8y_3y_4 + 21y_1y_2y_4 + 4y_2y_3y_4$. The values of a_k, t_k and $a_k + t_k$ are as follows:

k :	1	2	3	4
a_k :	−8	−1	−3	−4
t_k :	27	27	12	37
$a_k + t_k$:	19	26	9	33

It is clear that neither RO nor RC is applicable here, since the coefficient of the term $21y_1y_2y_4$ is too large. Therefore, we try to reduce this coefficient by applying the RP.

An upper bound of $UB = 51$ to the original problem can be obtained by setting $y_1 = y_4 = 1$ and $y_2 = y_3 = 0$. A lower bound to the problem under the restriction $y_1 = y_2 = y_4 = 1$ is 73, since $H_{[F|C]}(1, 1, 0, 1) = 73$. Using RP therefore, we can reduce the coefficient of $21y_1y_2y_4$ by $73 - 51 - \varepsilon = 20$, so that the new Hammer function with the same set of optimal solutions as the original function becomes $H'(\overline{y}) = 59 - 8y_1 - y_2 - 3y_3 - 4y_4 + 2y_1y_2 + 4y_1y_4 + 8y_3y_4 + 1y_1y_2y_4 + 4y_2y_3y_4$. The updated values of a_k, t_k, and $a_k + t_k$ are presented below.

k :	1	2	3	4
a_k :	−8	−1	−3	−4
t_k :	7	7	12	17
$a_k + t_k$:	−1	6	9	13

RC can immediately be applied in this situation to set $y_3 = 1$. Updating $H'(\overline{y})$, we can apply RO and set $y_2 = y_4 = 0$. This allows us to apply RC again to set $y_3 = 1$, thus giving us an optimal solution (i.e. (1,0,1,0)) to the instance, with a cost of 48.

6.3 The Data Correcting Algorithm

The basic idea behind the data correcting algorithm is to modify the Hammer function in a way, such that the RO and RC rules can be applied to the modified instance. While modifying the instance, care is taken so that an optimal solution to the modified instance is not too sub-optimal for the original instance. We make use of the following suitably modified version of Lemma 3.1 for this problem.

Lemma 6.2 *Consider two Hammer functions $H_1(\overline{y})$ and $H_2(\overline{y})$. Let $\overline{y_1^*}$ and $\overline{y_2^*}$ be the optimal solutions to $H_1(\overline{y})$ and $H_2(\overline{y})$ respectively. Then*

$$H_1(\overline{y_1^*}) - H_1(\overline{y_2^*}) \leq \sum_{i=0}^{m-1} \sum_{j=1}^{n} |\Delta c^1[i,j] - \Delta c^2[i,j]|.$$

Consider a SPLP instance with accuracy parameter α in which RO and RC cannot be applied. Clearly, in the Hammer function for this instance, $a_k < 0$ and $a_k + t_k > 0$ for all k. Let $k_0 = \arg\min\{|a_k|, a_k + t_k\}$. Also let $|a_{k_0}| \leq a_{k_0} + t_{k_0}$. In this case, if we change (correct) the Hammer function of the instance by increasing the coefficient of y_{k_0} to zero, then RO can be applied to the corrected instance and preprocessing can continue. However, this is allowed only if $\min\{|a_{k_0}|, a_{k_0} + t_{k_0}\} \leq \alpha$. In such a situation, if $|a_{k_0}| > a_{k_0} + t_{k_0}$, then the instance can be corrected by decreasing the coefficient of y_{k_0} by $a_{k_0} + t_{k_0}$; then RC can be applied to the corrected instance and preprocessing can continue. Notice that while correcting the instance, we allow for suboptimality to the extent of $|a_{k_0}|$ in the first case, and $a_{k_0} + t_{k_0}$ in the second case. Thus, the accuracy parameter for the corrected instance is reduced appropriately.

It may happen however, that $\min\{|a_{k_0}|, a_{k_0} + t_{k_0}\} > \alpha$. In that case, correction is not possible at this stage and the problem has to be broken down into subproblems. This is done by a branching operation. Goldengorin *et al.* [16] suggest that the algorithm branches on an index from $\arg\max\{t_k\}$.

The logic behind this rule is the following. A plant would have been located in this site in an optimal solution if the coefficient of linear term involving y_k in the Hammer function would have been increased by $-a_k$. We could have predicted that a plant would not be located there if the same coefficient would have been decreased by $t_k + a_k$. Therefore we could use the average of $-a_k$ and $a_k + t_k$ as a measure of the chance that we will *not* be able to predict the fate of site k in any subproblem of the current subproblem. If we want to reduce the size of the branch and bound tree by assigning values to such variables, then we can think of a branching function that branches on the index k_0 with the largest average value, i.e. the largest value of $-a_k + (a_k + t_k)$, i.e. the largest value of t_k.

On the basis of the discussion above, the pseudocode for a data correcting algorithm for SPLP is given below. It works by maintaining three sets, Ω containing the sites where facilities are to be located, Λ containing the sites where facilities are not to be located, and Ψ containing the rest of the sites. RO and RC rules are assumed to be able to manipulate these three sets.

Algorithm DCA-SPLP$(H_{[F|C]}(\overline{y}), \alpha)$
Output: A solution $\overline{y^\alpha}$ to the SPLP such that $H_{[F|C]}(\overline{y^\alpha}) \leq H_{[F|C]}(\overline{y^*}) + \alpha$.

Code:

```
1. begin
2.       apply RP to initialize Ω, Λ and Ψ;
3.       y^α := Int-DCA-SPLP(Ω, Λ, Ψ, H_[F|C](·), α);
4.       return y^α;
5. end.
```

Function Int-DCA-SPLP$(\Omega, \Lambda, \Psi, H_{[F|C]}(\cdot), \alpha)$

```
1. begin
2.       while RO or RC is applicable
3.             update Ω, Λ, Ψ, and H_[F|C](·) by applying RO and RC;
4.       k* ∈ arg{k ∈ Ψ| min{|a_k|, a_k + t_k} =
...            min{min{|a_s|, a_s + t_s}|s ∈ Ψ}};
5.       if |a_k*| ≤ a_k* + t_k* then begin
6.             if |a_k*| ≤ α then                    (* Correct *)
```

```
7.                    return Int-DCA-SPLP(Ω+k*, Λ, Ψ−k*, H_[F|C](·), α−|a_k*|);
8.              else begin                                          (* Branch *)
9.                    k^b := arg max{t_k|k ∈ Ψ};
10.                   ȳ1 :=Int-DCA-SPLP(Ω + k^b, Λ, Ψ − k^b, H_[F|C](·), α);
11.                   ȳ2 :=Int-DCA-SPLP(Ω, Λ + k^b, Ψ − k^b, H_[F|C](·), α);
12.                   return arg max{H_[F|C](ȳ1), H_[F|C](ȳ2)};
13.             end;
14.       else begin
15.             if a_k* + t_k* ≤ α then                             (* Correct *)
16.       return Int-DCA-SPLP(Ω, Λ + k*, Ψ − k*, H_[F|C](·), α − a_k* − t_k*);
17.             else begin                                          (* Branch *)
18.                   k^b := arg max{t_k|k ∈ Ψ};
19.                   ȳ1 :=Int-DCA-SPLP(Ω + k^b, Λ, Ψ − k^b, H_[F|C](·), α);
20.                   ȳ2 :=Int-DCA-SPLP(Ω, Λ + k^b, Ψ − k^b, H_[F|C](·), α);
21.                   return arg max{H_[F|C](ȳ1), H_[F|C](ȳ2)};
22.             end;
23.       end;
24. end;
```

6.4 Computational Experience with SPLP Instances

We report our computational experience with the DCA-SPLP on several benchmark instances of the SPLP in the remainder of this section. The performance of the algorithm is compared with that of the algorithms described in the papers that suggested these instances. We used one of two bounds in the implementations of RP and DCA-SPLP: a combinatorial Khachaturov-Minoux bound (Khachaturov [22] and Minoux [29]); and a much stronger Erlenkotter bound based on a LP dual-ascent algorithm (Erlenkotter [7]). We implemented the DCA-SPLP in PASCAL, compiled it using Prospero Pascal, and ran it on a 733 MHz Pentium III machine. The computation times we report are in seconds on our machine.

6.4.1 Testing the Effectiveness of the Reduction Procedure RP

Given an instance of the SPLP, the reduction procedure RP reduces it to a smaller core instance by making decisions to locate or not locate plants in several sites. The effectiveness of the RP can thus be measured either by computing the number of free locations in the core instance, or by computing

the number of non-zero nonlinear terms present in the Hammer function of the core instance. Tables 3 and 4 shows how the various methods of reduction perform on the benchmark SPLP instances in the OR-Library (Beasley [3]). In the tables, procedure (a) refers to the use of the "delta" and "omega" rules from Khumawala [24], procedure (b) to the RP with the Khachaturov-Minoux combinatorial bound to obtain a lower bound, and procedure (c) to the RP with the Erlenkotter bound to obtain a lower bound.

Table 3: Number of free locations after preprocessing SPLP instances in the OR-Library

Problem	m	n	Procedure		
			a	b	c
cap71	16	50	4	0	0
cap72	16	50	6	0	0
cap73	16	50	6	3	3
cap74	16	50	2	0	0
cap101	25	50	9	0	0
cap102	25	50	13	3	0
cap103	25	50	14	0	0
cap104	25	50	12	0	0
cap131	50	50	34	32	8
cap132	50	50	27	25	5
cap133	50	50	25	19	10
cap134	50	50	19	0	0

The existing preprocessing rules due to Khumawala [24] and Goldengorin *et al.* [15] (i.e. procedure (a), which was used in the SPLP example in Goldengorin *et al.* [14]) cannot solve any of the OR-Library instances to optimality. However, the variants of the new reduction procedure (i.e. procedures (b) and (c)) solve a large number of these instances to optimality. Procedure (c), based on the Erlenkotter bound is marginally better than procedure (b) in terms of the number of free locations (Table 3), but substantially better in terms of the number of non-zero nonlinear terms in the Hammer function (Table 4).

Tables 3 and 4 also demonstrate the superiority of the new preprocessing rule over the "delta" and "omega" rules. Consider for example the problem cap132. The "delta" and "omega" rules reduce the problem size from $m = 50$ and 2389 non-zero nonlinear variables to $m = 27$ and 112 non-zero nonlinear

Table 4: Number of non-zero nonlinear terms in the Beresnev function after preprocessing SPLP instances in the OR-Library

Problem	Non-zero terms before preprocessing	Procedure		
		a	b	c
cap71	699	6	0	0
cap72	699	12	0	0
cap73	699	13	2	2
cap74	699	1	0	0
cap101	1147	24	0	0
cap102	1147	33	2	0
cap103	1147	38	0	0
cap104	1147	29	0	0
cap131	2389	163	135	8
cap132	2389	112	92	3
cap133	2389	101	60	11
cap134	2389	62	0	0

variables. However, the new preprocessing rule reduces the same problem to one having $m = 5$ and 3 non-zero nonlinear variables!

6.4.2 Bilde and Krarup-type Instances

These are the earliest benchmark problems that we consider here. The exact instance data is not available, but the process of generating the problem instances is described in Bilde and Krarup [5]. There are 22 different classes of instances and in this subsection we use the nomenclature used in Bilde and Krarup [5]. In our experiments we generated 10 instances for each of the types of problems, and used the mean values of our solutions to evaluate the performance of our algorithm with the one used in Bilde and Krarup [5]. In our implementation, we used the Khachaturov-Minoux combinatorial bound in the reduction procedure RP as well as in the DCA-SPLP.

The reduction procedure was not useful for these instances, but the DCA-SPLP could solve all the instances in reasonable time. The results of our experiments are presented in Table 5. The performance of the algorithm implemented in Bilde and Krarup [5] was measured in terms of the number of branching operations performed by the algorithm and its execution time in CPU seconds on a IBM 7094 machine. We estimate the number of branching operations by our algorithm as the logarithm (to the base 2) of the number of subproblems it generated. From the table we see that the DCA-SPLP

Table 5: Results from Bilde and Krarup-type instances

Problem	DCA		Bilde and Krarup	
Type	Branching	CPU time	Branching	CPU Time†
B	11.72	0.67	43.3	4.33
C	17.17	14.81	⋆	>250
D1	13.80	0.65	216	11
D2	12.13	0.38	218	24
D3	10.87	0.19	169	19
D4	10.25	0.15	141	17
D5	9.24	0.07	106	14
D6	8.99	0.09	101	15
D7	8.79	0.09	83	13
D8	8.60	0.09	55	11
D9	8.15	0.07	47	11
D10	7.29	0.03	43	11
E1	18.66	35.28	1271	202
E2	16.14	8.64	1112	172
E3	14.59	3.81	384	82
E4	13.65	2.74	258	65
E5	12.73	2.01	193	53
E6	11.82	0.90	136	43
E7	10.82	0.53	131	42
E8	10.79	0.68	143	48
E9	10.62	0.76	117	44
E10	10.36	0.69	79	37

† IBM7094 seconds.
⋆ could not be solved in 250 seconds.

reduces the number of subproblems generated by the algorithm in Bilde and Krarup [5] by a factor of 1000. This is especially interesting because Bilde and Krarup use a bound (discovered in 1967) identical to the Erlenkotter bound in their algorithm (see Körkel [25]) and we use the Khachaturov-Minoux combinatorial bound. The CPU time required by the DCA-SPLP to solve these problems were too low to warrant the use of any $\alpha > 0$.

6.4.3 Galvão and Raggi-type Instances

Galvão and Raggi [8] developed a general 0-1 formulation of the SPLP and presented a 3-stage method to solve it. The benchmark instances suggested in this work are unique, in that the fixed costs are assumed to come from a Normal distribution rather than the more commonly used Uniform dis-

tribution. The reader is referred to Galvão and Raggi [8] for a detailed description of the problem data.

As with the data in Bilde and Krarup [5], the exact data for the instances are not known. So we generated 10 instances for each problem size, and used the mean values of the solutions for comparison purposes. In our DCA-SPLP implementation, we used the Khachaturov-Minoux combinatorial bound in the reduction procedure RP and in the DCA-SPLP. The comparative results are given in Table 6. Since the computers used are different, we cannot make any comments on the relative performance of the solution procedures. However, since the average number of subproblems generated by the DCA-SPLP is always less than 10 for each of these instances, we can conclude that these problems are easy for our algorithm. In fact they are too easy for the DCA-SPLP to warrant $\alpha > 0$.

Table 6: Results from Galvão and Raggi-type instances

Problem Size ($m = n$)	DCA				Galvão and Raggi	
	# solved by pre-processing	# of sub-problems†	CPU time†	# of open plants†	CPU time⋆	# of open plants
10	6	2.3	<0.001	4.7	<1	3
20	5	2.4	<0.001	9.0	<1	8
30	7	1.8	0.002	13.6	1	11
50	7	2.6	0.002	20.3	2	20
70	2	3.8	0.004	28.8	6	31
100	3	3.5	0.011	41.1	6	44
150	1	7.8	0.010	64.4	25	74
200	4	2.9	0.158	81.8	63	84

† Average over 10 instances.
⋆ IBM 4331 seconds.

Notice that the average number of opened plants in the optimal solutions to the instances we generated is quite close to the number of opened plants in the optimal solutions reported in Galvão and Raggi [8]. Also notice that the reduction procedure was quite effective — it solved 35 of the 80 instances generated.

6.4.4 Instances from the OR-Library

The OR-Library [3] has a set of instances of the SPLP. These instances were solved in Beasley [2] using an algorithm based on the Lagrangian heuristic for

the SPLP. Here too, we used the Khachaturov-Minoux combinatorial bound in the reduction procedure RP as well as in the DCA-SPLP. We solved the problems to optimality using the DCA. The results of the computations are provided in Table 7. The execution times suggest that the DCA-SPLP is faster than the Lagrangian heuristic described in Beasley [2]. The reduction procedure was also quite effective for these instances, solving 4 of the 16 instances to optimality, and reducing the number of free sites appreciably in the other instances. Once again the use of $\alpha > 0$ cannot be justified, considering the execution times of the DCA.

Table 7: Results from OR-Library instances

Problem name	m	n	DCA: m after preprocessing	DCA: # of subproblems	DCA: CPU time	CPU time (Beasley [2])†	# of open plants
cap71	16	50	⋆	0	<0.01	0.11	11
cap72	16	50	⋆	0	<0.01	0.08	9
cap73	16	50	⋆	0	<0.01	0.11	5
cap74	16	50	⋆	0	<0.01	0.05	4
cap101	25	50	9	6	<0.01	0.18	15
cap102	25	50	13	16	<0.01	0.16	11
cap103	25	50	14	16	<0.01	0.14	8
cap104	25	50	12	7	0.01	0.11	4
cap131	50	50	34	196	0.01	0.31	15
cap132	50	50	27	183	0.02	0.28	11
cap133	50	50	25	71	<0.01	0.29	8
cap134	50	50	19	25	<0.01	0.15	4

⋆ instance solved by preprocessing only.
† Cray-X-MP/28 seconds.

6.4.5 Körkel-type Instances with 65 Sites

Körkel [25] described several relatively large Euclidean SPLP instances ($m = n = 100$, and $m = n = 400$) and used a branch and bound algorithm to solve these problems. The bound used in that work is an improvement on a bound based on the dual of the linear programming relaxation of the SPLP due to Erlenkotter [7] and is extremely effective. In this subsection, we use instances that have the same cost structure as the ones in Körkel [25] but for which $m = n = 65$. Instances of this size were not dealt with in Körkel [25]. We implemented the Khachaturov-Minoux combinatorial bound both for

the reduction procedure RP and the DCA-SPLP.

In Körkel [25], 120 instances of each problem size are described. These can be divided into 28 sets (the first 18 sets contain 5 instances each, and the rest contain 3 instances each). We solved all the 120 instances we generated, and found out that the instances in Sets 1, 2, 3, 4, 10, 11, and 12 are more difficult to solve than others. We therefore used these instances in the experiments in this section. The transportation cost matrix for a Körkel instance of size $n \times n$ is generated by distributing n points in random within a rectangular area of size 700×1300 and calculating the Euclidean distances between them. The fixed cost are computed as in Table 8.

Table 8: Description of the fixed costs for instances in Körkel (1989)

Problem Set	# of instances	Fixed cost for i^{th} instance
Set 1	5	Identical, set at $141 + 6.6i$
Set 2	5	Identical, set at $174 + 6.6i$
Set 3	5	Identical, set at $207 + 6.6i$
Set 4	5	Identical, set at $174 + 66i$
Set10	5	Identical, set at $7170 + 660i$
Set11	5	Identical, set at $7120.5 + 333.3i$
Set12	5	Identical, set at $8787 + 333.3i$

The values of the results that we present for each set is the average of the values obtained for all the instances in that set. Interestingly, the pre-processing rules were found to be totally ineffective for all of these problems. Since the fixed costs are identical for all the sites, the sites are distributed randomly over a region, and the variable cost matrix is symmetric, no site presents a distinct advantage over any other. This prevents our reduction procedure to open or close any site. Table 9 shows the variation in the costs of the solution output by the DCA-SPLP with changes in α, and Table 10 shows the corresponding decrease in execution times.

The effect of varying the acceptable accuracy α on the cost of the solutions output by the DCA-SPLP is also presented graphically in Figure 11. We define the *achieved accuracy* β as

$$\beta = \frac{\text{cost of the DCA-SPLP output} - \text{cost of an optimal solution}}{\text{cost of optimal solution}}$$

and the *relative time* τ as

Table 9: Costs of solutions output by the DCA-SPLP on Körkel-type instances with 65 sites

Problem Set	Optimal	Acceptable accuracy*				
		1%	2%	3%	5%	10%
Set 1	6370.0	6404.8	6450.6	6480.6	6569.2	6781.0
Set 2	6920.6	6952.2	6971.4	7028.4	7123.8	7320.2
Set 3	7707.4	7738.0	7770.2	7797.6	7854.6	8053.8
Set 4	9601.2	9642.4	9680.2	9698.4	9786.6	9932.0
Set10	146691.2	146896.6	146909.6	147543.6	148062.0	151542.2
Set11	168598.4	168858.2	169655.0	170341.6	170597.0	173913.8
Set12	186386.3	186729.7	187112.0	188002.7	188854.2	192528.7

⋆ As a percentage of the optimal cost.

Table 10: Execution times for the DCA-SPLP on Körkel-type instances with 65 sites

Problem Set	Optimal	Acceptable accuracy*				
		1%	2%	3%	5%	10%
Set 1	119.078	90.948	70.758	55.494	43.200	20.426
Set 2	290.388	225.108	172.422	145.828	96.240	36.966
Set 3	458.370	339.420	259.022	203.036	150.216	50.378
Set 4	158.386	129.694	109.754	89.666	65.548	30.058
Set10	428.598	370.120	319.804	283.832	230.078	142.090
Set11	542.530	476.350	418.628	408.594	290.338	160.744
Set12	479.092	416.472	370.832	326.572	261.835	149.038

⋆ As a percentage of the optimal cost.

$$\tau = \frac{\text{execution time for the DCA-SPLP for acceptable accuracy } \alpha}{\text{execution time for the DCA-SPLP to compute an optimal solution}}$$

Note that the achieved accuracy β varies almost linearly with α, with a slope close to 0.5. Also note that the relative time τ of the DCA-SPLP reduces with increasing α. The reduction is slightly better than linear, with an average slope of -8.

6.4.6 Körkel-type Instances with 100 Sites

We solved the benchmark instances in Körkel [25] with $m = n = 100$ to optimality and observed that the instances in Sets 10, 11, and 12 required

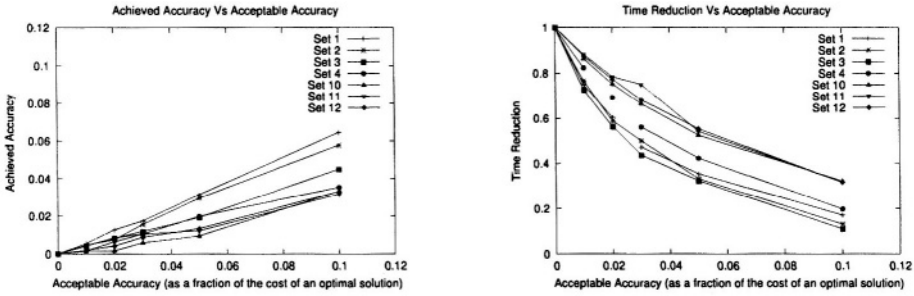


Figure 11: Performance of the DCA-SPLP for Körkel-type instances with 65 sites

relatively longer execution times. So we restricted further computations to instances in those sets. The fixed and transportation costs for these problems are computed in the procedure described in Subsection 6.4.5. Tables 11 and 12 show the results obtained by running the DCA-SPLP on these problem instances. In our DCA-SPLP implementation for solving these instances, we used the Erlenkotter bound in both the reduction procedure RP and the DCA-SPLP.

Table 11: Costs of solutions output by the DCA-SPLP on Körkel-type instances with 100 sites

Problem	Optimal	Acceptable accuracy*				
Set		1%	2%	3%	5%	10%
Set10	190782.0	191550.8	192755.4	192080.6	195983.2	203934.2
Set11	219583.4	220438.8	222393.6	221947.2	228467.2	235963.4
Set12	240402.4	241609.6	243336.8	244209.4	247417.6	259168.6

$\star$ As a percentage of the optimal cost.

Figure 12 illustrates the effect of varying the acceptable accuracy α on the cost of the solutions output by the DCA-SPLP for the instances mentioned above. The nature of the graphs is similar to those in Figure 11. However, in several of the instances we noticed that β reduced when α is increased, and in some other instances τ increased when α was increased.

Table 12: Execution times for the DCA-SPLP on Körkel-type instances with 100 sites

Problem Set	Optimal	Acceptable accuracy*				
		1%	2%	3%	5%	10%
Set10	133.746	91.774	65.99	65.908	44.2	32.074
Set11	81.564	55.356	39.554	38.348	33.628	17.598
Set12	111.272	85.858	65.608	55.928	61.758	33.014

⋆ As a percentage of the optimal cost.

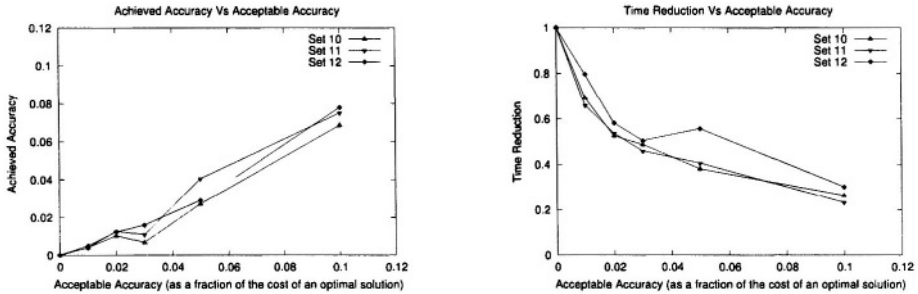


Figure 12: Performance of the DCA-SPLP for Körkel-type instances with 100 sites

References

[1] E. Balas and P. Toth, Branch and Bound Methods, Chapter 10 in Lawler *et al.* [26].

[2] J.E. Beasley, Lagrangian Heuristics for Location Problems, *European Journal of Operational Research* Vol.65 (1993) pp. 383-399.

[3] J.E. Beasley, OR-Library, `http://mscmga.ms.ic.ac.uk/info.html`

[4] V.L. Beresnev, On a Problem of Mathematical Standardization Theory, *Upravliajemyje Sistemy* Vol.11 (1973) pp. 43-54 (in Russian).

[5] O. Bilde and J. Krarup, Sharp Lower Bounds and Efficient Algorithms for the Simple Plant Location Problem, *Annals of Discrete Mathematics* Vol.1 (1977) pp. 79-97.

[6] G. Cornuejols, G.L. Nemhauser, and L.A. Wolsey, The Uncapacitated Facility Location Problem, in P.B. Mirchandani and R.L. Francis (eds.)

Discrete Location Theory, (New York:Wiley-Interscience, 1990) pp. 119-171.

[7] D. Erlenkotter, A Dual-Based Procedure for Uncapacitated Facility Location, *Operations Research* Vol.26 (1978) pp. 992-1009.

[8] R.D. Galvão and L.A. Raggi, A Method for Solving to Optimality Uncapacitated Location Problems, *Annals of Operations Research* Vol.18 (1989) pp.225-244.

[9] B. Goldengorin, Methods of Solving Multidimensional Unification Problems, *Upravljaemye Sistemy* Vol.16 (1977) pp. 63-72.

[10] B. Goldengorin, A Correcting Algorithm for Solving Some Discrete Optimization Problems, *Soviet Mathamетical Doklady* Vol.27 (1983) pp. 620-623.

[11] B. Goldengorin, A Correcting Algorithm for Solving Allocation Type Problems, *Automated Remote Control* Vol.45 (1984) pp. 590-598.

[12] B. Goldengorin, Correcting Algorithms for Solving Multivariate Unification Problems, *Soviet Journal of Computer Systems Science* Vol.1 (1985) pp. 99-103.

[13] B. Goldengorin, On the Exact Solution of Problems of Unification by Correcting Algorithms, *Doklady Akademii, Nauk, SSSR* Vol.294 (1987) pp. 803-807.

[14] B. Goldengorin, G. Sierksma, G.A. Tijssen, and M. Tso, The Data-Correcting Algorithm for Minimization of Supermodular Functions. *Management Science* Vol.45 (1999) pp. 1539-1551.

[15] B. Goldengorin, D. Ghosh, and G. Sierksma, *Equivalent Instances of the Simple Plant Location Problem,* (SOM Research Report-00A54, University of Groningen, The Netherlands 2000).

[16] B. Goldengorin, G.A. Tijssen, D. Ghosh, G. Sierksma, Solving the Simple Plant Location Problem Using a Data Correcting Approach, *Journal of Global Optimization.* Vol.25 (2003) pp. 377-406.

[17] B. Goldengorin, *Data Correcting Algorithms in Combinatorial Optimization,* (Ph.D. Thesis, SOM Research Institute, University of Groningen, Groningen, The Netherlands, 2002).

[18] B. Goldengorin and D. Ghosh, A Multilevel Search Algorithm for the Maximization of Submodular Functions, (to appear in *Journal of Global Optimization).*

[19] P.C. Gilmore, E.L. Lawler, and D.B. Shmoys, Well-Solved Special Cases, Chapter 4 in Lawler *et al.* [26].

[20] G. Gutin and A.P. Punnen (eds.) *The Traveling Salesman Problem and its Variations,* (Kluwer Academic Publishers, The Netherlands, 2002).

[21] P.L. Hammer, Plant Location — A Pseudo-Boolean Approach. *Israel Journal of Technology* Vol.6 (1968) pp. 330-332.

[22] V.R. Khachaturov, *Some Problems of the Consecutive Calculation Method and its Applications to Location Problems,* (Ph.D. Thesis, Central Economics and Mathematics Institute, Russian Academy of Sciences, Moscow, 1968), (in Russian).

[23] V.R. Khachaturov, *Mathematical Methods of Regional Programming,* (Moscow, Nauka, 1989), (in Russian).

[24] B.M. Khumawala, An Efficient Branch and Bound Algorithm for the Warehouse Location Problem, *Management Science* Vol.18 (1975) pp. B718-B731.

[25] M. Körkel, On the Exact Solution of Large-Scale Simple Plant Location Problems. *European Journal of Operational Research* Vol.39 (1989) pp. 157-173.

[26] E.L. Lawler, J.K. Lenstra, A.H.G. Rinooy Kan, and D.B. Shmoys, (eds.) *The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization,* (Wiley-Interscience, 1985).

[27] H. Lee, G.L. Nemhauser, and Y. Wang, Maximizing a Submodular Function by Integer Programming: Polyhedral Results for the Quadratic Case, *European Journal of Operational Research* Vol.94 (1996) pp. 154-166.

[28] L. Lovasz, Submodular Functions and Convexity, in A. Bachem, M. Grötschel, B. Korte (eds.) *Mathematical Programming: The State of the Art,* (Springer-Verlag, Berlin, 1983) pp. 235-257.

[29] M. Minoux, Accelerated Greedy Algorithms for Maximizing Submodular Set Functions, in J. Stoer (ed.) *Actes Congres IFIP,* (Springer, Berlin, 1977) pp. 234-243.

[30] G. Reinelt, TSPLIB 95, `http://www.iwr.uni-heidelberg.de/iwr/comopt/soft/TSPLIB95/TSPLIB.html`, 1995.

HANDBOOK OF COMBINATORIAL OPTIMIZATION (Vol 5)
D.-Z. Du and P.M. Pardalos (Eds.) pp. 55-81

The Steiner Ratio of Banach-Minkowski spaces - A Survey

Dietmar Cieslik
Institute of Mathematics and Computer Science
University of Greifswald, Germany
E-mail: cieslik@mail.uni-greifswald.de

Contents

1 Introduction

Starting with the famous book "What is Mathematics" by Courant and Robbins the following problem has been popularized under the name of Steiner:

> For a given finite set of points in a metric space find a network which connects all points of the set with minimal length.

Such a network must be a tree, which is called a Steiner Minimal Tree (SMT). It may contain vertices other than the points which are to be connected. Such points are called Steiner points.[1]
Given a set of points, it is a priori unclear how many Steiner points one has to add in order to construct an SMT, but one can prove that we need not more than $n-2$, whereby n is the number of given points.
A classical survey of Steiner's Problem in the Euclidean plane was presented by Gilbert and Pollak in 1968 [28] and christened "Steiner Minimal Tree" for the shortest interconnecting network and " Steiner points" for the additional vertices.
Without loss of generality, the following is true for any SMT for a finite set N of points in the Euclidean plane:

1. The degree of each vertex is at most three;
2. The degree of each Steiner point equals three; and two edges which are incident to a Steiner point meet at as angle of 120°;
3. There are at most $|N|-2$ Steiner points.

Moreover, in the paper by Gilbert and Pollak, there are a lot of interesting conjectures, stimulating the research in this field in the next years.

It is well-known that solutions of Steiner's problem depend essentially on the way in which the distances in space are determined. In recent years it turned out that in engineering design it is interesting to consider Steiner's Problem and similar problems in several two-dimensional Banach spaces and some specific higher-dimesnional cases. Over the years Steiner's Problem

[1]The history of Steiner's Problem started with P.Fermat [22] early in the 17th century and C.F.Gauß [27] in 1836. At first perhaps with the famous book *What is Mathematics* by R.Courant and H.Robbins in 1941, this problem became popularized under the name of Steiner.

has taken on an increasingly important role, it is one of the most famous combinatorial-geometrical problems. Consequently, in the last three decades the investigations and, naturally, the publications about Steiner's Problem have increased rapidly. Surveys are given by Cieslik [10], Hwang, Richards, Winter [32] and Ivanov, Tuzhilin [33]. However, all investigations showed the great complexity of the problem, as well in the sense of structural as in the sense of computational complexity. In other terms:

Observation I.

In general, methods to find an SMT are hard in the sense of computational complexity or still unknown. In any case we need a subtile description of the geometry of the space.

On the other hand, a Minimum Spanning Tree[2] (MST) can be found easily by simple and general applicable methods.

Observation II.

It is easy to find an MST by an algorithm which is simple to realize and running fast in all metric space.

Hence, it is of interest to know what the error is if we construct an MST instead of an SMT. In this sense, we define the Steiner ratio for a space to be the infimum over all finite sets of points of the length of an SMT divided by the length of an MST:

$$m := \inf \left\{ \frac{L(\text{SMT for } N)}{L(\text{MST for } N)} : N \text{ a finite set in the space} \right\}.$$

This quantity is a parameter of the considered space and describes the performance ratio of the the approximation for Steiner's Problem by a Minimum Spanning Tree.

This present paper concentrates on investigating the Steiner ratio. The goal is to determine or at least to estimate the Steiner ratio for many different spaces.

2 Banach-Minkowski Spaces

Obviously, Steiner's Problem depends essentially on the way how the distances in the plane are determined. In the present paper we consider finite-dimensional Banach spaces. These are defined in the following way: A_d

[2]This is a shortest tree interconnecting a finite set of points without Steiner points.

denotes the d-dimensional affine space with origin o. That means; A_d is a set of points and these points act over a d-dimensional linear space. We identify each point with its vector with respect to the origin. In other words, elements of A_d will be called either points when considerations have a geometrical character, or vectors when algebraic operations are applied. In this sense the zero-element o of the linear space is the origin of the affine space. The dimension of an affine space is given by the dimension of its linear space. A two-dimensional affine space is called a plane. A non-empty subset of a affine space which is itself an affine space is called an affine subspace.

The idea of normed spaces is based on the assumption that to each vector of a space can be assigned its "length" or norm, which satisfies some "natural" conditions.
A convex and compact body B of the d-dimensional affine space A_d centered in the origin o is called a unit ball, and induces a norm $||.|| = ||.||_B$ in the corresponding linear space by the so-called Minkowski functional:

$||v||_B = \inf\{t > 0 : v \in tB\}$ for any v in $A_d \setminus \{o\}$, and

$||o||_B = 0.$

On the other hand, let $||.||$ be a norm in A_d, which means:
$||.|| : A_d \to \mathbb{R}$ is a real-valued function satisfying

(i) positivity: $||v|| \geq 0$ for any v in A_d;

(ii) identity: $||v|| = 0$ if and only if $v = o$;

(iii) homogenity: $||tv|| = |t| \cdot ||v||$ for any v in A_d and any real t;

and

(iv) triangle inequality: $||v + v'|| \leq ||v|| + ||v'||$ for any v, v' in A_d.

Then $B = \{v \in A_d : ||v|| \leq 1\}$ is a unit ball in the above sense. It is not hard to see that the correspondences between unit balls B and norms $||.||$ are unique. That means that a norm is completely determined by its unit ball and vice versa. Consequently, a Banach-Minkowski space is uniquely defined by an affine space A_d and a unit ball B. This Banach-Minkowski space is abbreviated as $M_d(B)$. In each case we also have the induced norm $||.||_B$ in the space.

A Banach-Minkowski space $M_d(B)$ is a complete metric linear space if we define the metric by

$$\rho(v, v') = ||v - v'||_B. \tag{1}$$

Usually, a (finitely- or infinitely-dimensional) linear space which is complete with regard to its given norm is called a Banach space. Essentially, every Banach-Minkowski space is a finite-dimensional Banach space and vice versa.

All norms in a d-dimensional affine space induce the same topology, the well-known topology with coordinate-wise convergence.[3] In other words: On a finite dimensional linear space all norms are topologically equivalent, i.e. there are positive constants c_1 and c_2 such that

$$c_1 \cdot ||.|| \leq |||.||| \leq c_2 \cdot ||.|| \tag{2}$$

for the two norms $||.||$ and $|||.|||$.
Conversely, there is exactly one topology that generates a finite-dimensional linear space to a metric linear space satisfying the separating property by Hausdorff.

Let $M_d(B)$ and $M_d(B')$ be Banach-Minkowski spaces.
$M_d(B)$ is said to be isometric to $M_d(B')$ if there is a mapping $\Phi : A_d \to A_d$ (called an isometry) which preserves the distances:

$$||\Phi(v) - \Phi(v')||_{B'} = ||v - v'||_B \tag{3}$$

for all v, v' in A_d.
A well-known fact given by Mazur and Ulam says that each isometry mapping a Banach-Minkowski space onto another, such that it maps o on o, is a linear operator. Hence, $M_d(B)$ is isometric to $M_d(B')$ if and only if there is an affine map $\Phi : A_d \to A_d$ with $\Phi B = B'$. Also the affine map Φ is the isometry itself.

Steiner's Problem looks for a shortest network and in particular for a shortest length of a curve $\mathcal{C}$ joining two points. For our purpose, we regard a geodesic curve as any curve of shortest length.
If we parametrize the curve $\mathcal{C}$ by a differentiable map $\gamma : [0, 1] \to \mathbb{R}^d$ we define

$$\text{length of } \mathcal{C} = \int_0^1 ||\dot{\gamma}|| \, dt. \tag{4}$$

[3] This is the topology derived from the Euclidean metric.

It is not hard to see that among all differentiable curves $\mathcal{C}$ from the point v to the point v' the segment

$$\underline{vv'} = \{tv + (1-t)v' : 0 \leq t \leq 1\} \tag{5}$$

minimizes the length of $\mathcal{C}$.

A unit ball B in an affine space is called strictly convex if one of the following pairwise equivalent properties is fulfilled:

- For any two different points v and v' on the boundary of B, each point $w = tv + (1-t)v', 0 < t < 1$, lies in int$B$.
- No segment is a subset of bd$B$.
- $||v+v'||_B = ||v||_B + ||v'||_B$ for two vectors v and v' implies that v and v' are linearly dependent.

One property more we have in

Lemma 2.1 *All segments in a Banach-Minkowski space are shortest curves (in the sense of inner geometry). They are the unique shortest curves if and only if the unit ball is strictly convex.*

Hence, we can define the metric in a Banach-Minkowski space $M_d(B)$ by

$$\rho(v, v') = \frac{2 \cdot ||v - v'||_{B^e}}{||w - w'||_{B^e}}, \tag{6}$$

where $\underline{ww'}$ is the Euclidean diameter of B parallel to the line through v and v' and $||.||_{B^e}$ denotes the Euclidean norm.

A function F defined on a convex subset of the affine space is called a convex function if for any two points v and v' and each real number t with $0 \leq t \leq 1$, the following is true

$$F(tv + (1-t)v') \leq tF(v) + (1-t)F(v'). \tag{7}$$

A function F is called a strictly convex function, if the following is true for any two different points v and v' and each real number t with $0 < t < 1$:

$$F(tv + (1-t)v') < tF(v) + (1-t)F(v'). \tag{8}$$

A norm is a convex function. Moreover, the unit ball of a strictly convex norm is a strictly convex set.

Lemma 2.2 *For a norm $||.||$ in a finite-dimensional affine space the following holds:*

(a) *A norm $||.||$ in a finite-dimensional affine space is a convex and thus a continuous function.*

(b) *A norm $||.||$ is a strictly convex function if and only if its unit ball $B = \{v \in A_d : ||v|| \leq 1\}$ is a strictly convex set.*

The dual norm $||\cdot||_{DB}$ of the norm $||\cdot||_B$ is defined as

$$||v||_{DB} = \max_{w \neq o} \frac{(v, w)}{||w||_B} \tag{9}$$

and has the unit ball DB, called the dual unit ball, which can be described as

$$DB = \{w : (v, w) \leq 1 \text{ for all } v \in B\}.$$

(Here, (., .) denotes the standard inner product.) Immediately, we have that for any two vectors v and w the inequality

$$(v, w) \leq ||v||_{DB} \cdot ||w||_B; \tag{10}$$

is true and it is not hard to see that $B \subseteq B'$ holds if and only if $DB' \subseteq DB$. An example of non-Euclidean norms dual to each other is

$$||(t_1, \ldots, t_d)||_B = \max\{|t_1|, \ldots, |t_d|\} \tag{11}$$

and

$$||(t_1, \ldots, t_d)||_{DB} = |t_1| + \ldots + |t_d|, \tag{12}$$

whereby B is a hypercube and DB is a cross-polytope.

Particularly, we consider finite-dimensional spaces with **p-norm,** defined in the following way: Let A_d be the **d-dimensional** affine space. For the point $v = (x_1, \ldots, x_d)$ we define the norm by

$$||v|| = \left(\sum_{i=1}^{d} |x_i|^p \right)^{1/p}$$

where $1 \leq p < \infty$ is a real number. If p runs to infinity we get the so-called Maximum norm

$$||v||_\infty = \max\{|x_i| : 0 \leq i \leq d\}$$

In each case we obtain a Banach-Minkowski space written by $\mathcal{L}_p^d$.
$\mathcal{L}_1^d$ and $\mathcal{L}_\infty^d$ normed by a cross-polytope and a cube, respectively. For $1 < p < \infty$ the space $\mathcal{L}_p^d$ is strictly convex. The spaces $\mathcal{L}_p^d$ and $\mathcal{L}_q^d$ with $1/p + 1/q = 1$ are dual.

3 Steiner's Problem and the Steiner ratio

A (finite) graph $G = (V, E)$ with the set V of vertices and the set E of edges is embedded in the Banach-Minkowski space $M_d(B)$ in the sense that

- V is a finite set of points in the space;
- Each edge $\underline{vv'} \in E$ is a segment $\{tv + (1-t)v' : 0 \leq t \leq 1\}$, $v, v' \in V$; and
- The length of G is defined by

$$L(G) = L_B(G) = \sum_{\underline{vv'} \in E} ||v - v'||_B.$$

Now, **Steiner's Problem of Minimal Trees** is the following:

Given: A finite set N of points in the Banach-Minkowski space $M_d(B)$.

Find: A connected graph $G = (V, E)$ embedded in the space such that
- $N \subseteq V$ and
- $L_B(G)$ is minimal as possible.

A solution of Steiner's Problem is called a Steiner Minimal Tree (SMT) for N in the space $M_d(B)$.[4] The vertices in the set $V \setminus N$ are called Steiner points. We may assume that for any SMT $T = (V, E)$ for N the following holds: The degree of each Steiner point is at least three and

$$|V \setminus N| \leq |N| - 2. \tag{13}$$

If we don't allow Steiner points, that is if we connect certain pairs of given points only, then we refer to a Minimum Spanning Tree (MST). Starting with Boruvka in 1926 and Kruskal in 1956, Minimum Spanning Trees have a well-documented history [29] and effective constructions [3].

A minimum spanning tree in a graph $G = (N, E)$ with a positive length-function $f : E \to \mathbb{R}$, can be found with the help of Kruskal's [34] well-known method:

1. Start with the forest $T = (N, \emptyset)$;

[4] That for any finite set of points there an SMT always exists is not obvious. Particularly, it is proved in [10].

2. Sequentially choose the shortest edge that does not form a circle with already chosen edges;

3. Stop when all vertices are connected, that is when $|N| - 1$ edges have been chosen.

Then an MST for a finite set N of points in $M_d(B)$ can be found obtaining the graph $G = (N, \binom{N}{2})$ with the length-function $f(\underline{vv'}) = ||v - v'||_B$.

Let N be a finite set of points in $M_d(B)$. We saw that it is easy to find an MST for N; this is valid in the sense of the combinatorial structure as well as in the sense of computational complexity. On the other hand, methods to find an SMT for N are still unknown or at least hard in the sense of computational complexity. More exactly:

space	complexity	source
Euclidean plane	$\mathcal{NP}$-hard	[24]
Rectilinear plane $\mathcal{L}_1^2$	$\mathcal{NP}$-hard	[25]
$\mathcal{L}_p$-planes	algorithm needs exponential time	[13]
Banach plane	algorithm needs exponential time	[9]

For higher-dimensional spaces the problems are not easier than in the planes. For a complete discussion of these difficulties see [10] and [32].
Moreover, to solve Steiner's Problem we need facts about the geometry of the space. On the other hand, for an MST we only use the mutual distances between the points.

Consequently, we are interested in the value

$$m_d(B) := \inf \left\{ \frac{L_B(\text{SMT for } N)}{L_B(\text{MST for } N)} : N \subseteq M_d(B) \text{ is a finite set} \right\}, \qquad (14)$$

which is called the Steiner ratio of the space $M_d(B)$.
The quantity $m_d(B) \cdot L(\text{MST for } N)$ would be a convenient lower bound for the length of an SMT for N in the space $M_d(B)$; that means, roughly speaking, $m_d(B)$ says how much the total length of an MST can be decreased by allowing Steiner points.

For the space $\mathcal{L}_p^d$ the Steiner ratio will be briefly written by $m(d, p)$.

4 Basic properties for the Steiner ratio

It is obvious that $0 < m_d(B) \leq 1$ for the Steiner ratio $m_d(B)$ of each Banach-Minkowski space $M_d(B)$. Of course, if $d = 1$ then the MST and the SMT are identical, and it is $m_1(B) = 1$. Moreover,

Theorem 4.1 *(E.F.Moore in [28]) For the Steiner ratio of every Banach-Minkowski Space*

$$m_d(B) \geq \frac{1}{2} = 0.5$$

holds.

In the d-dimensional affine space A_d, the unit ball $B(1)$ is the convex hull of

$$N = \{\pm(0, ..., 0, 1, 0, ..., 0) : \text{the i'th component is equal to } 1, i = 1, \ldots, d\}. \tag{15}$$

The set N contains $2d$ points. The rectilinear distance of any two different points in N equals 2. Hence, an MST for N has the length $2(2d-1)$. Conversely, an SMT[5] for N with the Steiner point o = (0,...,0) has the length $2d$. This implies the first fact of

Theorem 4.2 *For the Steiner ratio of spaces with rectilinear norm the following are true.*

(a) *In the case of* d *dimensions we have*

$$m(d, 1) \leq \frac{d}{2d-1}. \tag{16}$$

(b) *(Hwang [31]) In two dimensions in (16) equality holds:*

$$m(2, 1) = \frac{2}{3}. \tag{17}$$

Graham and Hwang [30] conjectured that in (16) always equality holds, which is true in the planar case, (17), but the methods by Hwang do not seem to be applicable to proving the conjecture in the higher dimensional case.
Comparing the last two theorems, we observe the following:

[5] that this tree is indeed an SMT is not simple to see!

Corollary 4.3 *The lower bound* 1/2 *is the best possible for the Steiner ratio over the class of all Banach-Minkowski spaces.*

Let $M_d(B)$ be a d-dimensional Banach-Minkowski space, and let $A_{d'}$ be a d'-dimensional affine subspace $(d' \leq d)$ with $o \in A_{d'}$. Clearly, the intersection $B \cap A_{d'}$ can be considered as the unit ball of the space $A_{d'}$. This means that $M_{d'}(B \cap A_{d'})$ is a (Banach-Minkowski) subspace of $M_d(B)$. Let v and v' be two different points in $A_{d'}$. Then the line through v and v' lies completely in $A_{d'}$, and in view of 2.1 and (6) we see that the distance between the points v and v' is preserved:

$$||v - v'||_B = ||v - v'||_{B \cap A_{d'}}. \tag{18}$$

Then we have

Theorem 4.4 *Let* $M_{d'}(B')$ *be a (Banach-Minkowski) subspace of* $M_d(B)$. *Then* $m_{d'}(B') \geq m_d(B)$.

An interesting problem, but which seems as very difficult, is to determine the range of the Steiner ratio for d-dimensional Banach-Minkowski spaces, depending on the value d. More exactly, determine the best possible reals c_d and C_d such that

$$c_d \leq m_d(B) \leq C_d, \tag{19}$$

for all unit balls B of A_d.
The quantity C_d is defined as the upper bound of all numbers $m_d(B)$ ranging over all unit balls B of A_d:

$$C_d = \sup\{m_d(B) : B \in \underline{B}_d\}. \tag{20}$$

The sequence $\{C_d\}_{d=1,2,\ldots}$, starting with $C_1 = 1$ is a decreasing and bounded, consequently a convergent one. Is it true that $C_d = m(d, 2)$ for $d = 2, 3, \ldots$? On the other hand,

$$c_d = \inf\{m_d(B) : B \in \underline{B}_d\}. \tag{21}$$

is of interest. Does the equality $c_d = m(d, 1)$ for $d = 2, 3, \ldots$ hold?

5 The Steiner ratio of the Euclidean plane

Original, Steiner's Problem was considered in the Euclidean plane. Even here, we find that the complexity of computing an SMT is $\mathcal{NP}$-hard. The

complexity of computing SMT's in higher-dimensional spaces is demonstrably even more difficult, since here is no inherent combinatorial structure present in the problem, compare [40].

A long-standing conjecture, given by Gilbert and Pollak in 1968, said that $m(2,2) = \sqrt{3}/2$. Many persons have tried to show this; successively establishing that $\sqrt{3}/2$ does indeed hold for sets with a small number of points: Pollak [36] and Du, Yao, Hwang [15] have shown that the conjecture is valid for sets N consisting of $n = 4$ points; Du, Hwang, Yao [17] stated this result to the case $n = 5$, and Rubinstein, Thomas [37] have done the same for the case $n = 6$.
On the other hand, many attempts have been made to estimate the Steiner ratio for the Euclidean plane from below:

$m(2,2) \geq 1/\sqrt{3}$	$= 0.57735\ldots$	Graham, Hwang [30]
$m(2,2) \geq \sqrt{2\sqrt{3}+2-(7+2\sqrt{3})}$	$= 0.74309\ldots$	Chung, Hwang [5]
$m(2,2) \geq 4/5$	$= 0.8$	Du, Hwang [16]
$m(2,2)$	$\geq 0.82416\ldots$	Chung, Graham [6]

Finally, Du and Hwang created a lot of new methods and succeeded in proving the Gilbert-Pollak conjecture completely:

Theorem 5.1 *(Du, Hwang [18], [19]) The Steiner Ratio of the Euclidean plane equals*

$$m(2,2) = \frac{\sqrt{3}}{2} = 0.86602\ldots.$$

Now, we are interested in the sets of points which achieve the Steiner ratio. Clearly, when N contains the nodes of an equilateral triangle we have

$$\frac{L(\text{SMT for } N)}{L(\text{MST for } N)} = \frac{\sqrt{3}}{2} = m(2,2). \tag{22}$$

In a first view, it seems that no other finite set of points has this property. Probably, this is true, but Du and Smith [21] had an surprising idea: Let's look at some special set configurations created by joining equilateral triangles at a common side. This is actually called a 2-sausage, more formally:

1. Start with a unit circle;
2. Successively add unit circles so that the n'th circle you add is always touching the $\min\{2, n-1\}$ most recently added circles.

This procedure uniquely[6] defines an infinite sequence of interior-disjoint

[6] up to congruence

numbered circles. The centers of these circles form a discrete point set, which is called the (infinity) 2-sausage. The first n points of the 2-sausage will be called the "n-point 2-sausage" $N(n,2)$ or "flat-sausage" for simplicity.
What is remarkable about the sausages? At first their regularity and then the fact that the ratio between the length of an SMT and the length of an MST for $N(2n,2)$ decreases with increasing number n. Moreover,

Theorem 5.2 *(Du, Smith [21])*

$$\lim_{n\to\infty} \frac{L(SMT\ for\ N(2n,2))}{L(MST\ for\ N(2n,n))} = \frac{\sqrt{3}}{2} = m(2,2).$$

6 The Steiner ratio of $\mathcal{L}_p$-planes

If we have an analytic formula, which decribes the norm, we have also the possibility to estimate the Steiner ratio with direct calculations. In this section we will determine upper and lower bounds for the Steiner ratio $m(2,p)$ of two-dimensional $\mathcal{L}_p$-spaces.

Du and Liu determined an upper bound for the Steiner ratio using direct calculations of the ratio between the length of SMT's and of MST's for sets with three elements:

Theorem 6.1 *(Du, Liu [35]) The following is true for the Steiner ratio of the $\mathcal{L}_p$-planes $M_2(B(p))$:*

$$m(2,p) \le \frac{(2^p-1)^{1/p} + (2^q-1)^{1/q}}{4}, \tag{23}$$

where q is the conjugated number to p; that means $\frac{1}{p} + \frac{1}{q} = 1$.

Corollary 6.2 *For $1 < p < \infty$ it holds*

$$m(2,p) \le m(2,2) = \frac{\sqrt{3}}{2} = 0.866025\ldots.$$

Furthermore, equality holds if and only if $p = 2$.

The proof of 6.1 uses a specific triangle. Now, we will use a triangle which has a side parallel to the line $\{(x,x) : x \in \mathbb{R}\}$. Let $1 < p < \infty$

and $u = (0,1)$, $v = (1,0)$ and $w = (x_p, x_p)$. We want that the triangle spanned by u,v and w is equilateral and, additionally, x_p lies between 1 and 2. Considering this triangle Albrecht [1] gives the following new upper bounds for $m(2,p)$ for specific values p:

p	q	Theorem 6.1	new with p	new with q
1.1	11	0.782399...	0.775933...	0.775933...
1.2	6	0.809264...	0.797975...	0.797975...
1.3	4.3...	0.829043...	0.816708...	0.816708...
1.4	3.5	0.842759...	0.832320...	0.832320...
1.5	3	0.852049...	0.844625...	0.844625...
1.6	2.6...	0.858207...	0.853640...	0.853640...
1.7	2.428571...	0.862145...	0.859755...	0.859755...
1.8	2.25	0.864491...	0.863518...	0.863518...
1.9	2.1...	0.865681...	0.865460...	0.865460...
2.0	2	0.866025...	0.866025...	0.866025...

It is not hard to see, that considering sets with three points only creates estimates for the Steiner ratio which are at least 3/4, compare [28]. Using sets with four points gives

Theorem 6.3 *(Albrecht [1]) The Steiner ratio of* $\mathcal{L}_p^2$ *is essentially less than* $\frac{3}{4}$ *if* $p \leq 1.2$ *or if* $p \geq 6$.

How can we find a lower estimate for the Steiner ratio of the planes? Here, we use two facts:

1. The values for $\mathcal{L}_1^2$ and $\mathcal{L}_2^2$ are exactly known: $m(2,1) = 2/3$ and $m(2,2) = \sqrt{3}/2$.

2. We introduce a distance function between classes of Banach-Minkowski spaces in the following way: Let $\underline{B}_d$ denote the class of all unit balls in A_d, and let $[\underline{B}_d]$ be the space of classes of isometrics for $\underline{B}_d$. Then the Banach-Mazur distance abst is a metric on $[\underline{B}_d]$ defined as

$$abst([B],[B']) = \ln \inf\{h \geq 1 : \textit{there is an isometry } A \textit{ such that } B \subseteq AB \subseteq hB\}$$

for $[B]$, $[B']$ in $[\underline{B}_d]$. The space ($[\underline{B}_2]$, abst) is a compact metric space, with diameter $= \ln \frac{3}{2}$, compare [42].

With these facts in mind we find

Theorem 6.4 *(C. [8], [10]) The following are true for the Steiner ratio of the $\mathcal{L}_p$-planes:*

$$m(2,p) \geq \begin{cases} \frac{2^{1/p}}{3} & \text{if } 1 \leq p \leq \frac{ln\ 16}{ln\ 13.5} \\ \frac{\sqrt{6}}{2} \cdot 2^{-1/p} & \text{if } \frac{ln\ 16}{ln\ 13.5} \leq p \leq 2. \end{cases}$$

We can find bounds for $m(2,p)$, $p \geq 2$, if we replace p by $p/(p-1)$ on the right side.

7 The Steiner ratio of Banach-Minkowski planes

For many specific planes the Steiner ratio is known or well estimated. In the section before we find the values for $\mathcal{L}_p$-planes, including the exact value for the Euclidean plane and the plane with rectilinear norm. Additionally, it is an interesting question to consider planes which are normed by a regular polygon with an even number of corners.
We defined the so-called λ−geometry $M_2(B^{(\lambda)})$ in the following way: The unit ball $B^{(\lambda)}$ is a regular 2λ-gon with the x-axis being a diagonal direction. Particularly, it holds

$$M_2(B^{(2)}) = \mathcal{L}_1^2 \tag{24}$$
$$M_2(B^{(\infty)}) = \mathcal{L}_2^2. \tag{25}$$

We have the

Theorem 7.1 *(Sarrafzadeh, Wong [38]) For the Steiner ratio of the planes with λ-geometry it holds that*

$$m_2(B^{(\lambda)}) \geq \frac{\sqrt{3}}{2} \cos \frac{\pi}{2\lambda}. \tag{26}$$

It follows from the last theorem that $m_2(B^{(3)}) \geq 3/4$. $B^{(3)}$ is an affinely regular hexagon. In view of an isometry, we may assume that

$$B^{(3)} = \text{conv}\{(1,1),(-1,-1),(1,0),(-1,0),(0,1),(0,-1)\}, \tag{27}$$

which implies that

$$||(x_1,x_2)||_{B^{(3)}} = \max\{|x_1|,|x_2|,|x_1-x_2|\}. \tag{28}$$

Then it is easy to see that the set $N = \{(1,1), (-1,0), (0,-1)\}$ has an MST of the length 4 and an SMT of the length at most 3. Hence, the Steiner Ratio is not greater than 3/4, and we have

Theorem 7.2 *(Du et.al. [20]) Let B be an affinely regular hexagon in the plane. Then*

$$m_2(B) = \frac{3}{4} = 0.75.$$

What is known about the Steiner ratio of two-dimensional Banach spaces in general? A sharp lower bound for the Steiner ratio of any Banach-Minkowski plane we have in

Theorem 7.3 *(Gao, Du, Graham [23]) For the Steiner ratio of Banach-Minkowski planes the following is true:*

$$m_2(B) \geq \frac{2}{3}.$$

If there is a natural number n such that the bound 2/3 is adopted by a set of n points, then $n = 4$, and B is a parallelogram.

This bound is the best possible one, since the plane with rectilinear norm has Steiner ratio 2/3. Such a nice result for an upper bound is unknown, but we have

Theorem 7.4 *(Du et.al. [20]) For any unit ball B in the plane the following is true:*

$$m_2(B) \leq \frac{\sqrt{13}-1}{3} = 0.8685\ldots. \tag{29}$$

Is it true that

$$m_2(B) \leq \frac{\sqrt{3}}{2} = 0.86602\ldots, \tag{30}$$

which is the Steiner ratio of the Euclidean plane?

8 The Steiner ratio of $\mathcal{L}_p^3$

In this section we will determine upper bounds for the Steiner ratio $m(3,p)$ of three-dimensional $\mathcal{L}_p$-spaces. Our goal is to estimate the quantities $m(3,p)$ with help of investigations for configurations of points in a regular situation

in the space $\mathcal{L}_p^3$. To do this we start with the consideration of tetrahedrons: Let $1 < p < \infty$ and consider the four points

$$\begin{aligned} v_1 &= (1,0,0), \\ v_2 &= (0,1,0), \\ v_3 &= (0,0,1) \quad \text{and} \\ v_4 &= (1,1,1) \end{aligned}$$

which build an equilateral set. Hence,

Theorem 8.1 *(Albrecht [1]) Let $1 < p < \infty$ and let q be conjugated to p. Then we have for the Steiner ratio of $\mathcal{L}_p^3$*

$$m(3,p) \leq \begin{cases} \frac{1}{3}\left(2^{-1/p} + (2^q - 1)^{1/q}\right) & : \quad \text{if} \quad 1 < p \leq \frac{\log 3}{\log 3 - \log 2} \\ \left(\frac{2}{3}\right)^{1/q} & : \quad \text{otherwise} \end{cases}$$

Another way: We consider a cross-polytope created by the set of nodes $N = \{v_1, \ldots, v_6\}$ whereby

$$\begin{aligned} v_1 &= (1,0,0), \\ v_2 &= (0,1,0), \\ v_3 &= (0,0,1), \\ v_4 &= (-1,0,0), \\ v_5 &= (0,-1,0) \quad \text{and} \\ v_6 &= (0,0,-1). \end{aligned}$$

The points v_i and v_j, $i \neq j$, have the distance

$$\rho(v_i, v_j) = \begin{cases} 2 & : \text{ if } |i-j| = 3 \\ 2^{1/p} \leq 2 & : \text{ otherwise} \end{cases}$$

and consequently an MST for N has the length

$$L_p(\text{MST for } N) = 5 \cdot 2^{1/p}. \tag{31}$$

On the other hand, using the fact that it holds $\rho(v_i, o) = 1$ for $i = 1, \ldots, 6$, there is a tree for $N \cup \{o\}$ of the length 6. Hence,

Theorem 8.2 *For the Steiner ratio of a three-dimensional $\mathcal{L}_p$-space it holds*

$$m(3,p) \leq \frac{6}{5} \cdot \left(\frac{1}{2}\right)^{1/p}.$$

Now, we will consider a cross-polytope in another way which gives better bounds for the Steiner ratio if $p > 2$.[7]
At first we assume that $p \neq \infty$ and consider the set $N = \{v_1, \ldots, v_6\}$ of given points with

$$\begin{aligned} v_1 &= (x_0, x_0 - 1, 1 - x_0), \\ v_2 &= (x_0, x_0, 2 - x_0), \\ v_3 &= (1, 0, 1), \\ v_4 &= (0, 0, 0), \\ v_5 &= (0, 1, 1) \quad \text{and} \\ v_6 &= (x_0 - 1, x_0, 1 - x_0), \end{aligned}$$

whereby x_0 denotes the unique zero of the function f with

$$f(x) = x^p + 2(x-1)^p - 2 \tag{32}$$

over the range (1,2). Here,

Theorem 8.3 *(Albrecht [1]) Let $1 < p < \infty$, $1/p + 1/q = 1$ and let x_0 be the unique determined zero of the function f defined in (32) over the range (1,2). Then the Steiner ratio of $\mathcal{L}_p^3$ can be estimated by*

$$m(3,p) \leq \begin{cases} \frac{1}{5}\left((2^q - 1)^{1/q} + \left(\frac{1}{2}\right)^{1/p} + \left(\frac{3}{2}\right)^{1/p} x_0\right) & : \quad 1 < p \leq \frac{\log 3}{\log 3 - \log 2} \\ \frac{1}{5}\left(\frac{3}{2}\right)^{1/p}(x_0 + 2) & : \quad \frac{\log 3}{\log 3 - \log 2} < p < \infty \end{cases}$$

9 The Steiner ratio of Euclidean spaces

In the d-dimensional Euclidean space, we consider the set N of $d+1$ nodes of a regular simplex with exclusively edges of unit length. Then an MST for N has the length d. It is easy to compute that the sphere that circumscribes N has the radius $R(N) = \sqrt{d/(2d+2)}$. With the center of this sphere as Steiner point, we find a tree T interconnecting N with the length $L_{B(2)}(T) = (d+1)R(N)$. Hence, we find the following nontrivial upper bound:

[7] But between $p = 2$ and $p \approx 2.0619508$ the bound will be greater than the bounds given before.

Theorem 9.1 *The Steiner ratio of the d-dimensional Euclidean space can be bounded as follows:*

$$m(d,2) \leq \sqrt{\frac{1}{2} + \frac{1}{2d}}. \tag{33}$$

In the proof we used a Steiner point of degree $d+1$, but it is well-known that all Steiner points in an SMT in Euclidean space are of degree 3. Consequently, the tree T described above is not an SMT for N, if $d > 2$. Better estimates for $m(d,2)$, we find in

Theorem 9.2 *(Chung, Gilbert [4], Smith [39] and Du, Smith [21]) The Steiner ratio of the d-dimensional Euclidean space is bounded as follows:*

dimension	*upper bound by Chung, Gilbert*	*upper bound by Smith*	*upper bound by Du, Smith*
$= 2$	0.86602...		
$= 3$	0.81305...	0.81119...	0.78419...
$= 4$	0.78374...	0.76871...	0.74398...
$= 5$	0.76456...	0.74574...	0.72181...
$= 6$	0.75142...	0.73199...	0.70853...
$= 7$	0.74126...	0.72247...	0.70012...
$= 8$	0.73376...	0.71550...	0.69455...
$= 9$	0.72743...	0.71112...	0.69076...
$= 10$	0.72250...		0.68812...
$= 11$	0.71811...		0.68624...
$= 20$	0.69839...		
$= 40$	0.68499...		
$= 80$	0.67775...		
$= 160$	0.67392...		
$\to \infty$	0.66984...		

The first column was computed by Chung and Gilbert considering regular simplices. Here, Du and Smith [21] showed that the regular d-simplex cannot achieve the Steiner ratio if $d > 2$. That means that these bounds cannot be the Steiner ratio of the space when $d > 2$.

The second column given by Smith investigates regular octahedra, respectively cross polytopes. Note, that it is not easy to compute an SMT for the nodes of an octahedra.

The third column used the ratio of sausages, whereby a sausage is constructed by

1. Start with a ball (of unit diameter) in $\mathcal{L}_2^d$;
2. Successively add balls so that the n'th ball you add is always touching the $\min\{d, n-1\}$ most recently added balls.

This procedure uniquely[8] defines an infinite sequence of interior-disjoint numbered balls. The centers of these balls form a discrete point set, which is called the (infinity) d-sausage $N(\infty, d)$. The first n points of the d-sausage will be called the "n-point d-sausage" $N(n, d)$. Note, that $N(d+1, d)$ is a d-simplex if $d \geq 3$.
Du and Smith [21] present many properties of the d-sausage, in particular, that

$$u(d) := \frac{L(\text{SMT for } N(\infty, d))}{L(\text{MST for } N(\infty, d))} \tag{34}$$

is a strictly decreasing function of the dimension d.[9] Hence, $u(d)$, $d = 2, 3, \ldots$ is a convergent sequence, but the limit is still unknown.
It seems that probably there does not exist a finite set of points in the d-dimensional Euclidean space, $d \geq 3$, which achieves the Steiner ratio $m(d, 2)$. But, if such set in spite of it exists, then it must contain exponentially many points. More exactly: Smith and McGregor Smith [41] investigate sausages in the three-dimensional Euclidean space to determine the Steiner Ratio and following they conjectured that for the Steiner Ratio of the three-dimensional Euclidean space

$$\begin{aligned} m(3,2) &= \sqrt{\frac{283}{700} - \frac{3\sqrt{21}}{700} + \frac{9\sqrt{11-\sqrt{21}}\sqrt{2}}{140}} \\ &= 0.78419\ldots \end{aligned}$$

holds.

Moreover, Du and Smith used the theory of packings to get the following

[8] up to congruence

[9] Here, we use a generalization of Steiner's Problem to sets of infinetly many points. This is simple to understand. For a finite number of points it is shown that

$$\frac{L(\text{SMT for } N(2d+1, d))}{L(\text{MST for } N(2d+1, d))} \leq \frac{L(\text{SMT for } N(d+1, d))}{L(\text{MST for } N(d+1, d))},$$

which is a finite version of

$$\frac{L(\text{SMT for } N(\infty, d))}{L(\text{MST for } N(\infty, d))} \leq \frac{L(\text{SMT for } N(d+1, d))}{L(\text{MST for } N(d+1, d))},$$

for $d > 1$.

Theorem 9.3 *(Du, Smith [21]) Let N be a finite set of n points in the d-dimensional Euclidean space $M_d(B(2))$, $d \geq 3$, which achieves the Steiner ratio $m_d(B(2))$ of the space. Then*

$$n \geq \left\lceil \frac{1}{2} \cdot \sqrt{f\left(\frac{\pi}{3}, d\right)} \right\rceil + 1,$$

where

$$f(\theta, d) = \frac{2I_{d-2}(\pi/2)}{I_{d-2}(\theta)}$$

and

$$I_m(x) = \int_0^x (\sin u)^m \, du.$$

9.3 implies that the number n grows at least exponentially in the dimension d. Some numbers are computed:

$d =$	n is at least
49	49
50	53
100	2218
200	3481911
500	10^{16}
1000	$5 \cdot 10^{31}$

10 The Steiner ratio of $\mathcal{L}_p^d$

For our investigations we have the following facts: Let $1 < p < \infty$ and $d \geq 3$. Then there are in $\mathcal{L}_p^d$ at least $d+1$ equidistant points. This can be seen with the following considerations: For $i = 1, \ldots, d$ let

$$v_i = (x_{i,1}, \ldots, x_{i,d})$$

with

$$x_{i,j} = \begin{cases} 1 & : \text{ if } i = j \\ 0 & : \text{ otherwise.} \end{cases}$$

These are d points with $||v_i - v_j|| = 2^{1/p}$ for all $1 \leq i < j \leq d$.
For the point $v = (x, \ldots, x)$ it holds $||v - v_i|| = ||v - v_j||$ for all $1 \leq i, j \leq d$.
To create $||v - v_i|| = 2^{1/p}$ the value x has to fulfill the equation

$$((d-1)|x|^p + |1-x|^p)^{1/p} = 2^{1/p}.$$

This we can realize by the fact that the function $f : [0,1] \to \mathbb{R}$ with

$$f(x) = ((d-1)x^p + (1-x)^p)^{1/p} - 2^{1/p}$$

has exactly one zero in [0,1].

Theorem 10.1 *(Albrecht [1], [2]) Let* $1 < p < \infty$ *and* $d \geq 3$. *Then*

$$m(d,p) \leq \frac{d+1}{2d} \cdot \left(\frac{d}{2}\right)^{1/p}.$$

Moreover, when we use an idea by Liu and Du [35] for the planar case extending to an approach using equilateral sets and a "center" we find:

Theorem 10.2 *(Albrecht [1], [2]) Let* $1 < p < \infty$. *Then*

$$m(d,p) < \frac{d+1}{d} \cdot \left(\frac{1}{2}\right)^{1/p}.$$

This bound is not sharp, since the estimation of the distance of the points to the center is to inefficient, at least for small dimensions. On the other hand, we only use one additional point, and it is to assume that more than one of such points decrease the length.

Now, we compare the bounds given in 10.1 and 10.2. Obviously,

$$\frac{d+1}{2d} \cdot \left(\frac{d}{2}\right)^{1/p} \leq \frac{d+1}{d} \cdot \left(\frac{1}{2}\right)^{1/p}$$

holds if and only if

$$d \leq 2^p.$$

Hence,

Corollary 10.3 *Looking for the Steiner ratio of high dimensional* $\mathcal{L}_p$*-spaces we have only to consider the bound given in 10.2.*

11 When the dimension runs to infinity

What do we know of the development of the Steiner ratio of Banach-Minkowski spaces if the dimension d runs to infinity? First, we consider the spaces $\mathcal{L}_p^d$, namely

$$\underline{m}(p) = \lim_{d \to \infty} m(d,p). \tag{35}$$

We know the following values and estimates for $\underline{m}(p)$:

$p =$	lower bound	exact value	upper bound	source
1		$\frac{1}{2} = 0.5$		4.2
2	$\frac{1}{\sqrt{3}} = 0,57735\ldots$		$0.66984\ldots$	9.2
∞		$\frac{1}{2} = 0.5$		calculation
arbitrary	$\frac{1}{2} = 0.5$		$\left(\frac{1}{2}\right)^{1/p}$	10.2

Moreover, combining all facts, we have

Theorem 11.1 *Let* $\underline{m}(p) = \lim_{d\to\infty} m(d,p)$. *Then it holds*

$$\frac{1}{2} \leq \underline{m}(p) \leq \min\left\{\left(\frac{1}{2}\right)^{1/p}, \underline{m}(2)\right\}$$

for any real $1 < p < \infty$, *and*

$$\underline{m}(1) = \underline{m}(\infty) = \frac{1}{2}.$$

Above we discussed the limiting process for the class of all Banach-Minkowski spaces considering the sequence $\{C_d\}_{d=1,2,\ldots}$ whereby

$$C_d = \sup\{m_d(B) : B \in \underline{B}_d\}.$$

Moreover, we find that our conjectures from the end of section 4 are true if we go to infinity.

Theorem 11.2 *(C. [12])*

$$\lim_{d\to\infty} C_d = \lim_{d\to\infty} m(d,2),$$

whereby

$$0.57735\ldots \leq \lim_{d\to\infty} m(d,2) \leq 066984\ldots;$$

and

$$\lim_{d\to\infty} c_d = \lim_{d\to\infty} m(d,1) = 0.5.$$

References

[1] J. Albrecht. *Das Steinerverhältnis endlich dimensionaler L_p-Räume.* Master's thesis, Ernst-Moritz-Arndt-Universität Greifswald, 1997. Diplomarbeit.

[2] J. Albrecht and D. Cieslik. The Steiner ratio of finite dimensional L_p spaces. In D.Z. Du, J.M. Smith, and J.H. Rubinstein, editors, *Advances in Steiner Trees,* pages 1–13. Kluwer Academic Publishers, 2000.

[3] D. Cheriton and R.E. Tarjan. Finding Minimum Spanning Trees. *SIAM J. Comp.,* 5:724–742, 1976.

[4] F.R.K. Chung and E.N. Gilbert. Steiner Trees for the regular simplex. *Bull. of the Inst. of Math. Ac. Sinica,* 4:313–325, 1976.

[5] F.R.K. Chung and F.K. Hwang. A lower bound for the Steiner Tree Problem. *SIAM J. Appl. Math.,* 34:27–36, 1978.

[6] F.R.K. Chung and R.L. Graham. A new bound for Euclidean Steiner Minimal Trees. *Ann. N.Y. Acad. Sci.,* 440:328–346, 1985.

[7] D. Cieslik. The Steiner-ratio in Banach-Minkowski planes. In R. Bodendieck, editor, *Contemporary Methods in Graph Theory,* pages 231–247. Bibliographisches Institut (BI), Mannheim, 1990.

[8] D. Cieslik. The Steiner ratio of $\mathcal{L}_p^2$. In U.Faigle and C.Hoede, editors, *3rd Twente Workshop on Graphs and Combinatorial Optimization,* pages 31–34, 1993. Memorandum no 1132.

[9] D. Cieslik. Graph-theoretical method to approximate Steiner Minimal Trees in Banach-Minkowski Planes. In D.Z. Du and X.S. Zhang and W. Wang, editors, *Lecture Notes in Operations Research,* 1:213–220, 1995.

[10] D. Cieslik. *Steiner Minimal Trees.* Kluwer Academic Publishers, 1998.

[11] D. Cieslik. The Steiner ratio of $\mathcal{L}_{2k}^d$. *Applied Discrete Mathematics,* 95:217-221, 1999.

[12] D. Cieslik. Using Dvoretzky's theorem in network design. *Journal of Geometry,* 65:7-8, 1999.

[13] D. Cieslik and J. Linhart. Steiner Minimal Trees in $\mathbf{L}_p^2$. *Discrete Mathematics,* 155:39–48, 1996.

[14] E.J. Cockayne. On the Steiner Problem. *Canad. Math. Bull.,* 10:431–450, 1967.

[15] D.Z. Du, E.Y. Yao, and F.K. Hwang. A Short Proof of a Result of Pollak on Steiner Minimal Trees. *J. Combin. Theory,* Ser. A, 32:396–400, 1982.

[16] D.Z. Du and F.K. Hwang. A new bound for the Steiner Ratio. *Trans. Am. Math. Soc.,* 278:137–148, 1983.

[17] D.Z. Du, F.K. Hwang, and E.N. Yao. The Steiner ratio conjecture is true for five points. *J. Combin. Theory,* Ser. A,38:230–240, 1985.

[18] D.Z. Du and F.K. Hwang. An Approach for Proving Lower Bounds: Solution of Gilbert-Pollak's conjecture on Steiner ratio. In *Proc. of the 31st Ann. Symp. on Foundations of Computer Science,* St. Louis, 1990.

[19] D.Z. Du and F.K. Hwang. A Proof of the Gilbert-Pollak Conjecture on the Steiner Ratio. *Algorithmica,* 7:121–136, 1992.

[20] D.Z. Du, B. Gao, R.L. Graham, Z.C. Liu, and P.J. Wan. Minimum Steiner Trees in Normed Planes. *Discrete and Computational Geometry,* 9:351–370, 1993.

[21] D.Z. Du and W.D. Smith. Disproofs of generalized Gilbert-Pollak conjecture on the Steiner ratio in three or more dimensions. *J. of Combinatorial Theory, A,* 74:115–130, 1996.

[22] P. Fermat. *Abhandlungen über Maxima und Minima.* Number 238. Oswalds Klassiker der exakten Wissenschaften, 1934.

[23] B. Gao, D.Z. Du and R.L. Graham. A Tight Lower Bound for the Steiner Ratio in Minkowski Planes. *Discrete Mathematics,* 142:49–63, 1993.

[24] M.R. Garey, R.L. Graham, and D.S. Johnson. The complexity of computing Steiner Minimal Trees. *SIAM J. Appl. Math.,* 32:835–859, 1977.

[25] M.R. Garey and D.S. Johnson. The rectilinear Steiner Minimal Tree Problem is $\mathcal{NP}$-complete. *SIAM J. Appl. Math.,* 32:826–834, 1977.

[26] M.R. Garey and D.S. Johnson. *Computers and Intractibility.* San Francisco, 1979.

[27] C.F. Gauß. Briefwechsel Gauß-Schuhmacher. In *Werke Bd. X, 1,* pages 459–468. Göttingen, 1917.

[28] E.N. Gilbert and H.O. Pollak. Steiner Minimal Trees. *SIAM J. Appl. Math.,* 16:1–29, 1968.

[29] R.L. Graham and P. Hell. On the History of the Minimum Spanning Tree Problem. *Ann. Hist. Comp.,* 7:43–57, 1985.

[30] R.L. Graham and F.K. Hwang. A remark on Steiner Minimal Trees. *Bull. of the Inst. of Math. Ac. Sinica,* 4:177–182, 1976.

[31] F.K. Hwang. On Steiner Minimal Trees with rectilinear distance. *SIAM J. Appl. Math.,* 30:104–114, 1976.

[32] F.K. Hwang, D.S. Richards, and P. Winter. *The Steiner Tree Problem.* North-Holland, 1992.

[33] A.O. Ivanov and A.A. Tuzhilin. *Minimal Networks - The Steiner Problem and Its Generalizations.* CRC Press, Boca Raton, 1994.

[34] J.B. Kruskal. On the shortest spanning subtree of a graph and the travelling salesman problem. *Proc. of the Am. Math. Soc.,* 7:48–50, 1956.

[35] Z.C. Liu and D.Z. Du. On Steiner Minimal Trees with L_p Distance. *Algorithmica,* 7:179–192, 1992.

[36] H.O. Pollak. Some remarks on the Steiner Problem. *J. Combin. Theory,* Ser. A,24:278–295, 1978.

[37] J.H. Rubinstein and D.A. Thomas. The Steiner Ratio conjecture for six points. *J. Combin. Theory,* Ser. A,58:54–77, 1991.

[38] M. Sarrafzadeh and C.K. Wong. Hierarchical Steiner tree construction in uniform orientations. Preprint.

[39] W.D. Smith. How to find Steiner Minimal Trees in Euclidean d-Space. *Algorithmica,* 7:137–178, 1992.

[40] J.M. Smith. Steiner Minimal Trees in e^3: Theory, Algorithms, and Applications. In D.Z. Du and P.M. Pardalos, editors, *Handbook of Combinatorial Optimization,* volume 2, pages 397–470. Kluwer Academic Publishers, 1998.

[41] W.D. Smith and J.M. Smith. On the Steiner Ratio in 3-space. *J. of Combinatorial Theory, A,* 65:301–332, 1995.

[42] W. Stromquist. The maximum distance between two dimensional Banach-spaces. *Math. Scand.,* 48:205–225, 1981.

[43] P.J. Wan, D.Z. Du, and R.L. Graham. The Steiner ratio of the Dual Normed Plane. *Discrete Mathematics,* 171:261–275, 1997.

HANDBOOK OF COMBINATORIAL OPTIMIZATION (VOL 5)
D.-Z. Du and P.M. Pardalos (Eds.) pp. 83-191

Probabilistic Verification and Non-Approximability

Mario Szegedy
Department of Computer Science
Rutgers, the State University of NJ
110 Frelinghuysen Road,
Piscataway, NJ 08854-8019
E-mail: szegedy@cs.rutgers.edu

Contents

1 Introduction

Probabilistic verification in the broad sense includes the zero knowledge proof systems of Golwasser, Mikhali and Rackoff [80], the Arthur-Merlin games of Babai and Moran [25], various multi-prover games and Probabilistically Checkable Proofs (PCP), which will be the focus of our investigations.

In 1991 Feige, Goldwasser, Lovász, Safra and Szegedy [62] showed that results about multi-prover systems, or equivalently PCPs imply the relative hardness of approximating the maximum clique number of a graph. Their discovery married the subject of probabilistic verification with the theory of NP optimization for a long time if not forever. One can call this bond the one in between the beauty and the beast, where the later is the monstrous body of code constructions and combinatorial-algebraic lemmas we need for the PCP theorems. Not surprisingly most surveys like to display the beauty, and keep the beast at bay. With this chapter it is our goal to give a fair treatment to both.

The basic PCP theorem states that every proof has a probabilistically checkable version where there is a verifier who looks only at a constant number of bits and tells with high certainty whether the proof is correct. We give a complete proof of the basic theorem that, we hope, highlights the necessary technical steps of the proof. We do not turn the beast into a prince, but at least we groom it a little bit. For those readers' sake who prefer to skip the specifics of the constructions we made the construction part self-contained, and placed it to the end of the entire text. This way the proof of the basic theorem shrinks to a thin section (Section 2.4) which uses the former as a black box. Not counting the appendix style addendum of the code constructions the text is composed of three parts.

1. *Probabilistic verification (Section 2):* We start this part with basic definitions and the history of probabilistic verification. In Section 2.3 the reader can find a short explanation of the language cryptographers use. The main mathematical content of the verification part is the topmost layer of the proof of the basic PCP theorem. This is different from what is considered the topmost layer in other papers and reflects the writer's special mathematical taste.

2. *Non-approximability (Section 3):* Sections in the non-approximability part represent one of two didactic approaches. Sections **??**, 3.2, 3.3, 3.4 and 3.5 give general methods for proving hardness of approximation. In contrast, Sections 3.6, 3.7, 3.8 and 3.9 give details on specific NP optimization problems. It was our choice to focus only on a few number of problems, namely MaxClique, the chromatic number, the set cover and various constraint satisfaction problems, and to leave out many interesting ones. Since new advances in the PCP theory are tied to non-approximability issues, they can also be found in Section 3, either together with the corresponding non-approximability results or in one of the general sections. Of course we had to make a concession and omit the hard proofs in this part, like that of

the breath-taking result of Håstad that if $NP \not\subseteq ZPP$ then the MaxCliqe cannot be efficiently approximated within a factor of $n^{1-\epsilon}$ [85].

3. *Structure theoretical consequences (Section 4):* The theory of approximation preserving reductions and the structure theory for non-approximability classes have been developing at a rapid rate due to breakthroughs in the theory of PCP. We present two completeness theorems that well signify this development: MAX SAT is complete for APX-PB with respect to the E reduction, and MAX SAT is complete for APX with respect to the AP reduction.

1.1 Literature and Acknowledgments

Several excellent articles deal with the PCP theory and NP optimization. Among them are the survey article of László Babai, that he wrote for the European Congress of Mathematics [20], a relatively early work of Bellare, O. Goldreich and M. Sudan [31], which is an original paper, but well summarizes results up to that stage; the survey article of Arora and Lund [9], which mostly deals with non-approximability; the theses of Arora [5] and Sudan [131] just to mention a few. There is a book on the subject by G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi [15], which also contains a compendium of NP optimization problems. Many good non-approximability results are not included in any of the surveys, and it is often best to reach for the original source. The theory of reductions get a very good treatment in Trevisan's works [54, 55, 99, 45], including his thesis [136].

The author is grateful to Johan Håstad for letting him use his unpublished proof of Feige's set cover result. Many thanks go to Marek Karpinski for making detailed comments on the first manuscript. Remarks or references coming from Uri Feige, Luca Trevisan improved on the quality of the text.

2 The Basics of PCP Theory

The theory of PCP has at least four interpretations:

- Construction of very efficiently checkable mathematical proofs
- Design of a generic mechanism for instant checking of computations
- New definitions for the class NP

- NP hardness of approximating certain optimization problems

The first two points are explained in Babai, Fortnow, Levin and Szegedy [23] and further developed in [120], the third point was made by Arora and Safra [12], and the fourth point is the subject of a great number of papers starting with [62]. The theory developed so quickly in the beginning of the 90's and so many researchers gave their contributions to it, that it has became a loose collection of a technically very remarkable, but occasionally handwaivingly written results.

In this chapter we would like to put the basics of the theory in a unified framework. The construction of such framework is difficult because its presentation requires concepts from different branches of mathematics and theoretical computer science such as the theory of NP, cryptography, combinatorics, algebra and probability theory. We tried to separate those components of the theory that can be dealt with within one or two of these paradigms. In particular, we eliminated the language of cryptography from the most technical component, and treated it as an independent theory in Section 5.

We discuss only the results of Arora and Safra [12] and Arora, Lund, Motwani, Sudan and Szegedy [10] and their prerequisites in details. What justifies our choice for drawing the line in between the basics and the more advanced part exactly here is that most subsequent results rely upon the above two in an essential way.

2.1 Probabilistically Checkable Proofs

Probabilistic verification stands in contrast with deterministic verification. Recall that every language $L \in NP$ is deterministically verifiable in polynomial time, meaning that for every $L \in NP$ there is a polynomial time machine V such that if $x \in L$ then there exists a a membership proof P such that $V(x, P) = 1$ and if $x \notin L$ then for every P, $V(x, P) = 0$. The concept of probabilistic verification is more complicated, but analogous:

Definition 2.1 (Probabilistic Verification) *For functions $f, g : \mathbf{N} \to \mathbf{N}$, a probabilistic polynomial-time verifier $V(x, P, r)$ is (f, g)-restricted if, for every input x of size n, it uses at most $f(n)$ random bits and examines at most $g(n)$ bits in the membership proof while checking it. Let Σ be an alphabet and $L \subseteq \Sigma^*$. V is an (f, g)-restricted polynomial-time probabilistic verifier for L if there exists an $\epsilon > 0$ such that for every input x:*

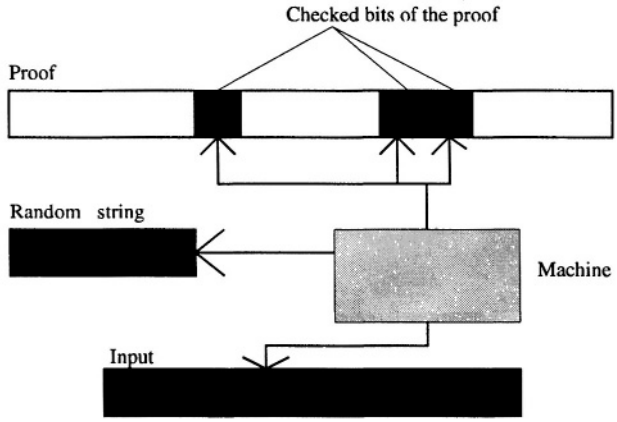


Figure 1: A schematic picture of a probabilistic verifier. Only the black bits are read for a fixed random choice r.

- *if $x \in L$, then there exists a membership proof P such that:*

$$Prob_r(V(x, P, r) = 1) = 1,$$

(i.e., accepts for every choice of its random bits);

- *if $x \notin L$, then for any membership proof P:*

$$Prob_r(V(x, P, r) = 1) \leq 1 - \epsilon.$$

Here we have to make a couple of comments. First, it is best to think of V as a random access machine. Since Turing machines and random access machines are polynomial time equivalent, we may also view V as a Turing machine, but we have to assume that V has random access to the bits of P. Another issue is the *adaptivity* of V. Are the bits of P which V accesses determined in advance from x and r, or the location of the second bit read from P may depend on value of the first bit read from P, etc.? In the first case we call the verifier *non-adaptive,* and in the second case we call the verifier *adaptive.* Most verifier constructions in the PCP theory are non-adaptive.

Definition 2.2 (PCP Classes[12]) *A language L is in the class $PCP(f, g)$ iff there exists a constant $c > 0$ and an $(f(|x|^c), g(|x|^c))$-restricted polynomial-time probabilistic verifier for L.*

Clearly, $NP \subseteq PCP(0, |x|)$.

Lemma 2.3 *If* $L \in PCP(f, g)$ *and* L_1 *many-one reduces to* L *then* $L_1 \in PCP(f, g)$.

Proof. Since L_1 many-one reduces to L, there is a polynomially computable T such that for every $x \in \Sigma^*$: $T(x) \in L$ iff $x \in L_1$. Let $V(x, P, r)$ be a probabilistic verifier for L. Then $V(T(x), P, r)$ is a probabilistic verifier for L_1. □

The PCP notation allows a compact description of many known results:

Arora=A, Babai=B, Feige=Fe, Fortnow=F, Goldwasser=G Levin=Le, Lovasz=Lo, Lund=Lu, Motwani=M, Safra=Sa Sudan=Su, Szegedy=Sz	
$\text{NEXP} = \text{PCP}(n, n)$	BFLu [22]
$\text{NP} \subseteq \text{PCP}(\log n \log\log n, \log n \log\log n)$	BFLeSz [23]
$\text{NP} \subseteq \text{PCP}(\log n \,\text{loglog}\, n, \log n \,\text{loglog}\, n)$	FeGLoSaSz [62]
$\text{NP} = \text{PCP}(\log n, \sqrt{\log n})$	ASa [12]
$\text{NP} = \text{PCP}(\log n, 3)$	ALuMSuSz [10]

In Section 2.4 we are going to prove the last one ([10]) which we also call the basic PCP theorem because it is the basis of many further improvements. To describe these improvements we need to introduce new parameters such as free bit complexity, amortized complexity, and their combinations. We can further parameterize PCP classes by allowing the acceptance and rejection probabilities to vary rather than fixing them to 1 and $1 - \epsilon$. The various parameters will be discussed in details in Section 3.3.

2.2 A Brief History of Probabilistic Verification

The usual definition of NP uses the notion of a deterministic verifier who checks membership proofs of languages in polynomial time. Goldwasser, Micali and Rackoff [80] and Babai [17, 25] were the first who allowed the verifier to be a probabilistic polynomial-time Turing Machine that interacts with a "prover," which is an infinitely powerful Turing Machine trying to convince the verifier that the input x is in the language. A surprising result due to Lund, Fortnow, Karloff and Nisan [107, 18] and Shamir [129] has shown that every language in PSPACE — which is suspected to be a much larger class

than NP — admits such "interactive" membership proofs. Another variant of proof verification, due to Ben-Or, Goldwasser, Kilian and Wigderson [35], involves a probabilistic polynomial-time verifier interacting with more than one mutually non-interacting provers. The class of languages with such interactive proofs is called MIP (for Multi-prover Interactive Proofs). Fortnow, Rompel and Sipser [66] gave an equivalent definition of MIP as languages that have a probabilistic polynomial-time oracle verifier that checks membership proofs (possibly of exponential length) using *oracle access* to the proof.

Babai, Fortnow and Lund [22] showed that MIP is exactly NEXP, the class of languages for which membership proofs can be checked deterministically in exponential time. This result is surprising because NEXP is just the exponential analogue of NP, and its usual definition involves no notion of randomness or interaction. Therefore researchers tried to discover if the MIP = NEXP result can be "scaled-down" to say something interesting about NP. Babai, Fortnow, Levin and Szegedy [23] introduced the notion of *transparent* membership proofs, namely, membership proofs that can be checked in poly-logarithmic time, provided the input is encoded with some error-correcting code. They showed that NP languages have such proofs. Feige et al. [62] showed a similar result, but with a somewhat more efficient verifier. Arora and Safra [12] further improved the efficiency of checking membership proofs for NP languages up to the point that they were able to give a new definition for NP this way.

They define a parameterized hierarchy of complexity classes called PCP (for Probabilistically Checkable Proofs). This definition uses the notion of a "probabilistic oracle verifier" of Fortnow et al. [66] and classifies languages based on how efficiently such a verifier can check membership proofs for them. The notion of "efficiency" refers to the number of random bits used by the verifier as well as the number of bits it reads in the membership proof. Note that we count only the bits of the *proof* that are read - not the bits of the *input* which the verifier is allowed to read fully. The definition of a class very similar to PCP was implicit in the work of Feige et al. [62].

The paper of Arora, Lund, Motwani, Sudan and Szegedy [10] in 1992 building on [12] was the first to decrease the number of check bits to constant. It is not just philosophically important that every transparently written proof can be checked with high accuracy by looking only at a constant number of places in the proof, but the construct of ALMSS also serves as a building block for nearly every construct that comes after it. Later results were greatly motivated by their non-approximability consequences.

2.3 The Language of Cryptography

Articles in cryptography often describe mathematical objects such as information, knowledge, and secret through interactions between human characters. While the framework originally was used almost exclusively by cryptographers, it has become an integral part of the entire computer science culture.

In the remaining part of the text we are going to need expressions such as "true prover," "verifier," and "cheating prover." But what do they mean exactly? When we prove that a probabilistic verifier V recognizes a language L, our argument consists of two parts. In the first part we show that if $x \in L$, the verifier accepts with probability 1. In the second part we show that if $x \notin L$ then the verifier rejects with probability at least ϵ.

The first part of the argument features the *true prover* who is responsible to provide a string P which is accepted by the verifier with probability one. In formula: $\text{Prob}_r(V(x, P, r) = 1) = 1$. The presumed structure of P is expressed in terms of the actions of the true prover. For instance, when we say that "the good prover encodes string a using an encoding E," it means that the presumed structure of the proof is a code word $E(a)$.

When we hypothesize that $x \notin L$, the main character is the *cheating prover*. By assumption he is an ultimately devious creature whose proof does not adhere to the structure of the true prover. He has to cheat somewhere, since no correct proof exists when $x \notin L$. This property is required by our notion of a good proof system. The goal of the cheating prover is to maximize the probability with which the verifier accepts P, the "proof" he presents. It is important to emphasize that the proof of the cheating prover is an arbitrary string formed from the letters of the given alphabet, typically $\Sigma = \{0, 1\}$. When we argue about the $x \in L$ case we never need to talk about the cheating prover.

Like in the case of other formal proof systems in the case of PCPs, the proof itself is just a vehicle to indicate that a statement holds, and itself is not interesting for the verifier at all. Only the mere existence or non-existence of its corrected version matters. The verifier does not want to learn the entire proof or to verify that the proof adheres to the structure of the good prover *everywhere*. It is sufficient for us to show that if $x \notin L$, no matter what string the cheating prover may write down, it is so far from being consistent either with the structure of the true prover or with other constraints defined by input x that the verifier is able to catch the error with probability ϵ.

2.4 The Proof of the Basic PCP Theorem

Theorem 2.4 $NP \subseteq PCP(\log n, 3)$. *Moreover we can also assume that the verifier is non-adaptive and his accepting criterion is a disjunct (i.e. "or") of the literals made from the three bits he looks at.*

Notice that by Definition 2.2 the factor $\log n$ scales by a constant, but not the second argument. First we show that

Lemma 2.5 $NP \subseteq PCP(\log n, c)$ *for some fixed constant* c.

This lemma is equivalent to Theorem 2.4 and it is also often called the basic PCP theorem. We however prefer to call the stronger form this way, which is implied from the seemingly weaker form by Lemma 2.10.

Proof. [of Lemma 2.5] By Lemma 2.3 it is sufficient to present a probabilistic verifier for a specific NP complete problem.

Definition 2.6 (QSAT) *The input instance of QSAT over* $\mathbf{F}_2$ *is a set of quadratic equations* $\Psi = \{\sum_{1\le j\le n} a_{i,j}x_j + \sum_{1\le k,l\le n} a_{i,k,l}x_k x_l = c_i \mid 1 \le i \le m\}$, *where all coefficients are from* $\mathbf{F}_2$. Ψ *is accepted iff there are* $x_1, \ldots, x_n \in \mathbf{F}_2$ *that satisfy all equations of* Ψ *simultaneously.*

Lemma 2.7 *QSAT is NP complete.*

Proof. In order to see that *QSAT* is NP complete we make a derivation to it from 3SAT. Let $F = C_1 \wedge C_2 \wedge \ldots \wedge C_m$ be a 3SAT instance, so that each clause C_i contains three literals. We keep the names of the original variables and add m auxiliary variables z_i $(1 \le i \le m)$. Then we arithmetize each clause as the following example shows: If $C_i = x_1 \vee x_2 \vee x_3$, we turn C_i into $x_1 \vee x_2 = z_i$ and $z_i \vee x_3 = 1$, and express these equations arithmetically over $\mathbf{F}_2$ as $x_1 + x_2 - x_1x_2 - z_i = 0$ and $z_i + x_3 - z_ix_3 - 1 = 0$. If x_j is negated in C_i we replace x_j with $1 - x_j$ in the expressions. Clearly, F is satisfiable if and only if the $2m$ quadratic equations we constructed above are simultaneously satisfiable. □

Next we give a necessary and a sufficient conditions to $\Psi \in QSAT$. Let y_i $(1 \le i \le n)$, $y_{i,j}$ $(1 \le i, j \le n)$ be $n + n^2$ elements of $\mathbf{F}_2$. Define:

$$\begin{aligned} \mathbf{y} &= (y_j)_{1\le j\le n} \quad \text{(a row vector of length } n\text{)}, \\ Y &= (y_{k,l})_{1\le k,l\le n} \quad \text{(an } n \times n \text{ matrix)}, \end{aligned}$$

$$\mathbf{s}_\Psi = \left(\sum_{1 \le j \le n} a_{i,j} y_j + \sum_{1 \le k,l \le n} a_{i,k,l} y_{k,l} \right)_{1 \le i \le m} \quad \text{(a vector of length } m\text{)}$$

$$\mathbf{c}_\Psi = (c_i)_{1 \le j \le m} \text{ (a vector of the right hand sides of } \Psi\text{)}.$$

Clearly, $\Psi \in QSAT$ iff a $\mathbf{y}$ exists which satisfies the equations

$$\mathbf{s}_\Psi = \mathbf{c}_\Psi, \tag{1}$$

$$Y = \mathbf{y}^T \mathbf{y}. \tag{2}$$

We will encode these equations using error correcting codes. Let M be the matrix of an $(m_1, m, m_1/4)$ error correcting code over $\mathbf{F}_2$, and N be the matrix of an $(n_1, n, n_1/4)$ error correcting code over $\mathbf{F}_2$, such that both codes are efficiently constructible, n_1 is polynomial in n and m_1 is polynomial in m. We state the existence of these codes in Proposition 5.25 in Section 5.1.8. Equations (1) and (2) are satisfied by some $\mathbf{y}$ if and only if

$$\mathbf{s}_\Psi M = \mathbf{c}_\Psi M \tag{3}$$

$$N^T Y N = N^T \mathbf{y}^T \mathbf{y} N \tag{4}$$

hold for the same y, since the M and N have full row-rank. We also claim that

Lemma 2.8 *If $\Psi \notin QSAT$ then for every $\mathbf{y} \in \mathbf{F}_2^n$, $Y \in \mathbf{F}_2^{n^2}$ at least one of the following holds:*

1. *$\mathbf{s}_\Psi M$ and $\mathbf{c}_\Psi M$ differ in at least $m_1/4$ entries.*
2. *$N^T Y N$ and $N^T \mathbf{y}^T \mathbf{y} N$ differ in at least $n_1^2/16$ entries.*

Proof. Let $\mathbf{y} = (y_i)_{1 \le i \le n}$, $Y = (y_{i,j})_{1 \le i,j \le n}$ be arbitrary. If $y_{i,j} = y_i y_j$ for $1 \le i, j \le n$ then we have that $\mathbf{s}_\Psi \neq \mathbf{c}_\Psi$, since Ψ is not satisfiable. Then M takes $\mathbf{s}_\Psi$ and $\mathbf{c}_\Psi$ to two different code words. By definition these code words differ in at least $m_1/4$ places.

Consider now the case when there is an $1 \le i, j \le n$ pair such that $y_{i,j} \neq y_i y_j$. Then $Y \neq \mathbf{y}^T \mathbf{y}$, or in other words, $Y - \mathbf{y}^T \mathbf{y} = X$, where X is not the identically zero matrix. We claim that in this case $N^T X N$ has at least $n_1^2/16$ non-zero entries implying that 2. holds. Indeed, if X is non zero, then XN has a row with at least $n_1/4$ ones in it. This follows from

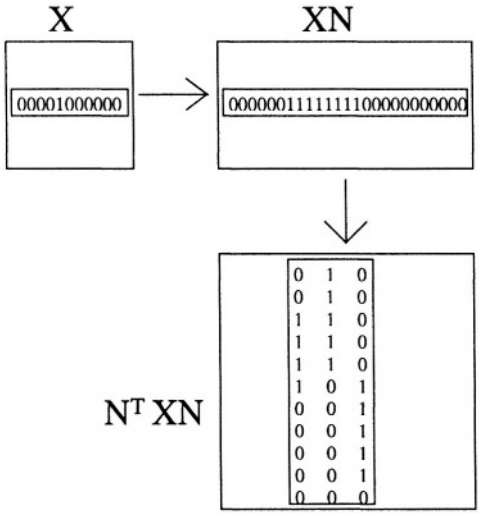


Figure 2: If we start form any matrix X of size $n \times n$ with at least one 1 in it, the the fraction of the non-zero entries of $N^T XN$ is at least $\frac{1}{16}$

the error correction property of N, and from the fact that it can be thought of being applied to each row of X separately. Thus at least $n_1/4$ columns of XN are not identically zero. When multiplied from the left by N^T, each such column is taken into a column of $N^T XN$ which contains at least $n_1/4$ non-zeros (see Figure 2). The $n_1^2/16$ lower bound on the number of ones in $N^T XN$ follows. □

Naive Protocol: In the first attempt we can try to construct a PCP for the QSAT by requesting that the prover writes down the entries of $\mathbf{s}_\Psi M$, $N^T YN$, and $\mathbf{y}N$. The verifier checks:

1. a random entry of $\mathbf{s}_\Psi M$ and compares it with the corresponding entry of $\mathbf{c}_\Psi M$ (the later the verifier can compute without help from the prover).
2. a random entry with index j, k of $N^T YN$ and compares it with the product of the j^{th} and k^{th} entries of $\mathbf{y}N$.

The verifier accepts if both comparisons give equality, otherwise rejects. It follows from Equations (3) and (4) that if $\Psi \in QSAT$ and $\mathbf{y}$ is a solution for

Ψ, $Y = \mathbf{y}^T\mathbf{y}$, the verifier accepts with probability 1. It follows from lemma 2.8 that if $\Psi \notin QSAT$ no matter what $\mathbf{y}$ and Y the prover starts from, the verifier rejects with probability at least 1/16. Moreover, the verifier looks at at most four bits of the proof. The problem, however with this proof system is that the bits of $\mathbf{s}_\Psi M$, $N^T Y N$, and $\mathbf{y}N$ must satisfy certain linear relations. Therefore a cheating prover may mislead the verifier by presenting a "proof" where these relations do not hold. To remedy this problem in Section 5.3 we construct codes that consistently encode linear combinations of variables or else an error is detected. More precisely:

Lemma 2.9 (Basic Encoding Lemma) *For some fixed $\delta > 0$, $c > 0$ the following holds. Let $\{L_1 \ldots, L_p\}$ be arbitrary linear functions from $\mathbf{F}_2^k$ to $\mathbf{F}_2$ $(p \geq k)$. There is a $q \leq p^c$, a linear encoding $E : \mathbf{F}_2^k \to \mathbf{F}_2^q$, a decoding function $D : \mathbf{F}_2^q \to \mathbf{F}_2^k$ and a procedure $P^{\mathbf{w}}(r, i)$, which recovers $L_i(D(\mathbf{w}))$ from any $\mathbf{w} \in \mathbf{F}_2^q$ in the following weak sense: P inputs a number $1 \leq i \leq p$, selects a random number $1 \leq r \leq p^c$, reads c entries of $\mathbf{w} \in \mathbf{F}_n^n$ and outputs either an element of $\mathbf{F}_2$ or a symbol* "Reject," *such that:*

1. *If* w = E(y) *then the probability that P on input i outputs $L_i(\mathbf{y})$ is 1.*
2. *For any $\mathbf{w} \in \mathbf{F}_2^q$ and for any $1 \leq i \leq p$:*

$$\begin{aligned} Prob_{1 \leq r \leq p^c}(P^{\mathbf{w}}(r, i) = \text{"Reject"}) \quad &+ \\ Prob_{1 \leq r \leq p^c}(P^{\mathbf{w}}(r, i) = L_i(D(\mathbf{w}))) \quad &\geq \quad \delta \end{aligned}$$

Moreover P works in time polynomial in p.

The theorem is a consequence of Theorem 5.56 of Section 5.3.5 and the Decoding theorem for holographic codes (Section 5.3.6, Lemma 5.58). All constructions of Section 5 are explicit, so the decoder P of Lemma 2.9 works in polynomial time.

Correct Protocol: The prover is asked to encode $\mathbf{y}$ and Y with the code of Lemma 2.9 that is created specifically to retrieve the entries of $\mathbf{s}_\Psi M$, $N^T Y N$, and $\mathbf{y}N$. Note that these entries are all linear combinations of the entries of y and Y, so the lemma applies. The verifier works in two phases. In the first phase he uses decoding procedure P of the lemma to retrieve the four bits he needs for the naive test and in the second phase he performs the naive test. In this protocol the verifier performs the randomized retrieval of the four bits with four independent strings of random seeds.

If $\Psi \in QSAT$ the true prover follows the encoding instructions starting from a solution $\mathbf{y}$ of Ψ. By property 1 of Lemma 2.9 the verifier decodes the questioned entries from the code correctly never giving any error message, and the tests of the naive protocol also go through with probability 1, since $\mathbf{y}$ and Y were correctly given by the prover.

If $\Psi \notin QSAT$ then every proof $\mathbf{w}$ is cheating. Nevertheless $D(\mathbf{w}) = (\mathbf{y}, Y)$ exists, although the naive verifier rejects $(\mathbf{s}_\Psi M, N^T Y N, \mathbf{y}N)$ with probability at least 1/16. Recall that the naive verifier looks at four entries. Fixing any of those four tuples of entries that causes the verifier of the naive protocol to reject we compute the conditional property that the verifier of the correct protocol rejects this four tuple (decoding error in the first phase can result in wrongly accepting a four tuple). Let A_i (for $1 \leq i \leq 4$) be the event that the decoding procedure P either outputs "Reject" or it correctly decodes the ith member of the four tuple. By property 2 of Lemma 2.9 we have that $Prob(A_i) \geq \delta$ for $1 \leq i \leq 4$. Since A_i and A_j are independent for $1 \leq i < j \leq 4$, we have that $Prob(A_1 \cap A_2 \cap A_3 \cap A_4) \geq \delta^4$. Any event in $A_1 \cap A_2 \cap A_3 \cap A_4$ results in an error message either in the decoding phase, or — if all four bits are decoded correctly – in the naive protocol phase. Therefore the (unconditional) probability that the verifier rejects any $\mathbf{w}$ when $\Psi \notin QSAT$ is at least $\delta^4/16$. □

Lemma 2.10 *Let $c > 3$ be a constant and $f(n)$ be an arbitrary function from the positive integers to the positive integers. If there is a non-adaptive $(f(n), c)$-restricted verifier V which recognizes a language L then there is a non-adaptive $(f(n)+3c+6, 3)$-restricted verifier V' which recognizes L such that its checking criterion is always a disjunct of three literals made from the three bits he reads.*

Proof. We prove this lemma with a technique called *proof composition.* Proof composition is a general principle stating that if the prover gives an evidence that there exists a proof for $x \in L$ then this is a proof for $x \in L$. A very useful composition technique was introduced in [12], but here we need a rather trivial version of it.

Let us fix x. For every fixed random choice r the accepting criterion of V is a Boolean function $f_r(P) = V(x, P, r)$ of the c bits of P examined by V. The composed proof consists of the original proof, which we call the *core* and a small (constant size) proof β_r for every $r \in \{0,1\}^{f(n)}$. Each small proof is constructed as follows. Recall that every Boolean function $f(\alpha)$ of

c variables can be written in the form $(\exists\beta)\ g(\alpha,\beta)$, where g is a $3CNF$ formula of at most 2^{c+1} variables (including the variables in α), and at most 2^{3c+6} clauses. The variables in β are called the auxiliary variables. Let g_r be the $3CNF$ formula equivalent to f_r in the above manner. The small proof associated with $r \in \{0,1\}^{f(n)}$ is any β_r for which $g_r(\alpha_r,\beta_r) = 1$. Here α_r is the sequence of c bits V reads from P for random string r. Verifier V' picks a random r and a random clause of g_r and accepts if the later evaluates to 1. Since g_r has at most 2^{3c+6} clauses, the composed verifier needs at most $3c+6$ more random bits than the original one. If V accepts a proof P with probability 1 then for every r there exists a β_r such that $g_r(\alpha_r,\beta_r) = 1$, so there exists a composed proof which V' accepts with probability 1. On the other hand if V rejects every proof with probability at least ϵ (for input x), then no composed proof will be accepted with probability greater than $1-\frac{\epsilon}{2^{3c+6}}$ by V'. To see this consider a composed proof with core P. Since V rejects P with probability at least $1-\epsilon$, $f_r(\alpha_r) = 0$ for at least ϵ fraction of rs. For any such r no matter what β_r is we have $g_r(\alpha_r,\beta_r) = 0$. Then (conditioned on r) a false clause is found by V' with probability at least 2^{-3c-6}.□

3 Non-Approximability Results

NP optimization problems (NPO) (see [90] [53] [52]) have one of the following forms:

- Find $\max_{|y|\le q(|x|)} P(x,y)$ given input x. [Maximization]
- Find $\min_{|y|\le q(|x|)} P(x,y)$ given input x. [Minimization]

Here $P(x,y)$ is a polynomially computable function that assigns non-negative rationals to pairs of strings, and q is a polynomial. (In the literature witness y is selected from an arbitrary polynomial time computable set of strings not necessarily only of the form $\{y \mid |y| \le q(|x|)\}$. Our definition is sufficient to describe all relevant problems, and enables us to avoid anomalies that arise from the more general definition.)

While one can construct optimization problems that are not in NPO, the class plays a primary role in optimization theory with such well known problems as coloring, allocation, scheduling, steiner tree problems, TSP, linear and quadratic programming, knapsack, vertex cover etc. The consequences of the PCP theory almost solely concern NPO problems. There are exceptions [49, 50], but we do not discuss them here.

Most NPO problems are NP hard to compute exactly. A (polynomial time) algorithm A is said to *approximate* an NP maximization problem P to within a factor $r(x) \geq 1$, if A outputs a witness y such that the optimal solution for input x lies within $\max_{|y| \leq q(|x|)} P(x,y)/r(x)$ and $\max_{|y| \leq q(|x|)} P(x,y)$. Here $r(x)$ is called *approximation ratio,* and can be generalized also to minimization problems in an obvious way.

Below we give three examples to NPO problems:

MAX3SAT: Let x represent a 3CNF formula, y represent an assignment, and $P(x,y)$ be the number of clauses in x satisfied by y. [Maximization]

Setcover: Let x represent a polynomial size set system, y represent a selection of sets, and $P(x,y)$ be the function, which equals to the number of selected sets if the selected sets cover the universe, otherwise the size of the entire set system. [Minimization]

PCP: Let V be an $(\log n, g(n))$-restricted probabilistic verifier for a language L. x represents an input, y represents a proof, and $P(x,y)$ is the acceptance probability of proof y for input x. Since the verifier uses $\log n$ randomness, we can run V on all random strings and compute this probability in polynomial time. [Maximization]

If in the latest example we take L to be an NP complete language and V to be the verifier of Theorem 2.4, then it is easy to see that if we could efficiently approximate the acceptance probability of V to within a factor better than $1 - \epsilon$ (for the ϵ of the theorem), then we could also solve the membership problem for L. This is an example to an argument about non-approximability. While it is easy to define functions that cannot be approximated unless $P = NP$, the advantage of this example is that the value of $P(x,y)$ is proportional to a number of very simple predicates that y satisfies, namely each predicate depends only on three bits of y. This special type of NPO problem is called a *constraint satisfaction problem.* Before the theory of PCP there were no non-approximability results for any constraint satisfaction problem. The new result gave rise to an entirely new theory of non-approximability. We devote the next few sections to this theory.

3.1 A Brief History of Approximating NPO Problems

The NP completeness theory of Cook [51] and Levin [103] puts NPO problems into two categories: NP-hard or not NP-hard. The first alarm that this

characterization is too coarse was sent off in an early paper of D. Johnson [90] entitled "Approximation Algorithms for Combinatorial Problems."

Motivated by exact bounds on the performance of various bin packing heuristics of [71] Johnson gave algorithms for the Subset Sum, the Set Cover and the MAX k-SAT problems with guarantees on their performances $(1 + o(1), O(\log |S|), 2^k/(2^k - 1)$, respectively). He also gave non-approximability results, but unfortunately they referred only to specific algorithms. Nevertheless, he has brought up the issue of classifying NPO problems by the best approximation ratio achievable for them in polynomial time.

For years advances were very slow. Sahni and Golzales [126] proved the non-approximability of the non-metric traveling salesman and other problems, but their proofs were very similar to standard NP completeness proofs, and they effected only problems where approximation algorithms were not explicitly sought for. A more promising approach was found by Garey and Johnson [72] who used graph products to show that the chromatic number of a graph cannot be approximated to within a factor of $2 - \epsilon$ unless $P = NP$, and an approximation algorithm for the max clique within *some* constant factor could be turned into an algorithm which approximates max clique within *any* constant factor.

The old landscape of approximation theory of NPO radically changed when in 1991 Feige, Goldvasser, Lovász, Safra and Szegedy [62] for the first time used Babai, Fortnow and Lund's characterization of NEXP in terms of multi-prover interactive proof systems [22] to show that approximating the clique within any constant factor is hard to $\mathrm{NTIME}(n^{1/\log\log n})$. Simultaneously Papadimitriou and Yannakakis defined a subclass of NPO what they called MAXSNP, in which problems have an elegant logical description and can be approximated within a constant factor. They also showed that if MAX3SAT,vertex cover, MAXCUT, and some other problems in the class could be approximated with an arbitrary precision, then the same would hold for all problems in MAXSNP. They established this fact by reducing MAXSNP to these problems in an *approximation preserving* manner. Their original intention was most likely to build a theory of non-approximability via defining suitable reductions, analogous to those in the theory of NP, but new non-approximability results in [10] let them achieve much more. The theory of NP and the theory of non-approximability met, and research developed rapidly in three directions:

1. Non-approximability results for NPO problems;

2. Construction of approximation algorithms achieving optimal or near optimal ratios;

3. Development of approximation preserving reductions and discovery of new (non)-approximability classes.

Today we have a a precise idea about the non-approximability status of many NPO problems. The on-line compendium of Pierluigi Crescenzi and Viggo Kann [52] keeps track of the growing number of results and continuously updates data about most known NPO problems.

3.2 The gap-3SAT Instance

The gap-3SAT instance is a 3SAT formula, which is either satisfiable or at most $1-\epsilon$ of its clauses are satisfiable.

Theorem 3.1 ([10]) *There exists an $\epsilon > 0$ such that for every $L \in NP$ there is polynomial time computable map from Σ^* to 3SAT instances (mapping $x \in \Sigma^*$ to ϕ_x) such that:*

1. *If $x \in L$ then ϕ_x is satisfiable.*

2. *If $x \notin L$ then every assignment leaves at least ϵ clauses of ϕ_x unsatisfied.*

Indeed, build a PCP for L as in Theorem 2.4, and define

$$\phi_x = \wedge_r c_r,$$

where c_r is the verifier's accepting criterion for random string r, and which is by the assumption of Theorem 2.4 is a disjunct of three literals. The variables are the bits of the proof. If $x \in L$ then all c_is evaluate to 1, but if $x \notin L$ then ϵ fraction of the clauses must remain unsatisfied for every assignment, i.e. for every proof of the prover.

The theorem has the immediate consequence that *MAX3SAT* cannot be approximated in polynomial time to within a factor better than $1/(1-\epsilon)$, otherwise we could use this algorithm to efficiently compute if $x \in L$. Since MAX3SAT is MAX-SNP complete in the sense of Papadimitriou and Yannakakis [115], we get:

Theorem 3.2 ([10]) *No MAX-SNP complete problem can be efficiently approximated to within a factor arbitrarily close to 1 unless P=NP.*

In Theorem 3.2 the non-approximability gap $\epsilon > 0$ is not explicit. We make it explicit in Section 3.6 for various constraint satisfaction problems.

3.3 Refined Parameters of PCPs

In Section 2.1 we parameterized probabilistic verifiers with the number of random bits, $f(n)$, and the number of check bits $g(n)$ (See Definition 2.1). When we make a reduction from a PCP to an NPO problem, the exact non-approximability gap of the later may depend on several other parameters of the PCP. Here we list some, starting with the most important ones:

1. The *completeness probability,* $q(n)$, which is the least probability of acceptance, when $x \in L$. Originally $q(n) = 1$, but Håstad [84, 85] proved that we can improve on other parameters if we allow $q(n)$ to be slightly less than 1.

2. The *soundness probability,* $p(n)$, which is the greatest probability of acceptance, when $x \notin L$.

3. The *free bit complexity.* In Section **??** we defined $ACC_V(x, r)$. Using this notation:

$$free(n) = \max_{|x|=n} \log \max_{r} |ACC_V(x, r)|.$$

4. The *average free bit complexity,*

$$free_{av}(n) = \max_{|x|=n} \log \left(\frac{1}{2^{f(n)}} \sum_{r} |ACC_V(x, r)| \right).$$

   note that $\sum_r |ACC_V(x, r)|$ is the size of the FGLSS graph.

5. The alphabet size $|\Sigma|$. So far we have assumed that the witness $\mathbf{w}$ is a string over the binary alphabet $\{0, 1\}$. Raz and Safra [122] showed the relevance of using alphabets which may contain up to n symbols.

The parameters we have discussed so far were defined from the model of PCP. We also consider *derived parameters* that can be expressed in terms of the parameters we have defined so far.

1. The *gap* function $gap(n) = q(n)/p(n)$.
2. The *amortized query bit complexity* defined as $\frac{g(n)}{\log_2 gap(n)}$.
3. The amortized *free bit complexity* defined as $\frac{free(n)}{\log_2 gap(n)}$.

The following notations for PCP classes with refined parameters are more or less standard in the literature:

$PCP_{q,p}(f,g)$ is the class of languages for which there is an $(f(n^c), g(n^c))$-restricted PCP verifier with completeness probability $q(n^c)$ and soundness probability $q(p^c)$. $FPCP_{q,p}(f, free)$ denotes the class, where the query complexity is replaced with the free bit complexity. $\overline{FPCP}(f, \overline{free})$ is $\cup FPCP_{q,p}(f, free)$, where the union runs through for all choices of $q(n)$, $p(n)$ and $free(n)$ such that the amortized free bit complexity is at most $\overline{free}(n)$.

3.4 Amplification Techniques

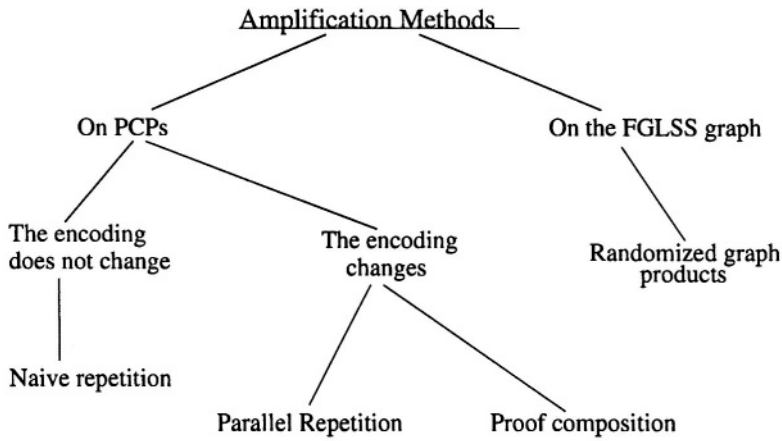


Figure 3: Various types of amplifications

Amplification techniques are transformations on Probabilistically checkable proofs or on the FGLSS graph that improve on their parameters. An ancestor of these techniques was used in [72] to prove that the chromatic number cannot be approximated to within a factor of $2 - \epsilon$. In this section we review the naive repetition, the parallel repetition, the proof composition

and the randomized graph product (see also in Figure 3). All of them are playing important roles in the theory of non-approximability.

3.4.1 Naive Repetition

In the case of naive repetition the new verifier's query is the conjunct of k independently chosen queries of the original verifier. It increases the gap in between the completeness and the soundness probabilities as shown in the table below:

	Original Verifier	Repeated Verifier
Number of Random Bits:	$f(n)$	$kf(n)$
Number of Check-bits:	$g(n)$	$kg(n)$
Completeness Probability:	$q(n)$	$q(n)^k$
Soundness Probability:	$p(n)$	$p(n)^k$

Perhaps the most troublesome aspect of the naive repetition is that it increases the number of random bits the verifier needs to use. Zuckermann [142] suggests a "less naive" repetition, where the verifier chooses its k tuple from a set S of polynomially many k-tuples. Unfortunately S is constructed randomly, so the *PCP* verifier cannot produce it by himself. *PCP* verifiers who work with advice S we call a PCP^S verifier. With high probability over a random S with $|S| = 2^{F(n)}$ the following holds for Zuckerman's Verifier:

	Original Verifier	Zuckerman's Verifier
Number of Random Bits:	$f(n)$	$F(n)$
Number of Check-bits:	$g(n)$	$g(n)(F(n)+2)$
Completeness Probability:	1	1
Soundness Probability:	$1/2$	$2^{f(n)-F(n)}$

Lars Engebretsen and Jonas Holmerin ([59], Lemma 14) have computed the performance of the Zuckerman's Verifier for various values of the completeness and soundness probabilities. This is what they got:

	Original Verifier	EH-Verifier
Number of Random Bits:	$f(n)$	$F(n)$
Number of Check-bits:	$g(n)$	$g(n)\frac{F(n)+2}{-\log p(n)} = g(n)D$
Completeness Probability:	$q(n)$	$q^D(n)/2$
Soundness Probability:	$p(n)$	$2^{f(n)-F(n)}$

In the case of perfect completeness the EH-Verifier also has perfect completeness.

3.4.2 Parallel Repetition

Parallel repetition of PCPs, as opposed to naive repetition requires to change not only the verifier's behavior, but also the encoding. Its usefulness later will be clear.

Assume that an (f, g)-restricted verifier V expects P to be segmented, i.e. P consists of blocks (usually of some fixed block size). The verifier can enforce this structural restriction even on a cheating prover, because it effects only the positions the verifier looks at and not the proof content. Assume that the verifier always accesses data only from two different blocks. A further (albeit non-essential) restriction is that P consists of two parts, P_1 and P_2, such that V always accesses one block from each. A PCP with this structural restriction is called a *two prover system,* since P is being thought to be given by two different provers whom the verifier confronts with one another.

Any naive repetition for any $k > 1$ would upset this structure, but parallel repetition keeps the two prover property of the protocol. We shall denote a verifier V repeated k times in parallel by V^k. The true prover for V^k writes down all possible ordered k tuples of segments from P_1 and all possible ordered k tuples of segments from P_2, and these k tuples are the segments of the repeated provers. The verifier uses $kf(n)$ randomness to simulate k independent checks $(c_{1,1}, c_{2,1}), \ldots (c_{1,k}, c_{2,k})$ of V, and queries segments $(c_{1,1}, \ldots, c_{1,k})$ and $(c_{1,1}, \ldots, c_{1,k})$. V^k accepts iff V would accept all pairs $(c_{1,i}, c_{2,i})$ for $1 \leq i \leq k$.

It is easy to compute the acceptance probabilities if the proof has the P_1^k, P_2^k structure, but there is no guarantee that cheating provers adhere to this structure. Surprisingly we can still say something about the maximum acceptance probability for V^k.

Theorem 3.3 (Raz's Parallel Repetition Theorem [121]) *For every verifier V of a 2-prover PCP there is a constant $0 \leq c \leq 1$:*

$$\max_{P'} \mathrm{Prob}_{r_1,\ldots,r_k}(V^k(x, P', r_1, \ldots, r_k) = 1) \leq c^k, \tag{5}$$

where c depends only on $p(V, x) = \max_P \mathrm{Prob}_r(V(x, P, r) = 1)$, and the maximum segment size for V, and is strictly less than 1 if $p(V, x) < 1$. (In Equation (5) P' ranges through all (possibly cheating) proofs for V^k.)

The properties of the repeated proof are summarized in the table below:

	Original Verifier	With k repetitions
Number of Random Bits:	$f(n)$	$kf(n)$
Number of Check-bits:	$g(n)$	$kg(n)$
Completeness Probability:	$q(n)$	$q(n)^k$
Soundness Probability:	$p(n)$	c^k (See Theorem 3.3)

The price we pay for keeping the two prover structure is a weaker bound on the soundness probability than in the case of naive repetition.

Two prover systems were introduced by Ben-Or, Goldwasser, Kilian and Wigderson [35], have been studied by Rompel and Sipser [66], Feige and L. Lovász[65], and by many others. In modern PCP constructions we need a fairly simple two prover protocol, considered folklore. The verifier of this protocol, which we call V_{3SAT}, comes from the gap-3SAT instance of Section 3.2 and works as below.

Let ϕ_x (associated with $x \in \Sigma^*$) be a 3SAT instance, which is either completely satisfiable or at most $1-\epsilon$ fraction of its clauses are satisfiable for some $\epsilon > 0$. By Theorem 3.1 we can assume that the language $L = \{x \mid \phi_x \text{ is satisfiable}\}$ is NP complete, and that ϕ_x is polynomial time computable from x. The verifier V_{3SAT} computes ϕ_x, picks a random clause in it, and asks the first prover to evaluate the three literals in the clause. If all the three literals evaluate to false, V_{3SAT} rejects outright. Otherwise he picks a random variable in the clause, and asks the second prover to evaluate this variable (the two prover do not hear each other's answers, but they can agree in a strategy in advance, and they also know x). If the two provers evaluate the variable differently, the verifier rejects, otherwise accepts.

In order to see that the above informal description indeed defines a two prover system in our former sense, identify P_i with the concatenation of all answers to all possible questions of the verifier to prover i. Each answer is one segment, and the number of segments in P_i equals to the number of different questions the verifier may ask from prover i. The segment size is equal to the answer size.

Next we show that the above protocol has a gap at least $\epsilon/3$. Clearly, when $x \in L$, i.e. when ϕ_x is satisfiable, the provers just agree in a satisfying assignment to the variables of ϕ_x, and V_{3SAT} accepts with probability 1. Otherwise every assignment fails to satisfy at least ϵ fraction of the clauses of ϕ_x. Observe that the strategy of P_2 corresponds to an assignment to the variables of ϕ_x. For the sake of simplicity we call this assignment P_2. This assignment must fail to satisfy at least ϵ fraction of the clauses of ϕ_x, and the verifier has probability ϵ that he finds one of these clauses. The first

prover of course may lie, and may not give the same evaluation to one of the variables in that clause as P_2, but (conditioned on the clause) the verifier notices this inconsistency with probability at least 1/3. Thus in case $x \notin L$, V_{3SAT} accepts with probability at most $1 - \epsilon/3$.

Feige [61] shows that we may also assume that in ϕ_x each variable appears exactly in 5 clauses and that each clause contains exactly 3 variables. By Feige's restriction we may assume that V_{3SAT} first selects a random variable and then chooses a clause randomly out of the five that contains this variable. Observe that the sequence in which the verifier asks the provers does not matter, and in the sequel we all the prover who provides the clauses the second prover. We may also assume that the verifier decides at his output after he has heard both provers.

In all applications V_{3SAT} is repeated in parallel ℓ times, where ℓ depends on the application and ranges from a large constant to $\log n$. The lemma below summarizes the properties of V^{ℓ}_{3SAT}.

Lemma 3.4 *Let ℓ be an arbitrary polynomially bounded function of $|x|$, and let $b = \{b_1, \ldots, b_N\}$ be the Boolean variables, and $c = \{c_1, \ldots, c_{5N/3}\}$ be the clauses of ϕ_x. (We say $b_i \in c_j$ if b_i or $\overline{b_i}$ appears in c_j.) Then V^{ℓ}_{3SAT} has the following features:*

1. *It randomly picks an ℓ tuple $U = (b_{i_1}, \ldots, b_{i_\ell})$ and checks it against an ℓ tuple $W = (c_{j_1}, \ldots, c_{j_\ell})$ such that $b_{i_\nu} \in c_{j_\nu}$ for $1 \leq \nu \leq \ell$. W is called a* companion *of U, and it is randomly picked from set of all companions of U. The verifier accepts if V_{3SAT} would accept (b_{i_ν}, c_{j_ν}) for every $1 \leq \nu \leq \ell$, i.e. all c_{j_ν} are satisfied, and the assignments of the variables in U and W are consistent.*

2. *It has perfect completeness.*

3. *It has soundness at most c^{ℓ} for some fixed constant $c < 1$.*

4. *It uses $O(\ell \log n)$ random bits and 4ℓ check bits. (Here parameter n is the length of x, and it is in polynomial relation with N.)*

3.4.3 Proof Composition

The idea of composing probabilistic verifiers first appears in the paper of Arora and Safra [12] and is aimed at improving on the number of check-bits of a probabilistic proof system. We implicitly use this idea in Section 5.3.2,

where we compose holographic codes and explicitly use a baby version of it in Lemma 2.10.

To describe proof composition in general let us assume we have a proof system with an (f, g)-restricted verifier V that recognizes a language L. From now on we call V the *outer verifier.* We would like to construct a new verifier V' which recognizes the exact same language as V, but uses less check-bits. Fix input x with $|x| = n$. The composed proof for $x \in L$ consists of the old proof P, which we call the *core,* and a proof P_r for every choice of the random string r with $|r| = f(n)$ which we describe later. We shall compose proof systems only with non-adaptive verifiers. Let Q_r be the set of bits that V reads from P when works with random string r. The set of accepting views $ACC_V(x, r)$ can be viewed as a language L_r restricted word-length $|Q_r|$. Let V_r be an (f', g')-restricted probabilistic verifier that recognizes this language, where $f'(n), g'(n)$ are some functions independent of r. The set of all V_rs for $|r| = g(n)$ is called the inner verifier (somewhat inaccurately).

The compound verifier picks a random r with $|r| = g(n)$ and a random string r' with $|r'| = f'(|Q_r|)$, runs V_r on input Q_r, random string r' and proof Q_r, and outputs its output.

Let us compute the parameters of the composed proof system. Assume that V has completeness probability $q(n)$ and soundness probability $p(n)$, and that each $V(r)$ has completeness probability $p'(n)$ and soundness probability $q'(n)$. Here is what we get:

	Original Proof	Composed Proof
Completeness Probability:	$q(n)$	$q(n)q'(n)$
Soundness Probability:	$p(n)$	$(1 - p(n))p'(n) + p(n)$
Number of Check-bits:	$g(n)$	$g(n) + g'(g(n))$

As we see, all parameters get worse. What is our gain then? The inner verifier uses $g(n) + g'(g(n))$ check bits because for some r he reads $g(n)$ bits from the core and $g'(g(n))$ bits from P_r. The second term is small, but we need to improve on the first term. Recall that the bits V_r reads from the core constitute the input of V_r, and for normal probabilistic verifiers there is no restriction on how many bits they read from their input.

Babai, Fortnow, Levin and Szegedy [23] have introduced a probabilistic verifier, which reads only a few bits from its input. The price of this restriction is the BFLS verifier cannot recognize every language. In fact it cannot recognize any language, only promise problems, or, as Szegedy [133]

interprets it languages over a three valued logic. We can still use the BFLS verifier as an inner verifier if the outer satisfies a *segmentation restriction,* as in Section 3.4.2, because of theorems of the following type:

"For every language L there is an encoding E, a decoding D and a BFLS verifier V^{BFLS} which reads at most $e(n)$ bits of its input. If the input to V^{BFLS} decodes to $x \notin L$ then V^{BFLS} accepts with probability at most p', and for every input of the form $E(x)$ with $x \in L$ the verifier accepts with probability at least q'. If words of length n are defined in k segments of length n/k each, then the encoding and decoding can be performed on the segments separately."

Here functions q', p', e and k depend on the particular BFLS type theorem but the first three parameters deteriorate as the fourth parameter grows. If the outer verifier's proof consists of disjoint data segments and the outer verifier reads at most k segments from its proof (this is the case for the two prover protocol of Section 3.4.2 with $k = 2$), then we can compose it with a BFLS verifier as above and get

	Original Proof	Composed Proof
Number of Check-bits:	$g(n)$	$e(g(n)) + g'(g(n))$

This technique was responsible for the great improvements in the number of check-bits in [12] and [10]. (Our explanation of Theorem 2.4 in this chapter circumvents the above sketched proof composition technique.) A technical detail is that the composed verifier expects the segments of the outer verifier to be encoded according to the code E of the BFLS inner verifier, so the core is not exactly the original proof P.

3.4.4 Randomized Graph Products

Graph products amplify gaps in sizes of independent sets [74] and and chromatic numbers [104]. There are many different types of graph products, but the one which is most useful for us is the *inclusive graph product.*

Definition 3.5 (Inclusive Graph Product) *Let G_1 and G_2 be a graph with vertex sets V_1, resp V_2, and edge sets E_1, resp. E_2. We define the inclusive graph product of $G_1 \times G_2$ on the vertex set $V_1 \times V_2$ such that (x_1, x_2) is connected with (y_1, y_2) if and only if x_1 is connected with y_1 in G_1 or x_2 is connected with y_2 in G_2*

It turns out that inclusive graph products behave particularly nicely with respect to the independent set sizes. Since the size of the maximum clique of the graph is the size of the maximum independent set size in the complement of the graph, if we want to amplify the gap in the clique size we first take the complement of the graph. In order to get amplification for the chromatic number we need to take a larger detour and first define the fractional chromatic number of a graph:

Definition 3.6 (Fractional Chromatic Number) *Let G be an undirected graph and I be a set of G. The indicator function X_I of I is map from V(G) to the reals which assigns 1 to the elements of I and 0 to the elements of V(G) \ I. Let Ind(G) be the collection of all independent sets of G. We define the fractional chromatic number of G by:*

$$\chi_f(G) = \min \sum_{I \in Ind(G)} \lambda_I$$

Subject to:

$$\lambda_I \geq 0 \text{ for all } I \in Ind(I);$$

$$\sum_{I \in Ind(G)} \lambda_I X_I \geq X_{V(G)} \text{ coordinate-wise.}$$

Clearly,

$$\chi(G) \geq \chi_f(G) \geq \max\left(\frac{V(G)}{\alpha(G)}, \omega(G)\right), \tag{6}$$

where $\alpha(G) = \omega(\overline{G})$ is the independence number of G.
Let G^k be the k-wise inclusive graph product of G with itself. Then

$$\alpha(G^k) = \alpha(G)^k \tag{7}$$

$$\chi(G^k) \leq \chi(G)^k \tag{8}$$

$$\chi_f(G^k) = \chi_f(G)^k \tag{9}$$

The Equation (9) is observed by Lovász [106]. We can amplify constant gaps in the independence number and the fractional chromatic number to polynomial by setting $k = c \log n$, but the size of the resulting graph becomes super-polynomial. In order to overcome this problem Berman and Schnitger [37] developed a randomized version of this graph product. Instead of taking G^k they randomly select an induced subgraph of it. Feige and Kilian [FeKi2] summarize the parameters of their construction in a lemma, which we slightly extended here:

Lemma 3.7 *Vertex induced subgraphs G' of G^k have the following properties:*

1. $\chi_f(G') \leq \chi_f^k(G)$
2. *If $\alpha(G) \leq C$ then $\alpha(G') \leq k|V(G)|$ for nearly all subgraphs G' with $\frac{|V(G)|^k}{C^k}$ vertices.*
3. $\chi_f(G') \geq \frac{|V(G')|}{k|V(G)|}$, *for all G' satisfying $\alpha(G') \leq k|V(G)|$.*
4. $\alpha(G') \geq \frac{|V(G')|}{c^k}$ *with high probability over a fixed sized subgraph G', where $c = |V(G)|/\alpha(G)$.*

3.5 The Long Code

The long code was invented by Bellare, Goldreich and Sudan [31], and it seems to be a necessary tool to prove exact non-approximability bounds for various constraint satisfaction problems. It is also used to prove the non-approximability of MaxClique to within a factor of $|V(G)|^{1-\epsilon}$ in [86].

The coordinates of the long code correspond to all possible functions $f : \{0,1\}^n \to \{0,1\}$. When we encode an $\mathbf{x} \in \{0,1\}^n$, the coordinate corresponding to f takes the value $f(\mathbf{x})$. Since there are 2^{2^n} Boolean functions on n inputs, the long code is a string of length 2^{2^n}.

One might ask: why do we want to use such a wasteful encoding in our constructions? From the long code we can very efficiently decode the value of *any* Boolean function on $\mathbf{x}$. We may think of the encoding with the long code as the composition of two encodings: first we encode $\mathbf{x} \in \{0,1\}^n$ to a string $\mathbf{x}_u \in \{0,1\}^{2^n}$, where $\mathbf{x}_u$ is one only at a single position, and zero everywhere else. Then we encode $\mathbf{x}_u$ with the Hadamard encoding i.e. with the one that contains the modulo two sum of the digits of $\mathbf{x}_u$ for every subset of the index set of $\mathbf{x}_u$.

$\mathbf{x}$		$\mathbf{x}_u$	Hadamard	
00	$\to$	0001	$\to$	0000000011111111
01	$\to$	0010	$\to$	0000111100001111
10	$\to$	0100	$\to$	0011001100110011
11	$\to$	1000	$\to$	0101010101010101

If we have a set of functions $h_1, \ldots, h_k$, and we know that $h_1(\mathbf{x}) = \ldots = h_k(\mathbf{x}) = 1$, then we can *condition* the long code only on those $\mathbf{x}$s that satisfy these conditions. In practice this is done by discarding those coordinates of

$\mathbf{x}_u$ that are indexed with such $\mathbf{x}$s for which the desired conditions do not hold. The Hadamard code is constructed for this shorter version of $\mathbf{x}_u$. This conditioning technique is due to Håstad [86], and it is a modification of an earlier technique of Bellare et. al. [31], called *folding*.

In this section we describe only the basic test to check the long code, which is the mother of all other tests. This test is invented by Håstad [85]. Checking the long code results in the verifier's rejection of a string which is "unrelated" to the long code. The relation we are looking for is described in Lemma 3.8. Below $\mathcal{F}$ denotes the set of all Boolean functions on n variables, and it is the index set for the table A we want to check. We denote by A_f the entry of A at index f. Note that $A_f \in \{0,1\}$, although some literature uses $\{1, -1\}$ instead of $\{0,1\}$.

Long Code Check [85]:

1. Choose f_0 and f_1 from $\mathcal{F}$ with uniform probability.
2. Choose a function $\mu \in \mathcal{F}$ by setting $\mu(\mathbf{x}) = 1$ with probability $1 - \epsilon$ and $\mu(\mathbf{x}) = 0$ otherwise, independently for every $\mathbf{x} \in \{0,1\}^n$.
3. Set $f_2 = f_0 \oplus f_1 \oplus \mu$, i.e. define f_2 for each $\mathbf{x} \in \{0,1\}^n$ by $f_2(\mathbf{x}) = f_0(\mathbf{x}) \oplus f_1(\mathbf{x}) \oplus \mu(\mathbf{x})$.
4. Accept if $A_{f_0} \oplus A_{f_1} \oplus A_{f_2} = 0$.

Lemma 3.8 *If A is the table of a word in the long code, the verifier accepts with probability at least $1 - \epsilon$. If A passes the test with probability at least $\frac{1+p}{2}$, then there is a B which is the modulo two sum of at most $\epsilon^{-1} \log 1/p$ words of the long code, and A and B agree in at least $\frac{1+p}{2}$ fraction of the 2^{2^n} bit positions.*

Because of the wasteful nature of the long code we use it only to encode very short (usually constant) piece of information in the context of proof recursion.

3.6 Constraint Satisfaction Problems

A general constraint satisfaction problem instance is a set of Boolean functions $S = \{f_i\}_{1 \le i \le m}$ on n variables, and the goal is to find an assignment to the variables such that the maximum number of f_is are satisfied. Specific constraint satisfaction problems have different restrictions on the types of

Boolean functions that S is allowed to contain. Here we discuss MAXkLIN, MAXkSAT and MAXkCSP. If k is a fixed constant then all of these problems are in MAXSNP and therefore approximable within some contant factor.

Problem Name: MAXkLIN:

Instance: A set of linear equations over $\mathbf{F}_2$. $L = \{x_{i,1} \oplus x_{i,2} \oplus x_{i,3} = c_i \mid 1 \leq i \leq m\}$ such that for $1 \leq i \leq m$, $1 \leq j \leq 3$: $x_{i,j} \in \{x_i\}_{1 \leq i \leq n}$, where x_i $(1 \leq i \leq n)$ are variables in $\mathbf{F}_2$, and c_i $(1 \leq i \leq m)$ are constants in $\mathbf{F}_2$.

Solution: An assignment to the variables x_i $(1 \leq i \leq n)$.

Objective: Maximize the number of equations set true in L.

It is obvious, using a random replacement, that for any $k \geq 1$ and any L at least half of the equations of L are satisfiable.

On the other hand Hastad constructs a PCP protocol [86] which shows:

Theorem 3.9 ([86]) *For any $\epsilon > 0$, $k \geq 3$ it is hard to approximate MAXkLIN to within a factor of $2 - \epsilon$.*

Håstad's protocol [85] briefly works as follows: We start with the V^{ℓ}_{3SAT} verifier of Lemma 3.4. By the result of Raz [121], this verifier has soundness probability c_1^{ℓ} for some $0 < c_1 < 1$, which can be made arbitrarily small if ℓ is a large enough constant. We then use the long code to encode every proof segment of this parallelized two prover PCP. The long code of the second prover's answer must be conditioned on the assumption that the corresponding clauses are satisfied. The new verifier works like V^{ℓ}_{3SAT}, except that the verifier checks the long code corresponding to each prover's answer, and the consistency in between the codes. This seems like a lot of checking, but Håstad does it very economically, only with three query bits. His check is a variation on the check described in Section 3.5. As a result he gets a verifier, which accepts with probability $1 - \delta$ if the formula of the gap-3SAT instance is satisfiable, and rejects with probability at least $\frac{1-\delta}{2}$ if it is not satisfiable, where δ can be made arbitrarily small. Moreover each check is a linear equation over $\mathbf{F}_2$ involving three variables. From this Theorem 3.9 follows.

Problem Name: MAXkSAT:

Instance: A set of disjuncts $\Phi = \{(t_{i,1} \vee t_{i,2} \vee t_{i,3}) \mid 1 \leq i \leq m\}$ such that for $1 \leq i \leq m$, $1 \leq j \leq 3$ $t_{i,j} \in \{x_1, \overline{x_1}, \ldots, x_n, \overline{x_n}\}$, where x_i $1 \leq i \leq n$ are Boolean variables.

Solution: A Boolean assignment to the variables x_i $1 \leq i \leq n$.

Objective: Maximize the number of disjuncts set true in Φ.

Johnson [90] in 1974 showed that the MAXkSAT is approximable to within a factor of $\frac{1}{1-2^{-k}}$ in polynomial time as long as each clause contains exactly k literals. Trevisan, Sorkin, Sudan, and Williamson [139] show that MAX3SAT is approximable within 1.249 without any restriction, and Karloff and Zwick [92] shows that it is approximable within a factor of 8/7 for satisfiable instances. Here we prove the non-approximability of MAX3SAT.

Theorem 3.10 ([86]) *For any $\epsilon > 0$ it is hard to approximate MAX3SAT to within a factor of $8/7 - \epsilon$.*

Proof. We reduce the problem to the non-approximability of MAX3*LIN*. Let L be an instance of MAX3*LIN* such that either at least $1 - \epsilon_1$ fraction of its equations are satisfied or at most $1/2 + \epsilon_2/2$ fraction of its equations are satisfied. We replace every equation $x \oplus y \oplus z$ of L with a set of four clauses $(x \wedge y \wedge \overline{z})$, $(x \wedge \overline{y} \wedge z)$, $(\overline{x} \wedge y \wedge z)$, and $(\overline{x} \wedge \overline{y} \wedge \overline{z})$. An assignment that satisfies the linear equation satisfies all the clauses, while an assignment that does not satisfies the linear equation satisfies three of the four clauses. The MAX3SAT instance Φ we construct for L is the multi-set union of these replacements. It is easy to see that either $1 - \epsilon_1$ fraction of the clauses of Φ are satisfied or at most $7/8(1 + \epsilon_2)$, and if ϵ_1 and ϵ_2 are small enough, we get the result. □

Actually, Håstad proves in general that MAXkSAT is not approximable within $\frac{1}{1-2^{-k}} - \epsilon$ for any k and ϵ, even if every clause consists of exactly k literals.

Problem Name: MAXkCSP:

Instance: A set $S = \{f_i \mid 1 \leq i \leq n\}$ of Boolean expressions (constraints), each depending on at most k of the Boolean variables x_i $1 \leq i \leq k$.

Solution: A Boolean assignment to the variables x_i $(1 \leq i \leq n)$.

Objective: Maximize the number of expressions set true in *S*.

The best known algorithm for the MAXkCSP problem has an approximation ratio 2^{k-1} [138]. Recently Samorodnitsky and Trevisan [130] showed that for any $\delta > 0$ and any NP complete language L there is a PCP with amortized query complexity $1+\delta$. It is easy to see that this implies that as k tends to infinity MAXkCSP is hard to approximate to within a factor of $2^{o(k)-k}$. In fact what they prove is slightly stronger:

Theorem 3.11 ([130]) *For every k the MAXkCSP problem is NP hard to approximate to within $2^{k-O(\sqrt{k})}$.*

Their proof relies on a very efficient version of testing the Hadamard code. We describe the standard way of checking the Hadarmard code in Section 5.2.1. Samorodnitsky and Trevisan recicle query bits in a parallel version of this test, and give a very clever analysis of their modified test.

3.7 The Max Clique

Instance: Undirected graph $G = (V, E)$.

Solution: A clique in G, i.e. a subset $C \subseteq V$ such that all two-element pairs in C are in E.

Objective: $\omega(G) = \max_C |C|$.

Boppana and Halldórsson [43] in 1992 proved that the maximum clique problem can be approximated within a factor of $O(|V|/(\log|V|)^2)$. The hope for a significantly more efficient algorithm has gradually faded away as a sequence of non-approximability results arose. The first such result was that of Feige, Goldwasser, Lovász, Safra and Szegedy [62] who established the connection in between clique approximation and proof checking. With the terminology developed in [12] (see also Definition 2.2 and Section 3.3) this connection can be formulated as below:

Theorem 3.12 (FGLSS) *Let $NP \subseteq PCP_{q,p}(f,g)$. Then if MaxClique of a graph of size $h(n) = 2^{f(n)+g(n)}$ can be approximated within a factor better than $q(n)/p(n)$ in time polynomial in $h(n)$, then $NP \subseteq DTIME(poly(h(n)))$.*

Proof. Let L be an NP complete language. Let $V(r, P, x)$ be a probabilistic verifier that recognizes L using $f(n)$ query bits, $g(n)$ random bits, and which accepts for at least $q(n)2^{f(n)}$ choices of r for *some* P, if $x \in L$, and accepts for at most $p(n)2^{f(n)}$ choices of r for *every* P, if $x \notin L$.

Consider the FGLSS graph, $G_V(x)$, for V and x defined in Section **??.** Conditions 1.-3. of Lemma **??** imply the theorem, since if we could approximate the max clique of $G_V(x)$ within a factor better than $q(n)/p(n)$ in time $poly(h(n))$ then we would be able to use this algorithm to tell whether $x \in L$ or not by declaring that $x \notin L$ iff the estimate the algorithm gives for the maximum clique size of $G_V(x)$ is at most $p(n)2^{f(n)}$ (note that by definition the algorithm always gives a lower estimate on the clique size). □

Let $f(n) = \log n \log\log n$. Feige et al. show that $NP \subseteq PCP_{1,1/2}(f, f)$. This argument used the scaled down version of the two prover interactive protocol by Babai Fortnow and Lund [22], and provided the first, albeit not very refined PCP construction. If we apply $\log^k n$ independent naive repetitions on the above verifier (see Section 3.4.1) we obtain $NP \subseteq PCP_{1,1/2^{\log^k n}}(f(n)\log^k n, f(n)\log^k n)$. By Theorem 3.12 this implies that for every fixed $\epsilon > 0$ MaxClique cannot be approximated within a factor of $2^{\log^{1-\epsilon}|V|}$ unless NP has quasi-polynomial time algorithms.

The next important important step was made by Arora and Safra [12] who showed the NP hardness of MaxClique up to an approximation factor $2^{\sqrt{\log|V|}}$. This result was the first to show the non-approximability of Max Clique under the natural $P \neq NP$ assumption, and provided the first PCP construction which characterized NP.

Arora, Lund, Motwani, Sudan and Szegedy [10] proved that $NP = PCP_{1,1/2}(\log n, O(1))$ (see Theorem 2.4). An amplification method [47] based on the "expander walk" technique of Ajtai, Komlós and Szemeredi [1] yields that $NP = PCP_{1,1/n}(\log n, \log n)$. Then Theorem 3.12 gives that there is an ϵ that it is NP-hard to approximate *MaxClique* up to a factor of $|V|^{\epsilon}$. Alon, Feige, Wigderson and Zuckerman [2] showed how to obtain the same result starting with FGLSS graph constructed from Theorem 2.4 by de-randomizing the graph product of Berman and Schnitger [37].

The constant ϵ was made explicit by Bellare, Goldwasser, Lund and Russel [32]. They also slightly improved on the proof methods of [10], and coupled it with a new amplification technique of [37, 142] to show that MaxClique is hard to approximate within a factor of $|V|^{1/25}$ unless NP is contained in $co - R\tilde{P}$. Here $\tilde{P}$ stands for the quasi polynomial time.

Notice that in the last result the hardness condition is different than before, because it involves the randomized class, $co - R\tilde{P}$. BGLR replace the condition $NP \subseteq PCP_{q(n),p(n)}(f(n), g(n))$ of Theorem 3.12 with $NP \leq_R PCP_{q(n),p(n)}(f(n), g(n))$, where $\leq_R$ stands for "randomly reducible." Although this weakens the hardness condition, it allows better amplification methods and thus better parameters.

Feige and Kilian [64] have observed that Theorem 3.12 can be replaced with the following lemma (for the definitions see Section 3.3):

Lemma 3.13 (Feige, Kilian [64]) *Let $NP \subseteq FPCP_{q,p}(f, free)$. Then if MaxClique of a graph of size $h(n) = 2^{f(n)+free(n)}$ can be approximated within a factor better than $q(n)/p(n)$ in time polynomial in $h(n)$, then $NP \subseteq DTIME(poly(h(n)))$.*

From the above lemma they showed that Max Clique cannot be approximated to within a factor of $|V|^{1/15-\epsilon}$ unless *NP = coRP*. Bellare and Sudan in [34] introduced the amortized free bit complexity, $\overline{free}$ (see Section 3.3), and notice that the best available amplification techniques give that:

Lemma 3.14 *If there is a PCP which uses $O(\log n)$ randomness and has amortized free bit complexity $\overline{free}$, then Max Clique cannot be approximated within a factor better than $n^{\frac{1}{1+\overline{free}}}$ unless $NP \subseteq coR\tilde{P}$.*

They prove that MaxClique cannot be approximated to within a factor better than $|V|^{1/4}$ unless $NP \subseteq coR\tilde{P}$. The next major step was made by Bellare, Goldreich and Sudan [31], who discovered the long code (see Section 3.5), and building it into their PCP immediately could reduce the free bit complexity. They prove the non-approximability of MaxClique to within a factor better than $|V|^{1/3}$ unless $NP \subseteq coRP$.

Although the constant in the exponent gradually improved, it seemed that theoretical limitations bar the improvement of the non-approximability exponent beyond 1/2. It has turned out, that the limitations could be overcome by dropping unnecessary assumptions about the PCP, such as the perfect completeness property. Relying on a new test for the long code of Bellare et. al., Håstad was able to show:

Theorem 3.15 (Håstad [85]) *Max Clique cannot be approximated to within a factor of $|V|^{1-\epsilon}$ for any $\epsilon > 0$ unless $NP \subseteq ZPP$.*

In the proof of this theorem Håstad uses the same probabilistically checkable proof construction as he does for constraint satisfaction problems, but the verifier works differently, and the analysis is different. He is able to achieve amortized free bit complexity ϵ for any $\epsilon > 0$.

Recently Sudan and Trevisan [132] and Samorodnitsky and Trevisan [130] simplified the involved analysis of Håstad, and constructed a PCP which has the additional benefit of having amortized query bit complexity $1 + \epsilon$. Building on [132] and [130] Lars Engebretsen and Jonas Holmerin [59] showed very recently, that unless $NP \subseteq ZPTIME(2^{O(\log n(\log\log n)^{3/2})})$, Max Clique cannot be approximated to within a factor of $|V|^{1-O(\sqrt{\log\log n})}$.

Arora=A, Babai=B, Bellare = Be, Feige=Fe, Goldwasser=G, Goldreich = Go, Håstad =H, Kilian=K, Lovász=Lo, Lund=Lu, Motwani=M, Russel = R, Safra=Sa, Sudan=Su, Szegedy=Sz		
Due to	Factor	Assumption
FeGLoSaSz	$2^{\log^{1-\epsilon}\|V\|}$ for any $\epsilon > 0$	$NP \not\subseteq \tilde{P}$
ASa	$2^{\sqrt{\log\|V\|}}$	$NP \neq P$
ALuMSuSz	$\|V\|^{\epsilon}$ for some ϵ	$NP \neq P$
BeGLuR	$\|V\|^{1/30}$	$NP \neq coRP$
BeGLuR	$\|V\|^{1/25}$	$NP \not\subseteq co\tilde{RP}$
FeK	$\|V\|^{1/15}$	$NP \neq coRP$
BeSu	$\|V\|^{1/6}$	$NP \neq P$
BeSu	$\|V\|^{1/4}$	$NP \not\subseteq co\tilde{RP}$
BeGoSu	$\|V\|^{1/4}$	$NP \neq P$
BeGoSu	$\|V\|^{1/3}$	$NP \neq coRP$
H	$\|V\|^{1-\epsilon}$ for any $\epsilon > 0$	$NP \not\subseteq ZPP$

3.8 The Chromatic Number

Instance: Undirected graph $G = (V, E)$.

Solution: A coloring of G, i.e. a function $C: V(G) \to [1, k]$ such that for every $(a, b) \in E(G) : C(a) \neq C(b)$.

Objective: $\chi(G) = \min_C k$.

The best known lower bound for approximating the chromatic number was obtained by Halldórsson in 1993 [83]. Given a graph G, he finds a

coloring of G in polynomial time with at most

$$\chi(G)\frac{O(|V(G)|\log^2\log|V(G)|)}{\log^3|V(G)|}$$

colors. From the other direction, an early result of Garey and Johnson [72] shows that $\chi(G)$ is NP hard to approximate to within any constant less than 2. In 1992 Lund and Yannakakis found a reduction from approximating the MaxClique of the FGLSS graph of the ALMSS verifier to the problem of approximating the chromatic number.

Theorem 3.16 (Lund and Yannakakis, [108]) *There is an ϵ such that it is NP hard to approximate $\chi(G)$ within a factor of $|V(G)|^{\epsilon}$.*

The proof of Lund and Yannakakis worked in three steps.
1. Construct the FGLSS graph $G_V(x)$ for the ALMSS verifier V of an NP-complete language. Observe that $\overline{G}_V(x)$ has bounded degree, and

1. If $x \in L$ then $\alpha(\overline{G}_V(x)) = 2^r$
2. If $x \notin L$ then $\alpha(\overline{G}_V(x)) \leq 2^r(1-\epsilon)$

2. Use the randomized graph products of Berman and Schnitger [37], and Blum [B] to obtain a graph G_1 from $\overline{G}_V(x)$, which has maximum independence number n^{ϵ_1} when $x \in L$, and maximum independence number n^{ϵ_2} when $x \notin L$ for some $\epsilon_1 > \epsilon_2 > 0$.
3. Apply another transformation which turns G_1 into a graph G_2 such that $\chi(G_2) \geq n^{\epsilon_3}$ if $x \notin L$ and $\chi(G_2) \leq n^{\epsilon_4}$ if $x \in L$ for some $\epsilon_3 > \epsilon_4 > 0$.

Improving upon this result and its subsequent sharpenings [97, 34], Fürer [70] gives a randomized reduction which shows that if $\omega(G)$ cannot be approximated to within $|V(G)|^{\frac{1}{f+1}}$ then $\chi(G)$ cannot be approximated to within $|V(G)|^{\delta}$, where $\delta = \min\left(\frac{1}{2}, \frac{1}{2f+1}\right) - o(1)$. He also proposes a simple reduction which shows that it is hard to approximate $\chi(G)$ to within a factor of $|V(G)|^{\epsilon}$ for some $\epsilon > 0$. These reductions can be applied whenever a non-approximability result for the clique number of the FGLSS graph of a verifier for an NP complete language is known. Feige and Kilian [FeKi2] took a different route and building on the MaxClique result of Håstad [85] show:

Theorem 3.17 (Feige and Kilian [FeKi2]) *Unless $NP \subseteq ZPP$ it is intractable to approximate $\chi(G)$ to within $|V(G)|^{1-\epsilon}$ for any constant $\epsilon > 0$.*

Here ZPP denotes the class of languages that are solvable with a randomized algorithm that makes no error, and on expectation runs in polynomial time.

The proof of this theorem shows more, namely that it is NP-hard (under randomized reductions) to distinguish between graphs with $\alpha(G) \leq |V(G)|^{\epsilon}$ and graphs with $\chi(G) \leq |V(G)|^{\epsilon}$. To understand their proof scheme we need to introduce a new parameter for a PCP which requires the notion of fractional chromatic number of Section 3.4.4 (Definition 3.6).

Definition 3.18 (Covering Parameter) *The Covering Parameter of a PCP with verifier V is*

$$\rho = \min_{x \in L} \frac{1}{\chi_f(G_V(x))} \tag{10}$$

Here $G_V(x)$ is the FGLSS graph for V and x.

Since for every graph G we have $\chi_f(G) \geq \omega(G)$, the Covering Parameter of a PCP for input x is at most $2^{-free(|x|)}$. The notion of Covering Parameter first appears in [FeKi2] in the context of RPCPs. RPCPs are PCPs such that for every $x \in L$ there is a probability distribution (possibly different for different inputs) on all the proofs. For a fixed $x \in L$ this distribution gives rise to a probability distribution on $Ind(\overline{G}_V(x))$, which in turn can be used to give a lower bound on the Covering Parameter of the PCP. This lower bound is what Feige and Kilian call the Covering Parameter of the RPCP. Whether we talk about RPCPs or immediately define the Covering Parameter of a PCP is a terminological question. RPCPs were introduced because their definition suggests an elegant connection to zero knowledge proof systems.

Feige and Kilian change the first step of the Lund Yannakakis proof and eliminate the third step. Here is their proof scheme: Assume that for an NP complete language L there is a polynomial time transformation that sends a word x into a graph $G(x)$ such that for some constants $0 < c_2 < c_1 < 1$:

$$\text{If } x \in L \quad \text{then} \quad \chi_f(G(x)) \leq 1/c_1 \tag{11}$$

$$\text{If } x \notin L \quad \text{then} \quad \alpha(G(x)) \leq c_2|V(G(x))| \tag{12}$$

Then, if we apply the randomized graph product construction of [37] on $G(x)$ with $k = c_3 log_{c_2^{-1}}|V(G(x))|$, where $c_3 > 1$ is a constant, and select a

random subgraph G' of G^k with size c_2^{-k}, then from Lemma 3.7 of Section 3.4.4:

$$\chi_f(G') \leq |V(G')|^{\frac{\log c_1}{\log c_2}} \qquad \text{if } x \in L, \tag{13}$$

$$\alpha(G') \leq |V(G')|^{\epsilon} \qquad \text{if } x \notin L, \tag{14}$$

where ϵ can be made arbitrarily small if c_3 is large enough. We immediately obtain:

Lemma 3.19 *Assume that for an NP complete language L there is a polynomial time transformation that sends a word x into a graph $G(x)$ such that for some constants $0 < c_2 < c_1 < 1$ conditions (11) and (12) hold. Then for every $\epsilon > 0$ we can construct a graph G' in randomized polynomial time such that:*

$$\chi(G') \leq |V(G')|^{\frac{\log c_1}{\log c_2}+\epsilon} \qquad \text{if } x \in L, \tag{15}$$

$$\chi(G') \geq |V(G')|^{1-\epsilon} \qquad \text{if } x \notin L. \tag{16}$$

Indeed the Inequality (15) follows from Inequality (14) and Inequality (16) is implied by Inequality (13) and by the following result of Lovász [106]:

$$\chi(G) \leq \chi_f(G) \log(1 + \alpha(G)). \tag{17}$$

If we combine Lemma 3.19 with Definition 3.18 of the covering parameter we obtain:

Lemma 3.20 *Suppose that one can generate a PCP for NP with $f(n)$ random bits, soundness probability $p = p(n)$, average free bit complexity $f_{av} = free_{av}(n)$ and $\rho = \rho(n)$. Assume that p, f_{av} and ρ are constants and that the size of the FGLSS graph, $2^{f(n)+f_{av}}$, is polynomially bounded as n grows to infinity. Then it is hard to approximate $\chi(G)$ to within $|V(G)|^{\frac{f_{av}-\log p+\log \rho}{f_{av}-\log p}-\epsilon}$, where ϵ is an arbitrarily small positive constant, assuming $NP \not\subseteq ZPP$.*

Starting from the construction of Håstad [86], Feige and Kilian shows the existence of PCPs for an NP complete problem such that $\log \rho/(f_{av} - \log p)$ is arbitrarily small. Theorem 3.17 is implied now by Lemma 3.20.

We can also use Lemma 3.19 to derive the Lund Yannakakis result from the non-approximability result from the MAX-SNP hardness of the MAX-3-coloring problem proved by Petrank [118]. This example was given by Feige and Kilian, and it is so easy that we can give their entire proof here:

Proof. [Theorem 3.16] Let H be the graph of Petrank's construction such that it has m edges and either it has a valid three coloring, or every three coloring of its vertices miscolor at least qm edges, where $q > 0$ is some fixed constant. From H we construct a graph G, which has $6m$ vertices of the form (e, c), where e ranges over the m edges of H and c ranges over the 6 valid 3-colorings of the two end-points of an edge. Two vertices (e_1, c_1) and (e_2, c_2) are connected by an edge if c_1 and c_2 intersect at a vertex of H and c_1 and c_2 disagree on the coloring of this vertex. A valid 3-coloring C of H induces an independent set of size m in G. Let π be a permutation of the three colors. As π ranges over its 6 possibilities, the composition of C with π induces 6 independent sets in G of size m. Furthermore, these independent sets are disjoint and cover $V(G)$. Hence G has a chromatic number of 6. If H is not 3-colorable (and hence qm of its edges are miscolored) then the largest independent set of G is of size at most $m(1-q)$. If we apply Lemma 3.19 for G with $c_1 = 1/6$, $c_2 = (1-q)/6$, we get Theorem 3.16. □

Finally we mention another type of in-approximability result for $\chi(G)$. S. Khanna, N. Linial and S. Safra [97] have shown that it is NP-hard to tell apart 3 chromatic graphs from 5 chromatic graphs. But the following question is open, and would be very interesting to resolve:

Problem 3.21 *Prove that for any fixed $k > 3$ it is NP hard to tell apart a k chromatic graph from a three chromatic graph.*

3.9 Set Cover

Instance: A collection $F = \{S_1, \ldots, S_s\}$ of subsets of $S = \{1, \ldots, n\}$.

Solution: A sub-collection F' of F such that $\cup_{S_i \in F'} S_i = S$.

Objective: $\nu(F) = \min_{F'} |F'|$.

In 1974 D. Johnson [90] showed that the greedy algorithm finds a collection F' for the problem such that $|F'|$ is to within $\log n$ factor optimal (here the log is based on $e = 2.71..$). Chvatal [46] extended this algorithm to the weighted version, and Lovász [106] showed that a linear programming relaxation of the integer program for the set cover also works in time $\log n$.

The first hardness result is that of Lund and Yannakakis [108]. They show using a construction coming from the PCP theory that set cover cannot be approximated to within a factor of $\frac{\log n}{4}$ unless $NP \subseteq TIME(n^{polylogn})$,

and within a factor of $\frac{\log n}{2}$ unless $NP \subseteq ZIME(n^{polylogn})$. Here $ZTIME(t)$ denotes the class of problems for which there is a probabilistic algorithm that makes no error and runs in expected time t.

Subsequent works went in two different directions. The best result up to date that represent these directions are that of Feige [61] which proves that set cover cannot be approximated efficiently to within a factor of $(1 - o(1)) \log n$ unless $NP \subseteq TIME(n^{O(\log \log n)})$, and that of Raz and Safra [122] which proves that set cover is NP hard to approximate within a factor of $\epsilon \log n$ for some fixed $\epsilon > 0$. Until now there is no result which would achieve both optimal non-approximability ratio and hardness condition $P \neq NP$.

A predecessor of [61] was the result of Bellare et.al. [32], which proved that the set cover cannot be approximated within any constant ratio unless $P = NP$, and that it cannot be approximated within a factor of $\frac{\log n}{4}$ unless $NP \subseteq TIME(n^{O(\log \log n)})$. Arora and Sudan [13] follow-up on [122] and achieve the same bound using elegant and difficult algebraic techniques such as Hilbert's Nullstellensatz.

We cannot give here the technically very involved result of Raz and Safra, but we present a complete proof of Feige's result. The proof is a modification of a simplified argument due to Håstad [87]. At many places we use his original wording. We ought to remark that simplification is formal to a large extent: the key components of the proof are the same as those in [61].

Theorem 3.22 (Feige [61]) *The set cover cannot be approximated efficiently to within a factor of* $(1-o(1)) \log n$ *unless* $NP \subseteq TIME(n^{O(\log \log n)})$. *The logarithm is* $e = 2.71\ldots$ *based.*

Proof. A partition system $B(m, p, k, d)$ [61] is a system of p partitions, $(p_i)_{i=1}^{p}$, on a basis set B with $|B| = m$ such that:

1. Each p_i $(1 \leq i \leq p)$ consists of k parts. The parts of p_i are denoted $p_{i,j}$ $(1 \leq j \leq k)$.

2. No collection $(p_{i_l,j_l})_{l=1}^{d}$ of d partition-segments with all i_l distinct covers the universe.

In other words if we only use sets from different partitions to cover, we need at least d sets, while a small cover is given by the k sets of one p_i.

Lemma 3.23 (Feige [61]) *For any $c > 0$ there exists a $B(m, p, k, d)$ partition system with $p \leq (\log m)^c$, k an arbitrary constant and $d = (1 - f(k))k \ln m$, where $f(k)$ tends to 0 when k tends to infinity, and m is large enough compared to k and c.*

Feige shows that a random partition system satisfies Properties 1-2, and since d is rather small we can simply check the properties. The construction can also be done deterministically. Now we describe our instance of set-cover.

Our starting point is the probabilistic verifier V^{ℓ}_{3SAT} of Lemma 3.4 in Section 3.4.2 for an NP complete language L with the choice of $\ell = c_0 \log\log n$, where we shall choose $c_0 > 0$ to be large enough. Recall that this verifier is associated with a gap-3SAT instance ϕ_x with N Boolean variables. Note that $|x| = n$ is in polynomial relationship with N. On the other hand, the instance of setcover we construct will have slightly super-polynomial size in n.

The verifier sends a subset U of variables with $|U| = \ell$ to the first prover who answers with an assignment σ_U to these variables. Independently, he sends a W companion of U to the second prover, i.e. a set of ℓ clauses containing these variables. The second prover answers with the evaluation of the 3ℓ variables in W. The verifier rejects if not all clauses are satisfied in W, or if σ_U is not equal to the projection $\pi_U(\sigma_W)$, i.e. the assignment that σ_W gives to U. What is important for our purposes is that if $x \notin L$, the verifier accepts with probability at most c_1^{ℓ} for some fixed $0 < c_1 < 1$. The structure of ϕ_x determines all U, W companion pairs, all the projections π_U, and we can compute these in $DTIME(n^{O(\ell)})$ from x. From now on, when we talk about a U, W pair, we always mean a pair, where W is a companion of U.

Let $k > 0$ be a constant such that $f(k)$ of Lemma 3.23 is less than ϵ. For each U and each k-tuple $W_1, W_2 \ldots W_k = \vec{W}$ of possible W companions of this U we construct a separate partition system on a new set of m points. We have N^{ℓ} different U and, given the choice of U, each W_i can be chosen in 5^{ℓ} ways. We thus have $R = N^{\ell}5^{k\ell}$ partition systems, and with the choice of $m = R^{\frac{1}{\epsilon}}$, the total size of the universe is $R^{1+\frac{1}{\epsilon}} = m^{1+\epsilon}$. We set the further parameters of the partition systems as $p = 2^{\ell}$, k, and $d = (1 - f(k))k \ln m$. Since $p = 2^{c_0 \log\log n} = (\log n)^{c_0}$ and $\log m > \log n$, the conditions of Feige's lemma hold, and the existence of such partition systems is guaranteed. Notice that $m = n^{O(\log\log n)}$ for a fixed ϵ. A particular set in one of the partition systems has an index given by $U, \vec{W}, \alpha \in \{0,1\}^{\ell}$ and

$i \in [k]$, and we denote it by $S_{U,\vec{W},\alpha,i}$.

The sets in our set-cover instance are indexed by W and β where $\beta \in \{0,1\}^W$ should be thought of as an answer from the second prover on the question W. We denote it by $T_{W,\beta}$ and it is a union of $S_{U,\vec{W},\alpha,i}$ which satisfy

$$(W_i = W) \wedge (\pi_U(\beta) = \alpha),$$

where we of course assume that β satisfies the clauses used to construct W.

Let $Q = (5N/3)^\ell$ be the number of different W. We first have the simple lemma telling us what happens when $x \in L$.

Lemma 3.24 *If $x \in L$, the associated set system has a cover of size Q.*

Proof. We cover the universe with $\{T_{W,\sigma_W}\}_W$, where σ_W is the answer of the second prover to question W. Since the answers are consistent with those of the first prover, for each U and $\vec{W}$ there is some σ_U so that the chosen sets contain $S_{U,\vec{W},\sigma_U,i}$ for all i. This σ_U is simply the answer of the first prover to question U. By definition, the system $\{S_{U,\vec{W},\sigma_U,i}\}_{1 \le i \le k}$ covers the m points of the associated $U, \vec{W}$ pair. □

Thus we just need to prove the lemma below (and choose c_0).

Lemma 3.25 *If $x \notin L$, the associated set system has a no cover of size $dQ/(k(1+\epsilon))$.*

Proof. Suppose we have a cover of size rQ. For each W there is a set A_W of assignments β on W such that $T_{W,\beta}$ belong to the cover. By assumption

$$\sum_W |A_W| = rQ. \tag{18}$$

We note that none of the A_W is empty since we need to cover the partition systems with $W_i = W$ for all i. Let us call a W *good* if $|A_W| \le r(1+\epsilon)$ We claim that for appropriate choice of c_0 either $r > \log m$ or there is $U, \vec{W}$ such that

1. $\pi_U(A_{W_i})$ $(1 \le i \le k)$ are pairwise disjoint,
2. W_i $(1 \le i \le k)$ are good.

Let us recall a simple lemma in set theory:

Lemma 3.26 *Let $A_1, \ldots, A_q$ be sets of size at most s with the property that no k of them are pairwise disjoint. Then there is an element which is contained in at least $\frac{1}{(k-1)s}$ fraction of the sets.*

Proof. Without loss of generality we may assume that $A_1, \ldots, A_l$ are pairwise disjoint, but for all $j > l$ the set A_j intersects $A = \cup_{i=1}^{l} A_i$. Note that by our assumption $l \leq k-1$, hence $|A| \leq (k-1)s$. Since every A_i intersects A, there must be an element of A which is contained at least $\frac{1}{(k-1)s} \leq \frac{1}{|A|}$ of the sets. □

Let

$$S_U = \{\pi_U(A_W) \mid W \text{ is a companion of } U, \text{ and } W \text{ is good}\}$$

be a set system on the assignments for U and let σ_U be an assignment which is contained in as large fraction of elements of S_U as possible (If S_U is empty then σ_U is arbitrary). Consider the strategy of the two provers when to question U the first prover answers σ_U, and to question W the second prover answers with a random $\sigma_W \in A_W$. (So the strategy of the second prover is randomized. An averaging argument shows that there is an equally good deterministic strategy.)

If for a fixed U there is no $\vec{W}$ such that both Conditions 1 and 2 hold, then among the good companions of U there are no $W_1, \ldots, W_k$ such that $\pi_U(A_{W_i})$ $(1 \leq i \leq k)$ are pairwise disjoint. By applying Lemma 3.26 to S_U we obtain that there is an assignment to U which occurs in at least $\frac{1}{(k-1)r(1+\epsilon)}$ fraction of the sets in S_U, and if fact σ_U is such. In what follows we assume that Conditions 1. and 2. do not hold for any $U, \vec{W}$.

By Equation (18) the expected value of $|A_W|$ for a random W is r, so for at least $1-(1+\epsilon)^{-1}$ fraction of Ws we have $|A_W| \leq r(1+\epsilon)$. The verifier chooses such a W with probability at least $1-(1+\epsilon)^{-1}$. Conditioned on the intersection of this event (i.e. that W is good) and the event that the verifier picks some fixed U, by our earlier remark, the verifier chooses a W such that $\pi_U(A_W)$ contains σ_U with probability at least $\frac{1}{(k-1)r(1+\epsilon)}$. Hence the probability of the event that the verifier picks a U,W pair such that W is good and $\sigma_U \in \pi_U(A_W)$, is at least

$$\frac{1-(1+\epsilon)^{-1}}{(k-1)r(1+\epsilon)}. \tag{19}$$

In the case of this event there is a $\frac{1}{|A_W|} \geq \frac{1}{r(1+\epsilon)}$ chance over $\sigma_W \in A_W$ that $\pi_U(\sigma_W) = \sigma_U$. Combining this with Equation (19) we get that V^{ℓ}_{3SAT}

accepts the described strategy of the two provers with probability at least

$$\frac{1-(1+\epsilon)^{-1}}{(k-1)r(1+\epsilon)}\frac{1}{r(1+\epsilon)} = \Omega(r^{-2}). \tag{20}$$

Either $r > \log m$ or, since $\log m = O(c_0 \log n \log\log n)$, we can choose a large enough c_0 that depends on ϵ, k, c_1 and $\log_n N = O(1)$, such that $2^{-c_1 \ell} = 2^{-c_1 c_0 \log\log n}$ becomes smaller than Expression (20), and we arrive at a contradiction.

It follows from the contradiction that there exist an U and $\vec{W}$ such that Conditions 1 and 2 hold. Fix this choice of U and $\vec{W}$ and consider how the elements from the corresponding partition system are covered. Since $|W_i| \leq r(1+\epsilon)$ for $1 \leq i \leq k$, the cover system has at most $kr(1+\epsilon)$ sets. By the disjointness of $\pi_U(A_{W_i})$s, these sets all come from distinct partitions. Using the property of the cover system we conclude that $kr(1+\epsilon) > d$.

In the case $r > \log m$, since we have $d \leq (1-f(k))kr$, we conclude the same. We are done, since the cover size, $|rQ| > dQ/k(1+\epsilon)$, as claimed. □

Since the universe for the set cover instance has total size $t = m^{1+\frac{1}{\epsilon}}$, we get that the quotient of the minimum cover sizes when $x \notin L$ versus when $x \in L$ is:

$$\frac{1-f(k)}{1+\epsilon}\log m = \frac{1-f(k)}{(1+\epsilon)^2}\log t,$$

which tends to $\log t$ when $k \to \infty$ and $\epsilon \to 0$.

3.10 Some Other Interesting Problems

There are several interesting unresolved problems in the theory of non-approximability. Here we survey some of them.

Problem Name: Minimum Vertex Cover

Instance: An undirected graph G.

Solution: A vertex cover for G, i.e., a subset $C \subseteq V(G)$ such that, for each edge (a,b), at least one of a and b belongs to C.

Objective: $\min_C |C|$.

The problem is approximable within $2 - \frac{\log\log|V|}{2\log|V|}$ [112] and [26]. Using a transformation from bounded MAX3SAT Håstad showed that the Minimum

Vertex Cover is not approximable within 1.1666 [86]. It is conjectured that the exact non-approximability ratio is $2 - \epsilon$.

Problem Name: MAX-CUT

Instance: An undirected graph G.

Solution: A cut for G, i.e., a subset $C \subseteq V(G)$.

Objective: $\max_C |\{(a,b) \in E(G) \mid a \in C, b \in \overline{C}\}|$.

By the result of Håstad [86] MAX-CUT is not approximable within 1.0624. In their seminal paper Goemans and Williamson [78] use semidefinite programming to approximate MAX-CUT to within a factor of 1.138. Analysis shows that their proof method cannot be extended in a simple way to get a better approximation algorithm. On the other hand this analysis also shows that unless ratio of the result of the semidefinite relaxation to the number of edges is exactly 0.845..., MAX-CUT is always better approximable. Are indeed graphs with this ratio the hardest to approximate, or is there an improvement to the GW algorithm?

Problem Name: Closest Codeword

Instance: Linear binary code C of length n and $\mathbf{x} \in \mathbf{F}_2^n$.

Solution: A codeword $\mathbf{y}$ of C.

Objective: $\min_{\mathbf{y}} Dist(\mathbf{x}, \mathbf{y})$, where *Dist* is the Hamming distance.

The closest codeword problem is not approximable within $2^{\log^{1-\epsilon} n}$ unless for any $\epsilon > 0$ unless $NP \subseteq QP$ [7]. Can we get a non-approximability ratio which is a root of n? The same non-approximability ratio and the same question applies to the Closest Lattice Vector problem.

Problem Name: Shortest Lattice Vector (SLP)

Instance: A lattice L in $\mathbf{Z}^n$.

Solution: A lattice vector $\mathbf{x} \in L$.

Objective: $\min_{\mathbf{x}} \sum_{i=1}^{n} x_i^2$.

D. Micciancio [110] has shown that SLP cannot be approximated within a factor of $\sqrt{2}$ unless $NP \subseteq ZPP$. It would be very interesting to show the non-approximability of SLP for an *arbitrary* constant.

Another question concerns MAX-SNP problems with sparsity constraints such as MAX3SAT with the restriction that each variable can occur in at most k clauses. Berman and Karpinski [36] have obtained explicit lower bounds for many of these instances (see the table below), but it would be interesting to reduce the gaps in between the factors of non-approximability and the factors that our approximation algorithms can achieve. Recently Feige, Karpinski and Langberg [63] made a step in this direction by modifying the MAX-CUT algorithm of Goemans and Williamson [78] so that the new algorithm works particularly well for bounded degree graphs. Yet, the gaps are still huge.

k-OCC = Every variables occur at most k times
Ek-LIN2 = Linear system over $\mathbf{F}_2$ with at most k variables in each equation
k-MAX-CUT = The maximum degree is k
k-MIS = Maximum independent set of graph with max degree k
k-Node Cover = The maximum degree is k
MIN-SBR = Sorting by reversal

Problem	Approx. Upper	Approx Lower
3-OCC-E2-LIN2	1.1383	1.003
3-OCC-E3-LIN2	2	1.0163
3-MAX-CUT	1.1383	1.003
3-OCC-MAX 2SAT	1.0741	1.0005
6-OCC-MAX 2SAT	1.0741	1.0014
3-MIS	1.2	1.0071
4-MIS	1.4	1.0136
5-MIS	1.6	1.0149
3-Node Cover	1.1666	1.0069
4-Node Cover	1.2857	1.0128
5-Node Cover	1.625	1.0138
MIN-SBR	1.5	1.0008

4 Structural consequences of the PCP Theory

What makes some NPO problems hard to approximate, while others easy? This natural question had already been raised by Johnson in [90] long before the PCP theory was developed: "Is there some stronger kind of reducibility than the simple polynomial reducibility that will explain [approximation] results, or are they due to some structural similarity between the problems as we define them?" In this section we present two completeness results that are consequences of the basic PCP theorem. These results give an evidence that different problems in the same non-approximability classes have the same reason for their hardness of approximation.

4.1 Polynomial Time Approximation Schemes

Not all natural NP-hard optimization problems are hard to approximate. Easy classes include PTAS and FPTAS. For the definition of these classes and for that of the class APX see the table below.

	FPTAS	PTAS	APX
Quantifiers	$\exists c\, \forall \epsilon > 0\, \exists A_\epsilon$	$\exists c(\epsilon)\, \forall \epsilon\, \exists A_\epsilon$	$\exists c\, \exists C > 1\, \exists A$
Algorithm	A_ϵ	A_ϵ	A
Running Time	$(\|x\| + 1/\epsilon)^c$	$\|x\|^{c(\epsilon)}$	$\|x\|^c$
Approximability Factor	$1+\epsilon$	$1+\epsilon$	C

Clearly, the following inclusions hold:

$$FPTAS \subseteq PTAS \subseteq APX \subseteq NPO.$$

These inclusions are strict if $P \neq NP$. Cesati and Trevisan [45] also introduce the class *EPTAS* which differs from *FPTAS* in that the bound on the running time is $f(\epsilon)n^c$, where $f(\epsilon)$ is an arbitrary function of ϵ, and $c > 0$ is an arbitrary constant.

An optimization problem is practically tractable if it has a fully polynomial time approximation scheme, or somewhat weaker, an EPTAS. If a problem has a PTAS, but not EPTAS, then there is ϵ such that the $|x|^{c(\epsilon)}$ running time practically prohibits approximating it within a factor better than $1+\epsilon$.

4.2 Approximation Preserving Reductions

Approximation preserving reductions in between NPO problems are used to show that if a problem P_1 is easy to approximate, then any problem P_2

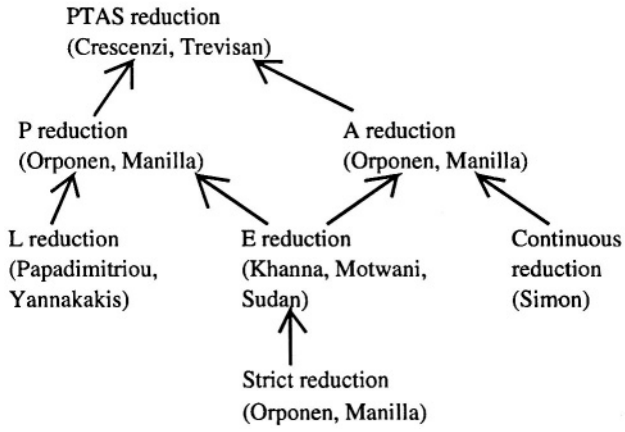


Figure 4: The taxonomy of Approximation preserving reducibilities. (From the courtesy of Crescenzi et. al. [53])

is also easy to approximate which reduces to P_1. Since the easiness of an approximation problem is associated with its membership in *PTAS,* almost all approximation preserving reductions preserve membership in PTAS.

The first paper which defines an *approximation preserving reduction* was that of Orponen and Mannila [114]. Up to the present time there are at least eight notions of approximation preserving reductions in use with a similar overall scheme. This scheme is the following:

Let $P_1(x, y)$ and $P_2(x', y')$ be two polynomial time functions that are to be optimized for y and y' (maximized or minimized in an arbitrary combination). Let $OPT_1(x)$ and $OPT_2(x')$ be the optimum of these problems. A *reduction* assumes two maps:

1. a map f to transform instances x of P_1 into instances $x' = f(x)$ of P_2 [Instance Transformation],
2. a map g to transform (input, witness) pairs (x', y') of P_2 into witnesses y of P_1.

   [Witness Transformation].

Let $OPT_1 = OPT_1(x)$, $OPT_2 = OPT_2(h(x))$, $APPR_1 = P_1(x, g(h(x), y'))$, and $APPR_2 = P_2(h(x), y')$. The centerpiece of any approximation scheme is a relation which is required to hold between these four quantities. This relation must roughly say: "If $APPR_2$ well approximates OPT_2, then $APPR_1$

well approximates OPT_1." To see that indeed this is what we need, assume that we have a PTAS for OPT_2, and that P_1 reduces to P_2. In order to get a good approximate solution for $OPT_1(x)$, where x is an arbitrary input instance of P_1, first we construct $h(x)$, and find a witness y' such that $P_2(h(x), y')$ approximates $OPT_2(h(x))$ well. By the central assumption of the reduction $P_1(x, y)$ well approximates $OPT_1(x)$ for $y = g(h(x), y')$. For the above argument to hold f and g must be computable in polynomial time.

Different reductions differ in the relation required in between the four quantities. The *L*-reduction of Papadimitriou and Yannakakis [116] requires that OPT_2 is upper bounded by $c_1 OPT_1$, and that $|APPR_1 - OPT_1|$ is upper bounded by $c_2|APPR_2 - OPT_2|$ for some constants c_1 and c_2. It follows from the next lemma, that *L*-reduction preserves PTAS.

Lemma 4.1 *A reduction scheme preserves PTAS iff it enforces, that* $|APPR_1 - OPT_1|/OPT_1 \to 0$ *whenever* $|APPR_2 - OPT_2|/OPT_2 \to 0$.

4.3 APX Complete Problems

APX is the class of constant factor approximable NPO problems, and APX-PB is its subclass with problems that have a polynomial bound on their objective function. Our goal in this section is to give complete problems for APX and $APX - PB$.

Completeness proofs assume an underlying reduction. Even though in the theory of APX the L reduction is the most widely used reduction, in [55] it has been shown that it does not allow to reduce some problems which are known to be easy to approximate to problems which are known to be hard to approximate. Another flaw of the L reduction is that it is too week in another sense, namely it is not always approximation preserving unless $P = NP \cap co - NP$ [53]. Therefore we cannot hope for completeness results for APX or APX–PB with respect to the L reduction. Instead, S. Khanna, R. Motwani, M. Sudan and U. Vazirani [98] define the E reduction which, using the notation of the previous section, requires, that

$$\max\left\{\frac{APPR_1}{OPT_1}, \frac{OPT_1}{APPR_1}\right\} \leq 1 + c\left(\max\left\{\frac{APPR_2}{OPT_2}, \frac{OPT_2}{APPR_2}\right\} - 1\right)$$

for some $c > 0$. It can be easily shown that the E reduction preserves PTAS. Using the basic PCP theorem Khanna et. al. show:

Theorem 4.2 (Khanna, Motwani, Sudan and U. Vazirani [98]) *The MAX SAT problem is complete for APX – PB with respect to the E reduction. Also,*

$$APX - PB = \overline{MAX\ SNP} = \overline{MAX\ NP},$$

where the closure means closure under the E reduction.

The E reducibility is still somewhat too strict. In [55] it has been shown that natural PTAS problem exists, such as $MAX\ KNAPSACK$, which are not E-reducible to to polynomially bounded APX problems such as $MAX3SAT$. This drawback is mainly due to the fact that an E reduction preserves optimum values (see [55]). Crescenzi, Kann, Silvestri and Trevisan [53] develop a reduction where functions f and g (see previous section) are allowed to depend on the *performance ratio,* where the performance ratio of an NPO problem A is defined as the function:

$$R_A(x,y) = \max\left\{\frac{A(x,y)}{OPT_A(x)}, \frac{OPT_A(x)}{A(x,y)}\right\}.$$

Definition 4.3 (AP Reduction [53]) *Let A and B be two NPO problems. A is said to be AP-*reducible *to B, if two functions f and g and a positive constant α exists such that:*

1. *For any x and for any $r > 1$, $f(x,r)$ is computable in time $t_f(|x|,r)$.*
2. *For any x and for any $r > 1$, and for any y, $g(x,y,r)$ is computable in time $t_g(|x|,|y|,r)$.*
3. *For any fixed r both $t_f(.,r)$ and $t_g(.,.,r)$ are bounded by a polynomial.*
4. *For any fixed n, both $t_f(n,.)$ and $t_g(n,n,.)$ are non-increasing functions.*
5. *For any x and any $r > 1$, and for any y $R_B(f(x,r),y) \leq r$ implies*

$$R_A(x,g(x,y,r)) \leq 1 + \alpha(r-1).$$

The triple (f,g,α) is said to be an LP reduction from A to B.

In [53] the following is claimed:

Theorem 4.4 *MAX SAT is APX complete with respect to the AP reduction.*

5 Checkable and Holographic Codes

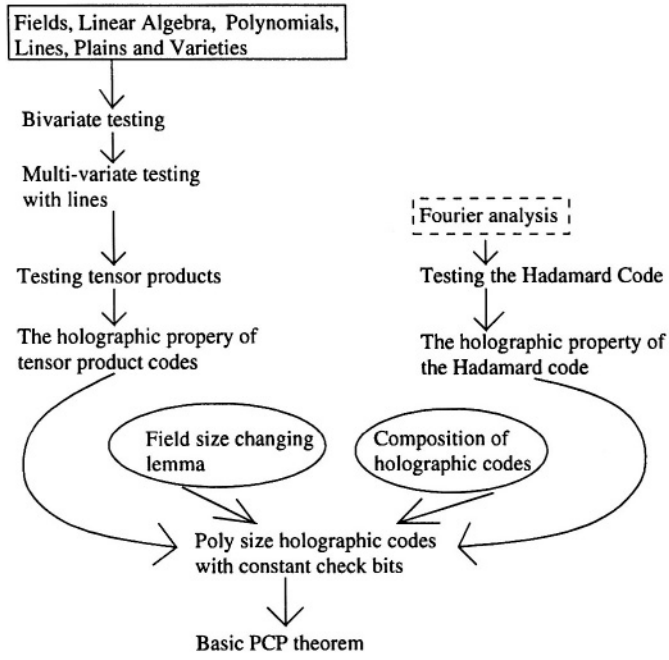


Figure 5: Technical components of the basic PCP theorem

The theory of PCP requires the construction of codes with properties that have not been considered by coding theorists before. In this section we define these properties and build codes with them. Our last construction is used as a black box in Section 2.4 to prove the basic PCP theorem.

The constructions we present are based on the results of Arora and Safra [12], Arora, Lund, Motwani, Sudan and Szegedy [10], Poischuk and Spielman [120], and the thesis of Madhu Sudan [131], but we have re-shuffled parts of their arguments, and simplified them whenever we could.

A major departure from previous presentations is that we eliminated the cryptographic language from this part, and use the language of algebra, combinatorics and coding theory. In order to make our explanation entirely self contained, we included a section of definitions and prerequisites.

5.1 Definitions and Prerequisites

In the current version(s) of the PCP theory the technical details are numerous. It is the author's belief that these technicalities will be simplified in the future, but the need for using fields, polynomials, codes and compound combinatorial constructs will not be eliminated altogether.

5.1.1 Fields

The field is perhaps the most fundamental structure in algebra. A field is a finite or infinite set on which two binary operations are defined: addition and multiplication. Examples to fields include:

Name	Notation
The field of rational numbers	$\mathbf{Q}$
The field of real numbers	$\mathbf{R}$
The field of complex numbers	$\mathbf{C}$
The Galois field of order q	$\mathbf{F}_q$

Only the last field is finite, but it is the one we will need in this chapter. The set of natural numbers $\mathbf{N} = \{0,1,\ldots\}$ is not a field, because addition and multiplication do not have inverses. Formally, a field $\mathbf{F}$ consists of a basic set, which we also denote with $\mathbf{F}$, a zero element, $0 \in \mathbf{F}$, and two functions from $\mathbf{F} \times \mathbf{F}$ to $\mathbf{F}$ called addition and multiplication such that the following axioms hold (the quantifiers range through $\mathbf{F}$):

F1:	$(\forall a, b)$	$a + b = b + a$	[Commutativity]
F2:	$(\forall a, b, c)$	$a + (b + c) = (a + b) + c$	[Associativity]
F3:	$(\exists 0)(\forall a)$	$a + 0 = a$	[Existence of Zero]
F4:	$(\forall a)(\exists - a)$	$a + (-a) = 0$	[Additive inverse]
F5:	$(\forall a, b)$	$ab = ba$	[Commutativity]
F6:	$(\forall a, b, c)$	$a(bc) = (ab)c$	[Associativity]
F7:	$(\exists 1 \neq 0)(\forall a)$	$a1 = a$	[Existence of Unit]
F8:	$(\forall 0 \neq a)(\exists a^{-1})$	$aa^{-1} = 1$	[Multiplicative inverse]
F9:	$(\forall a, b, c)$	$a(b + c) = ab + ac$	[Distributivity]

Axioms **F3** and **F7** assert the existence of additive and multiplicative unit elements. Axioms **F4** and **F8** assert the existence of additive and multiplicative inverses. Axioms **F1-F9** are sufficient to derive other basic known facts about fields such as the uniqueness of 0 and 1 and the uniqueness of additive and multiplicative inverses. Axiom **F7** implies that a field always has at least two elements.

Fields are the basic building blocks of many constructions in combinatorics, e.g. of expander graphs, projective planes, codes and block designs. A field is finite if its basic set is finite. Finite fields were introduced by the French mathematical genius, Galois, who proved that in general roots of equations of degree greater than four cannot be expressed in terms of radicals.

For every prime power q there is exactly one field, $\mathbf{F}_q$ (up to isomorphism) of size q. Conversely, the size of a finite field is always a prime power. If q is prime then $\mathbf{F}_q$ can be identified with the modulo q arithmetic acting on the residue classes $\{0, 1, \ldots, q-1\}$. If $q = p^\alpha$, where p is a prime number and $\alpha > 1$, the description of $\mathbf{F}_q$ is more complicated.

Definition 5.1 (Subfield) *Let* $\mathbf{K}$ *be a field. A set* $\mathbf{F} \subseteq \mathbf{K}$ *is a subfield of* $\mathbf{K}$, *if it forms a field with respect to the operations adopted from* $\mathbf{K}$. *This means that* $0, 1 \in \mathbf{F}$, *and every* $a, b \in \mathbf{F}$ *with and* $a \neq 0$ *we have that* $-a, a^{-1}, a+b, ab$ *are also elements of* $\mathbf{F}$.

The field $\mathbf{Q}$ is a subfield of $\mathbf{R}$ and $\mathbf{R}$ is a subfield of $\mathbf{C}$. The field $\mathbf{F}_2 = \{0, 1\}$ is a subfield of $\mathbf{F}_4 = \{0, 1, j, 1+j\}$, where j is defined by the equation $j^2 + j + 1 = 0$. This equation has no solution in $\mathbf{F}_2$. In general, $\mathbf{F}_{p^\alpha}$ is a subfield of $\mathbf{F}_{p^\beta}$ for any $\beta \geq \alpha$. We can obtain $\mathbf{F}_{p^\beta}$ by adjoining roots of irreducible polynomials of $\mathbf{F}_{p^\alpha}$ in the same way as we obtained $\mathbf{F}_4$ from $\mathbf{F}_2$. This mechanism is called *field extension.*

Definition 5.2 (Characteristic) *The characteristic of a field* $\mathbf{F}$ *is the cardinality of the set* {1,1 + 1,1 + 1 + 1,...}, *if it is finite, and 0 otherwise.*

The characteristic of a field is always zero or a prime number.

5.1.2 Vector Spaces

Vector spaces or in other word *linear spaces* are constructions arising from fields. A vector space V over a field $\mathbf{F}$ is a set equipped with a binary operation on V called addition, and with an operation called scalar multiplication that maps $\mathbf{F} \times V$ to V, such that the following axioms hold:

V1:	$(\forall \mathbf{x}, \mathbf{y} \in V)$	$\mathbf{x}+\mathbf{y}=\mathbf{y}+\mathbf{x}$
V2:	$(\forall \mathbf{x}, \mathbf{yz} \in V)$	$(\mathbf{x}+\mathbf{y})+\mathbf{z}=\mathbf{y}+(\mathbf{x}+\mathbf{z})$
V3:	$(\exists \mathbf{0} \in V)(\forall \mathbf{x} \in V)$	$\mathbf{x}+\mathbf{0}=\mathbf{0}$
V4:	$(\forall \mathbf{x} \in V)(\exists -\mathbf{x} \in V)$	$\mathbf{x}+(-\mathbf{x})=\mathbf{0}$
V5:	$(\forall c \in \mathbf{F})(\forall \mathbf{x}, \mathbf{y} \in V)$	$c(\mathbf{x}+\mathbf{y})=c\mathbf{x}+c\mathbf{y}$
V6:	$(\forall c, d \in \mathbf{F})(\forall \mathbf{x} \in V)$	$(c+d)\mathbf{x}=c\mathbf{x}+d\mathbf{x}$
V7:	$(\forall c, d \in \mathbf{F})(\forall \mathbf{x} \in V)$	$(cd)\mathbf{x}=c(d\mathbf{x})$
V8:	$(\forall \mathbf{x} \in V)$	$1\mathbf{x}=\mathbf{x}$

A set $S \subseteq V$ is linearly independent if $\sum_{i=1}^{k} c_i \mathbf{x}_i = \mathbf{0}$ implies $c_1 = \ldots = c_k = 0$ for all $c_1, \ldots, c_k \in \mathbf{F}$ and and $\mathbf{x}_1, \ldots, \mathbf{x}_k \in S$.

Lemma 5.3 (Dimension) *Let V be a vector space. All maximal linearly independent sets of V have the same cardinality. This cardinality is called the* dimension *of V and is denoted by* $\dim V$.

Vector spaces over a field $\mathbf{F}$ of dimension n are isomorphic with $\mathbf{F}^n$, the space of all n-tuples

$$\{(x_1, \ldots, x_n) \mid x_1, \ldots, x_n \in \mathbf{F}\},$$

where addition and multiplication with scalars are defined coordinate-wise. For this reason vector spaces over a field $\mathbf{F}$ of finite dimension are determined by their dimension up to isomorphism! From the field and vector space axioms the following is straightforward:

Lemma 5.4 *Let $\mathbf{F}$ be a subfield of $\mathbf{K}$. Then $\mathbf{K}$ is a vector space over $\mathbf{F}$.*

Definition 5.5 (Subspace) *Let V be a vector space over $\mathbf{F}$. A subset $W \subseteq V$ is a subspace of V (in notation $W \leq V$) if multiplication with scalars and addition does not lead out of W.*

Definition 5.6 (Direct Sum) *Let $V_1, \ldots, V_k$ be vector spaces over a field $\mathbf{F}$. We define $V = V_1 \oplus \cdots \oplus V_k$ as the vector space of the k-tuples $(v_1, \ldots, v_k)$, where $v_1 \in V_1, \ldots, v_k \in V_k$ and product with scalars and addition are defined coordinate-wise. We denote the tuple $(v_1, \ldots, v_k)$ with $v_1 \oplus \ldots \oplus v_k$.*

The dimension of $\bigoplus_{i=1}^{k} V_i$ is $\sum_{i=1}^{k} \dim V_i$. The dimension of a proper subspace of a vector space with finite dimension is less than that of the vector space.

5.1.3 Linear Maps

If V and W are vector spaces over a field $\mathbf{F}$ and φ is a map from V to W then φ is linear if:

1. $\varphi(c\mathbf{x}) = c\varphi(\mathbf{x})$ for every $\mathbf{x} \in V$ and $c \in \mathbf{F}$.
2. $\varphi(\mathbf{x}+\mathbf{y}) = \varphi(\mathbf{x}) + \varphi(\mathbf{y})$ for every $\mathbf{x}, \mathbf{y} \in V$.

Linear functions play a central role in linear algebra. The set of all linear functions from V to W is a vector space over $\mathbf{F}$ of dimension $(\dim V)(\dim W)$. In the special case when $W = \mathbf{F}$ we call the linear map a *linear function* (or functional). The space of all linear functions from V is called the dual space of V and is denoted by V^*. V^* has the same dimension as V. For a linear map $\varphi : V \to W$ we define:

- $\mathrm{Ker}(\varphi) = \{\mathbf{x} \in V \mid \varphi(\mathbf{x}) = \mathbf{0}\}$,
- $\mathrm{Im}(\varphi) = \{\mathbf{x} \in W \mid (\exists\, \mathbf{z} \in V)\ \ \mathbf{x} = \varphi(\mathbf{z})\}$.

Lemma 5.7 *Let $\varphi : V \to W$ be a linear map. Then*

$$\dim \mathrm{Ker}(\varphi) + \dim \mathrm{Im}(\varphi) = \dim V.$$

In particular, $\dim \mathrm{Im}(\varphi) \leq \dim V$.

5.1.4 Polynomials

An n-variate polynomial P over a field $\mathbf{F}$ is an expression of the form

$$P(x_1, \ldots, x_n) = \sum c_{\alpha_1, \cdots, \alpha_n} x_1^{\alpha_1} \cdots x_n^{\alpha_n}, \tag{21}$$

Often we abbreviate $\alpha = (\alpha_1, \ldots, \alpha_n)$ and $\mathbf{x} = (x_1, \ldots, x_n)$. We also abbreviate $c_\alpha = c_{\alpha_1, \cdots, \alpha_n}$, $\mathbf{x}^\alpha = x_1^{\alpha_1} \cdots x_n^{\alpha_n}$ and $\mathbf{F}[\mathbf{x}] = \mathbf{F}[x_1, \ldots, x_n]$ for the vector space of all n-variate polynomials over $\mathbf{F}$. The polynomials are treated as formal expressions, and if $P, Q \in \mathbf{F}[\mathbf{x}]$, we can define $PQ \in \mathbf{F}[\mathbf{x}]$ by the usual rules of multiplication on the symbols. $\mathbf{F}[\mathbf{x}]$ is not a field, since for instance x_1 does not have an inverse. Certain finite subspaces of $\mathbf{F}[\mathbf{x}]$ are of particular interest for us.

Definition 5.8 (Multi-Degree) *Let*

$$\mathbf{F}^{\mathbf{d}}[\mathbf{x}] = \left\{ \sum_{\alpha < \mathbf{d}} c_\alpha \mathbf{x}^\alpha \right\}, \tag{22}$$

where $\alpha < \mathbf{d}$ *is defined via* $\alpha_1 \leq d_1, \ldots, \alpha_n \leq d_n$. *This set is called the space of* multi-degree-$\mathbf{d}$ *polynomials.*

A special case of the above definition is when $d_1 = \ldots = d_n$. For an n-tuple $(d, \ldots, d)$ we shortly write $n \times d$. If it is clear from the context that the number of variables is n, we do not use the cumbersome multi-degree $n \times d$ notation, and we call $\mathbf{F}^{n \times d}[\mathbf{x}]$ simply the space of multi-degree d polynomials. If $d = \mathbf{1}$ then this space is the space of multi-linear polynomials.

A polynomial P is said to have multi-degree (exactly) $\mathbf{d}$ if $\mathbf{d}$ is the smallest vector (coordinate-wise) such that $P \in \mathbf{F}^{\mathbf{d}}[\mathbf{x}]$. In notation:

$$mdeg(P) = \mathbf{d}$$

A perhaps even more important degree concept is the *total degree,* or simply *degree* of a multi-variate polynomial.

Definition 5.9 ((Total) Degree) *Let:*

$$\mathbf{F}^{deg \leq d}[\mathbf{x}] = \left\{ \sum_{\alpha_1 + \cdots + \alpha_n \leq d} c_{\alpha_1, \cdots, \alpha_n} x_1^{\alpha_1} \cdots x_n^{\alpha_n} \right\}. \tag{23}$$

$\mathbf{F}^{deg \leq d}[\mathbf{x}]$ *is called the space of (total) degree at most* d *polynomials. We also use the notation:*

$$deg(P) = \min_{P \in \mathbf{F}^{deg \leq d}[\mathbf{x}]} d. \tag{24}$$

In the remaining part of this section we will focus on *evaluations* of polynomials. Let P be a polynomial of variables $x_1, \ldots, x_n$, and let $\mathbf{a} \in \mathbf{F}^n$. We obtain $P(\mathbf{a})$ if we replace x_i with a_i for $1 \leq i \leq n$ and compute the expression for P over $\mathbf{F}$. $P(\mathbf{a})$ is called the *replacement value* or *evaluation.*

When two polynomials, P and Q, are identical as formal polynomials we write $P = Q$. It may occur that $P(\mathbf{a}) = Q(\mathbf{a})$ for all $\mathbf{a} \in \mathbf{F}^n$, and yet $P \neq Q$. We sub-index the equation symbol with the name of the domain as in

$$P =_D Q \tag{25}$$

to express that P and Q are equal over $D \subseteq \mathbf{F}^n$. Since a high degree polynomial may coincide with a low degree polynomial on all points (think of x^p and x over $GF(p)$), we also need to introduce a degree notion that depends only on the replacement values. This will be the degree of the lowest degree polynomial that can represent the function in question. We denote this degree with deg', and its multi-degree variant with $mdeg'$, and call them *functional degree* and *functional multi-degree,* respectively. However, whenever we can, and if it does not lead to misunderstanding, we leave out the the apostrophe.

Lemma 5.10 *Let* $\mathbf{F}$ *be any field, and* $|X| \geq d+1$ *be a subset of* $\mathbf{F}$. *Let* P *and* Q *be* n*-variate polynomials over* $\mathbf{F}$ *with multi-degree at most* $\mathbf{d}$. *Then* $P =_{X^n} Q$ *implies* $P = Q$.

Lemma 5.11 *Let* $\mathbf{F}$ *be any field, and* X *be a subset of* $\mathbf{F}$ *with size* $d+1$. *Let* $f : X^n \to \mathbf{F}$ *be an arbitrary function. Then there is an* n*-variate polynomial* P *with multi-degree at most* $\mathbf{d}$ *such that* $P(\mathbf{a}) = f(\mathbf{a})$ *over* X^n. *(In other words,* F *has a multi-degree* $\mathbf{d}$ *extension.)*

Lemma 5.12 *Let* $\mathbf{F}$ *be a finite field and* $D \subseteq \mathbf{F}^n$ *be any domain of size at least* $|\mathbf{F}|^n - (|\mathbf{F}| - d)^n + 1$. *Let* P *and* Q *be* n*-variate polynomials over* $\mathbf{F}$ *with maximum degree* $\mathbf{d}$. *Then* $P =_D Q$ *implies* $P = Q$.

The above two lemmas concern multi-degree at most d polynomials. If the total degree of a polynomial is at most d, then its multi-degree is also at most d, but the converse does not hold. Next we give a sufficient condition for a polynomial to have (total) degree d.

Lemma 5.13 *Let* $|\mathbf{F}| > dn$. *If* $F : \mathbf{F}^n \to \mathbf{F}$ *is a degree at most* $\mathbf{d}$ *polynomial when restricted to any line of* $\mathbf{F}^n$ *then* F *is a polynomial of total degree at most* d.

Proof. The condition of the lemma can be expressed as $\deg_\lambda F(\mathbf{x}+\lambda\mathbf{y}) \leq d$. In particular, F is a multi degree $\mathbf{d}$ polynomial $\sum_{\alpha \leq \mathbf{d}} \mathbf{x}^\alpha$, where $\mathbf{d} = n \times d$. Therefore $F(\mathbf{x}+\lambda\mathbf{y})$ can be expressed as

$$\sum_{\alpha \leq \mathbf{d}} \sum_{\beta \leq \alpha} c_\alpha \lambda^{\sum_{i=1}^n \beta_i} \mathbf{x}^{\alpha-\beta} \mathbf{y}^\beta. \tag{26}$$

Observe that in expression (26) no two terms are the same, even if we disregard the λ-power and the coefficients. Let us denote the coefficient of

λ^j in (26) with $P_j(\mathbf{x},\mathbf{y})$. Since $P_{deg(F)}(\mathbf{x},\mathbf{y})$ has non-zero terms in it and hence it is not identically zero, and because of the individual degrees of the variables are less than $|\mathbf{F}|$, it does not disappear on the entire $\mathbf{F}^{2n}$ (see Lemma 5.10). Let us assume that $deg(F) > d$. Pick $\mathbf{x}_0$ and $\mathbf{y}_0$ such that $P_{deg(F)}(\mathbf{x}_0,\mathbf{y}_0) \neq 0$. Then $F(\mathbf{x}_0 + \lambda\mathbf{y}_0)$ is a degree $deg(F)$ polynomial in λ.

On the other hand, if we restrict F to the line $\mathbf{x}_0 + \lambda\mathbf{y}_0$, we get a polynomial whose functional degree is at most d. Since $|\mathbf{F}| > nd$, and $deg(F) \leq nd$, we conclude, that the degree of $F(\mathbf{x}_0 + \lambda\mathbf{y}_0)$ agrees with its functional degree:

$$deg(F) = deg_\lambda F(\mathbf{x}_0 + \lambda\mathbf{y}_0) = deg'_\lambda F(\mathbf{x}_0 + \lambda\mathbf{y}_0) = d.$$

□

From the argument we get more.

Lemma 5.14 *Let $|\mathbf{F}| \geq dn$. and let $\mathbf{F}$ be a multi-degree $\mathbf{d}$ polynomial of n variables, but not a total degree $\mathbf{d}$ polynomial. Then the number of $\mathbf{x}, \mathbf{y}$ pairs that $F(\mathbf{x} + \lambda\mathbf{y})$ has degree higher than $\mathbf{d}$ is at least*

$$(|\mathbf{F}| - d)^{2n}. \tag{27}$$

Proof. Notice that in the previous proof $P_{deg(F)}(\mathbf{x}_0,\mathbf{y}_0)$ is a multi-degree d polynomial with $2n$ variables. Thus it has $(|\mathbf{F}| - d)^{2n}$ non-zeros. □

5.1.5 Distances and Probability Measures

Let $A = \{a_1,\ldots,a_n\}$ be an arbitrary finite set. A *probability distribution* or *probability measure* on A is a function p from A to $\mathbf{R}$ such that:

1. $p(a_i) \geq 0$ for $1 \leq i \leq n$.
2. $\sum_{i=1}^{n} p(a_i) = 1$.

Definition 5.15 (DISTR) *Let A be a finite set. We denote the set all probability distributions on A by $DISTR(A)$.*

The uniform and the delta distributions are the simplest members of $DISTR(A)$, but we will also need to define compound distributions such as the product distribution and convex combination of distributions.

Uniform Distribution: Let $|A| = n$. The function which assigns $1/n$ to to every element of A is called the uniform distribution on A and is denoted by U_A.

Delta Distribution: Let A be a finite set and $a \in A$. The distribution which assigns 1 to a and 0 to every other element of A is a delta distribution and is denoted by I_a.

Product Distribution: Let $A_1, \ldots, A_k$ be finite sets and p_i be a distribution on A_i for $1 \leq i \leq k$. Then the distribution which assigns $\prod_{i=1}^{k} p_i(a_i)$ to $(a_1, \ldots, a_k) \in A_1 \times \cdots \times A_k$ is called the product of distributions p_i $(1 \leq i \leq k)$, and is denoted by $p_1 \times \cdots \times p_k$.

Convex Combination of Distributions: Let $A_1, \ldots, A_k$ be finite sets, let p_i be a distribution on A_i for $1 \leq i \leq k$ and p be a distribution on $\{1, \ldots, k\}$. Then the distribution which assigns $\sum_{a \in A_i} p(i)p_i(a)$ for any $a \in A_1 \cup \ldots \cup A_k$ is called the convex combination of p_i $(1 \leq i \leq k)$ according to p and is denoted by $\sum_{i=1}^{k} p(i)p_i$.

Definition 5.16 *The support of a distribution p on A is*

$$supp(p) = \{a \in A \mid p(a) \neq 0\}.$$

Let Σ be a finite alphabet and $\mathbf{v}, \mathbf{w} \in \Sigma^n$. (The elements of Σ^n are called *vectors,* but they do not form a vector space unless Σ is a field.) The *Hamming distance* in between $\mathbf{v}$ and $\mathbf{w}$, $Dist(\mathbf{v},\mathbf{w})$, is the number of coordinates in which $\mathbf{v}$ and $\mathbf{w}$ differ. We get the *normalized* Hamming distance by dividing the Hamming distance by n. The normalized Hamming distance is always in between zero and one. The following definition provides a generalization of this notion.

Definition 5.17 *Let p be a probability measure on $A = \{a_1, \ldots, a_n\}$, Σ be a finite alphabet and $\mathbf{v}, \mathbf{w} \in \Sigma^A$. The distance in between $\mathbf{v}$ and $\mathbf{w}$ according to p is defined by*

$$dist_p(\mathbf{v}, \mathbf{w}) = \sum_{v_{a_i} \neq w_{a_i}} p(a_i).$$

If the subscript p is omitted, it automatically means that p is uniform, in which case our distance notion coincides with the normalized Hamming distance.

Notice that the distance of two vectors according to any probability distribution is a real number in between zero and one.

5.1.6 Lines and Planes

Some low degree subsets of $\mathbf{F}^n$ will be very useful in constructing checkable polynomial codes. In this section we discuss lines and plains. Arbitrary parameterized varieties will be discussed in the next section.

Lines: Lines are one dimensional affine subspaces of $\mathbf{F}^n$. To denote the elements of $\Lambda(\mathbf{F}^n)$, the set of all lines of $\mathbf{F}^n$, we may use two different indexing.

Affine Indexing: $\Lambda(\mathbf{F}^n) = \{\mathcal{L}_{\mathbf{x},\mathbf{y}} \mid \mathbf{x} \in \mathbf{F}^n, \mathbf{0} \neq \mathbf{y} \in \mathbf{F}^n\}$, where $\mathcal{L}_{\mathbf{x},\mathbf{y}} = \{\mathbf{x} + \lambda\mathbf{y} \mid \lambda \in \mathbf{F}\}$. In case $\mathbf{y} = \mathbf{0}$, $\mathcal{L}_{\mathbf{x},\mathbf{y}}$ is a single point, $\mathbf{x}$. $\mathcal{L}_{\mathbf{x},\mathbf{y}} = \mathcal{L}_{\mathbf{x}',\mathbf{y}'}$ iff $\mathbf{x}' = \mathbf{x} + \mu\mathbf{y}$, $\mathbf{y}' = \nu\mathbf{y}$, where $\mu, \nu \neq 0 \in \mathbf{F}$. The values of μ and ν are determined by $\mathbf{x}, \mathbf{x}', \mathbf{y}, \mathbf{y}'$.

Projective indexing: Let $DIR(\mathbf{F}^n)$ be the set of all directions in $\mathbf{F}^n$. $|DIR(\mathbf{F}^n)| = \frac{|\mathbf{F}|^n - 1}{|\mathbf{F}| - 1}$. For a fixed direction $\delta \in DIR(\mathbf{F}^n)$ there are $|\mathbf{F}|^{n-1}$ parallel lines. Let $CLASS(\delta)$ be the collection of all lines in direction δ, and let us index its elements with the elements of $\mathbf{F}^{n-1}$ in any standard way. Let $\mathcal{L}_{\delta,\mathbf{x}} \in CLASS(\delta)$ be the member with $\mathbf{x} \in \mathbf{F}^{n-1}$. Note that $|\Lambda(\mathbf{F}^n)| = |\mathbf{F}|^{n-1}\frac{|\mathbf{F}|^n - 1}{|\mathbf{F}| - 1}$.

Planes: Planes are two dimensional affine subspaces of $\mathbf{F}^n$. To denote the elements of $\Pi(\mathbf{F}^n)$, the set of all planes of $\mathbf{F}^n$, we may use two different indexing.

Affine Indexing: $\Pi(\mathbf{F}^n) = \{\mathcal{P}_{\mathbf{x},\mathbf{y},\mathbf{z}} \mid \mathbf{x}, \mathbf{y}, \mathbf{z} \in \mathbf{F}^n\} \setminus (\Lambda(\mathbf{F}^n) \cup \mathbf{F}^n)$, where $\mathcal{P}_{\mathbf{x},\mathbf{y},\mathbf{z}} = \{\mathbf{x} + \lambda\mathbf{x} + \mu\mathbf{z} \mid \lambda, \mu \in \mathbf{F}\}$. In this indexing every element of $\mathbf{F}^n$ is represented uniquely, every element of $\Lambda(\mathbf{F}^n)$ is represented $|\mathbf{F}|(|\mathbf{F}|^2 - 1)$ times and every element of $\Pi(\mathbf{F}^n)$ is represented $|\mathbf{F}|(|\mathbf{F}|^2 - 1)(|\mathbf{F}|^2 - |\mathbf{F}|)$ times.

Projective indexing: We have $\frac{(|\mathbf{F}|^n - 1)(|\mathbf{F}|^n - |\mathbf{F}|)}{(|\mathbf{F}|^2 - 1)(|\mathbf{F}|^2 - |\mathbf{F}|)}$ different directions (i.e. parallel classes) for planes. Let us denote the set of these classes by $PDIR(F)$. For every $\pi \in PDIR(\mathbf{F})$ let $CLASS(\pi)$ be the set of all parallel planes in direction π. We can index the elements of $CLASS(\pi)$ with the elements of $\mathbf{F}^{n-2}$. Let $\mathcal{P}_{\pi,\mathbf{x}}$ be the plane in $CLASS(\pi)$ whose index is $\mathbf{x} \in \mathbf{F}^{n-2}$. We remark that $|\Pi(\mathbf{F}^n)| = |\mathbf{F}|^{n-2}\frac{(|\mathbf{F}|^n - 1)(|\mathbf{F}|^n - |\mathbf{F}|)}{(|\mathbf{F}|^2 - 1)(|\mathbf{F}|^2 - |\mathbf{F}|)}$

We shall define probability distributions on $\mathbf{F}^n$ that are concentrated on lines and planes.

Lemma 5.18 *Consider the following distributions on* $\Lambda(\mathbf{F}^n) \cup \mathbf{F}^n$*:*

$$D_1 = U_{\Lambda(\mathbf{F}^n)} = \frac{1}{|\Lambda(\mathbf{F}^n)|} \sum_{\delta \in DIR(\mathbf{F}^n), \mathbf{x} \in \mathbf{F}^{n-1}} I_{\mathcal{L}_{\delta,\mathbf{x}}}$$

$$D_2 = \frac{1}{|\mathbf{F}|^{2n}} \sum_{\mathbf{x},\mathbf{y} \in \mathbf{F}^n} I_{\mathcal{L}_{\mathbf{x},\mathbf{y}}}.$$

Then the L_1 *norm of* $D_1 - D_2$ *is* $2/|\mathbf{F}|^n$.

Lemma 5.19 *Consider the following distributions on* $\Pi(\mathbf{F}^n) \cup \Lambda(\mathbf{F}^n) \cup \mathbf{F}^n$*:*

$$D_1 = U_{\Pi(\mathbf{F}^n)} = \frac{1}{|\Pi(\mathbf{F}^n)|} \sum_{\pi \in PDIR(\mathbf{F}^n), \mathbf{x} \in \mathbf{F}^{n-2}} I_{\mathcal{P}_{\pi,\mathbf{x}}}$$

$$D_2 = \frac{1}{|\mathbf{F}|^{3n}} \sum_{\mathbf{x},\mathbf{y},\mathbf{z} \in \mathbf{F}^n} I_{\mathcal{P}_{\mathbf{x},\mathbf{y},\mathbf{z}}}.$$

Then the L_1 *norm of* $D_1 - D_2$ *is less than* $2(|\mathbf{F}| + 1)/|\mathbf{F}|^n$.

We interpret the above two lemmas that it essentially does not matter whether we select a random line (plane) uniformly from all lines (planes) or whether we select it by picking a random pair (triplet) of affine parameters. The next lemma says that random line of a random plane that goes through a fixed point of $\mathbf{F}^n$ is essentially random among all lines.

Lemma 5.20 *Let* $\mathbf{x}_0 \in \mathbf{F}^n$. *Let* D *be the distribution on* $\Lambda(\mathbf{F}^n)$ *that we obtain by picking a random plane* Π_π *among all the planes that contain* $\mathbf{x}_0$*, and after that a random element of* $\Lambda(\Pi_\pi)$*, (i.e. among all lines of* Π_π*). Then the* L_1 *distance of* D *from* $U_{\Lambda(\mathbf{F}^n)}$ *(the uniform distribution on the lines of* $\mathbf{F}^n$*) is less than* $2/|\mathbf{F}|$.

5.1.7 Parameterized Varieties

Let $y_1, \ldots, y_k$ be a set of parameters, each ranging in $\mathbf{F}$, and $P_1, \ldots, P_n$ be k-variate polynomials of multi-degree D. The set $\{(P_1(\mathbf{y}), \ldots, P_n(\mathbf{y})) \mid \mathbf{y} \in \mathbf{F}^k\}$ is a multi-degree D parameterized variety of dimension k in $\mathbf{F}^n$.

Lemma 5.21 *Let* P *be an* n*-variate polynomial of total degree* d *and* $\mathcal{V}$ *be a parameterized variety of dimension* k *and multi-degree* D*. Then* $P \circ \mathcal{V}$ *is a multi-degree at most* dD *polynomial of* k *variables.*

In what follows, we fix an arbitrary $X \subseteq \mathbf{F}$ with $0 \in X$ and $|X| = D+1$. Let $f : X^k \setminus \{0\} \to \mathbf{F}^n$, and for $\mathbf{z} \in \mathbf{F}^n$ let $f_{\mathbf{z}}$ be an extension of f to the domain X^k such that $f_{\mathbf{z}}(0) = \mathbf{z}$. If we let $\mathbf{z} \in \mathbf{F}^n$ run, we get a family of functions from X^k to $\mathbf{F}^n$. It is easy to see by Lemmas 5.10 and 5.11, that $f_{\mathbf{z}}$ uniquely extends to a parameterized variety $\mathcal{V}_{\mathbf{z}} : \mathbf{F}^k \to \mathbf{F}^n$ of $\mathbf{F}^n$ of multi-degree D. We call $\{\mathcal{V}_{\mathbf{z}}\}_{\mathbf{z} \in \mathbf{F}^n}$ a *standard family* of parameterized varieties.

We would like to embed arbitrary subsets of $\mathbf{F}^n$ into parameterized varieties. Given a set $H \subseteq \mathbf{F}^n$ with $|H| < (D+1)^k$ we can find a standard family of parameterized varieties of multi-degree D and dimension k, such that H is a subset of each members of the family.

This is relevant for the following reason: If P is an n-variate low degree polynomial over $\mathbf{F}$, then the restriction of P on H can be decoded from the restriction of P on any member of this variety even in the presence of noise. If the entire P is given with a small percentage of error, we will find, that most of the varieties will be sufficiently noise-free in order to decode the restriction of P on H from them. In particular, a randomly chosen member will likely work. This decoding procedure is a generalization of decoding the value of P just over a single point of $\mathbf{F}^n$ by casting a random line through the point [24]. In the rest of the section we give lemmas that establish these useful properties of the standard families of varieties. For a variety $\mathcal{V}$ of dimension k, and functions $F, G : \mathbf{F}^n \to \mathbf{F}$ we use the notation

$$dist_{\mathcal{V}}(F, G) = \frac{1}{|\mathbf{F}|^k} \sum_{F(\mathcal{V}(\mathbf{x})) \neq G(\mathcal{V}(\mathbf{x}))} \mathbf{1}.$$

Lemma 5.22 *Let $\mathcal{V}$ be an arbitrary parameterized variety with dimension k and multi-degree D. Let $P, Q \in \mathbf{F}^{deg<d}[\mathbf{x}]$ such that P and Q differ over at least one point of $\mathcal{V}$. Then*

$$dist_{\mathcal{V}}(P, Q) \geq \left(\frac{|\mathbf{F}| - Dd}{|\mathbf{F}|}\right)^k$$

Proof. We need to use Lemma 5.12 for $P \circ \mathcal{V}$ and $Q \circ \mathcal{V}$, that have multi-degree at most Dd by Lemma 5.21. □

Lemma 5.23 *Let $\{\mathcal{V}_{\mathbf{z}}\}_{\mathbf{z} \in \mathbf{F}^n}$ be a standard family of parameterized varieties in $\mathbf{F}^n$ and $\mathbf{x}_0 \in (\mathbf{F} \setminus (X \setminus \{0\}))^k$. Then $\mathbf{z} \to \mathcal{V}_{\mathbf{z}}(\mathbf{x}_0)$ is a one-one map from $\mathbf{F}^n$ to $\mathbf{F}^n$.*

Proof. Let $X = \{0, a_1, \ldots, a_D\}$. $\mathcal{V}_{\mathbf{z}} = \mathcal{V}_{\mathbf{0}} + \mathcal{V}$, where $\mathcal{V}$ is defined by

$$P_i(\mathbf{y}) = z_i \frac{\prod_{1 \le j \le k} \prod_{1 \le l \le D} (y_j - a_l)}{\prod_{1 \le j \le k} \prod_{1 \le l \le D} (-a_l)} \quad \text{for } 1 \le i \le n. \tag{28}$$

If we replace $\mathbf{y}$ with any $\mathbf{x}_0 \in (\mathbf{F} \setminus (X \setminus \{0\}))^k$, we can uniquely solve (28) in $\mathbf{z}$ for any setting of the left hand side.

Lemma 5.24 *Let $\{\mathcal{V}_{\mathbf{z}}\}_{\mathbf{z} \in \mathbf{F}^n}$ be a standard family of parameterized varieties in $\mathbf{F}^n$ of dimension k and degree D. Then for any $F, G : \mathbf{F}^n \to \mathbf{F}$ we have:*

$$\mathbf{E}_{\mathbf{z} \in \mathbf{F}^n}(dist_{\mathcal{V}_{\mathbf{z}}}(F, G)) \le dist(F, G) + \frac{|\mathbf{F}^k| - (|\mathbf{F}| - D)^k}{|\mathbf{F}^k|} \tag{29}$$

Proof. Let $H = \{\mathbf{x} \in \mathbf{F}^n \mid F(\mathbf{x}) \neq G(\mathbf{x})\}$. By definition the measure of H is exactly *dist*(F, G). Clearly,

$$\mathbf{E}_{\mathbf{z} \in \mathbf{F}^n}(dist_{\mathcal{V}_{\mathbf{z}}}(F, G)) = \frac{1}{|\mathbf{F}^n|} \sum_{\mathbf{z} \in \mathbf{F}^n} \sum_{\mathbf{x} \in H} \frac{1}{|\mathbf{F}^k|} \sum_{\mathbf{y} \in \mathbf{F}^k, \mathcal{V}_{\mathbf{z}}(\mathbf{y}) = \mathbf{x}} 1 =$$

$$\frac{1}{|\mathbf{F}|^{n+k}} \sum_{\mathbf{z} \in \mathbf{F}^n} \sum_{\mathbf{x} \in H} \left(\sum_{\mathbf{y} \in (\mathbf{F} \setminus (X \setminus \{0\}))^k, \mathcal{V}_{\mathbf{z}}(\mathbf{y}) = \mathbf{x}} 1 \right) + \tag{30}$$

$$\frac{1}{|\mathbf{F}|^{n+k}} \sum_{\mathbf{z} \in \mathbf{F}^n} \sum_{\mathbf{x} \in H} \left(\sum_{\mathbf{y} \in \mathbf{F}^k \setminus (\mathbf{F} \setminus (X \setminus \{0\}))^k, \mathcal{V}_{\mathbf{z}}(\mathbf{y}) = \mathbf{x}} 1 \right) \tag{31}$$

Term (31) can simply be estimated from above by $\frac{|\mathbf{F}^k| - (|\mathbf{F}| - D)^k}{|\mathbf{F}^k|}$ and term (30) can be estimated from above by

$$\frac{1}{|\mathbf{F}|^{n+k}} \sum_{\mathbf{x} \in H} \left(\sum_{\mathbf{y} \in (\mathbf{F} \setminus (X \setminus \{0\}))^k} 1 \right) = dist(F, G) \frac{|\mathbf{F} \setminus (X \setminus \{0\})|^k}{\mathbf{F}^k} \le dist(F, G).$$

□

5.1.8 Linear Codes

Let $\mathbf{F}$ be a field. For a vector $\mathbf{w} \in \mathbf{F}^n$ we define $weight(w)$ as the number of non-zero entries of $\mathbf{w}$. If $\mathbf{w_1}$ and $\mathbf{w_2}$ belong to the same vector space over

$\mathbf{F}$, then $Dist(\mathbf{w_1}, \mathbf{w_2})$ is by definition the weight of $\mathbf{w_1} - \mathbf{w_2}$. The distance function is symmetric, since $weight(\mathbf{w_1} - \mathbf{w_2}) = weight(\mathbf{w_2} - \mathbf{w_1})$. A linear subspace C of F^n is also called a *linear code.* In practice we call a linear subspace C of $\mathbf{F}^n$ a code if special attention is payed to the quantity:

$$d(C) = \min_{\mathbf{w} \in C, \mathbf{w} \neq 0} weight(\mathbf{w}).$$

Note that because of the linearity, $d(C)$ is also a lower bound for the distance in between any two elements of the code. The parameters of $C \subseteq \mathbf{F}^n$ are its length, n, its dimension, k, and its minimum distance, $d(C)$, for which we use the shorthand d. With the above parameters C is called an (n, k, d) code. An element of an (n, k, d) code encodes $k \log |\mathbf{F}|$ bits of information.

We can construct an (n, k, d) code C either via a k times n matrix M such that $C = \{\mathbf{x}M \mid \mathbf{x} \in \mathbf{F}^k\}$ or via an n times l matrix P, where $l \geq n - k$ such that $C = \{\mathbf{w} \in \mathbf{F}^n \mid P\mathbf{w}^T = 0\}$. The matrix M is called the matrix of the code, and the matrix P is called the parity-check matrix of the code, and they are not unique. A family of codes is efficiently given if their matrices and their parity check matrices have a polynomial time description in terms of the length of the members.

Proposition 5.25 *There is a constant $c > 0$ such that for every $k > 0$ there are $(c'k, k, c'k/4)$ codes where $c' \leq c$ It also holds that these codes are efficiently constructible.*

5.2 Checkable Codes

In PCP theory the task of checking often specializes to checking linear codes. When we check a code we have a system of the subsets of the index set from which we choose a random element, and make a yes/no decision on the basis of what we see in these positions. The evaluation of the entries the checker sees is called a *view.* An *accepting view* is what the checker accepts, and a *rejecting view* is what the checker rejects. According to the code checking model we need to accept all words of the code with probability one (or close to one, depending on the model), and reject with high probability all words that are far from any code word. A code is usually called checkable if there is a checking scheme such that the number of entries we need to look at is significantly smaller than the dimension (not the length!) of the code.

Consider a negative example: recall that the usual Reed Solomon codes are tables for uni-variate polynomials over a field $\mathbf{F}$. We encode a polynomial

$P \in \mathbf{F}^d[\mathbf{x}]$ into the vector of its replacement values: $P \to (P(a_1), \ldots, P(a_n))$ where $a_1, \ldots, a_n$ are distinct elements of $\mathbf{F}$. The Reed Solomon codes have the property that for an arbitrary set of entries $i_1 < \ldots < i_{d+1}$ and for any view $b_1, \ldots, b_{d+1}$ on these entries there is a code word which has value b_j at entry i_j for $1 \leq j \leq d+1$. This holds because there is always a degree d polynomial such that $P(a_{i_j}) = b_j$ for $1 \leq j \leq d+1$. Therefore either the size of some check sets should be greater than $d+1$, or we also need to reject views that come from a code word.

We say that a checking procedure has the *perfect soundness property* if we do not reject any view that can come from a codeword. In the case of perfectly sound checking procedures the set of entries we look at (for a fixed random choice) uniquely determines the accepting views: they are exactly those that can occur as a restriction of some codeword on the set of entries in question. Although in the advanced theory of PCP codes with imperfect soundness play a very important role in achieving the best parameters (see Håstad [84, 85]), they are not linear. Here we restrict ourselves only to linear codes and to checking procedures with perfect soundness.

One of the first known checkable codes was the code of multi-linear functions. Babai, Fortnow, and Lund [22] used this property of multi-linear codes to prove that $MIP \neq NEXP$. Another checkable code was provided by Blum, Luby, and Rubinfeld for an entirely different application [42]. The results of Feige, Goldwasser, Lovász, Safra, and Szegedy, [62], Babai, Fortnow, Levin, and Szegedy [23], and Arora and Safra [12] use the code of multi-variate polynomials with restriction on the multi-degree. Arora et.al. [10] uses the code of multi-variate polynomials with restriction on the total degree (See Section 5.2.3) and also the code of Blum, Luby, and Rubinfeld (see next section).

In the above results code checking was not differentiated from checking in general. Indeed there is not a definite distinction in between checking linear codes and other sets of strings. Nevertheless, because of the relation in between different checkable linear codes it makes sense to collect the code checking results into a single section. We also introduce a slightly different way to measure the success of the checking procedure for codes. Our new measure is the expected distance of the received view to a view that may come from a code word. We request that if the checked word is outside the neighborhood, this expected distance be large. This definition, of course, could be generalized to any collection of strings, which is not necessarily a linear code.

Code checking has four interesting parameters: the radius parameter for

the neighborhood we want to check, the measure of success with which we can detect if a string is outside the neighborhood, the maximum number of entries we need to look at, and the size of the assembly from which we choose our check set. Here is the definition of code checking we shall use throughout the next sections.

Definition 5.26 *The ϵ-neighborhood of $\mathbf{x} \in \mathbf{F}^n$ is $\{\mathbf{y} \mid dist(\mathbf{x}, \mathbf{y}) \leq \epsilon\}$. The ϵ-neighborhood of a code C is the union of the ϵ-neighborhoods of its members.*

Definition 5.27 (Code Checking) *Let $\mathbf{F}$ be a field. The ϵ-neighborhood of a code $C \leq \mathbf{F}^n$ is δ-checkable with check-size g if there are $W_1, \ldots, W_m \in DISTR(\{1, \ldots, n\})$ such that $|supp(W_i)| \leq g$ for $1 \leq i \leq m$ and $Q \in DIST(\{1, \ldots, m\})$ such that for every $\mathbf{v} \in \mathbf{F}^n$ outside the ϵ-neighborhood of C we have:*

$$\mathbf{E}_Q \left(\min_{z \in C} dist_{W_i}(z, v) \right) \geq \delta \tag{32}$$

If the ϵ-neighborhood of a family of codes $\mathcal{C}$ is δ-checkable, where $\epsilon > 0$ and $\delta > 0$ are constants, and the check-size is at most poly-logarithmic, we informally say, that the "membership in any $C \in \mathcal{C}$ is checkable."

5.2.1 Checking the Hadamard Code

The simplest and most fundamental checkable code over a finite field $\mathbf{F}$ is the (generalized) Hadamard code. It has dimension k, length $|\mathbf{F}|^k$, and its matrix $M_{hm(\mathbf{F},k)}$ is a k by $|\mathbf{F}|^k$ matrix with all possible columns of length k that can be made out of the elements of $\mathbf{F}$. The minimum distance of the Hadamard code is $|\mathbf{F}|^k(1 - 1/|\mathbf{F}|)$.

Here we discuss only the case when the field is $\mathbf{F}_2$. The code-length in this case is 2^n, the dimension is n and the distance of any two distinct code-words is 2^{n-1}. Entries are indexed with vectors of length n over $\mathbf{F}_2$. The value that appears in the entry indexed with $\mathbf{x}$, when we encode $\mathbf{y} \in \mathbf{F}_2^n$ is $\sum_{i=1}^n x_i y_i$, the inner product of $\mathbf{x}$ and $\mathbf{y}$ in $\mathbf{F}_2$. For any code-word $\mathbf{z}$ of the Hadamard code and for any $\mathbf{x}_1, \mathbf{x}_2 \in \mathbf{F}_2^n$ we have the equation:

$$\mathbf{z}_{\mathbf{x}_1} + \mathbf{z}_{\mathbf{x}_2} = \mathbf{z}_{\mathbf{x}_1 + \mathbf{x}_2}.$$

For this reason the set of entries $\{\mathbf{x}_1, \mathbf{x}_2, \mathbf{x}_1 + \mathbf{x}_2\}$ is an appropriate check set for the Hadamard code, since only 4 out of 8 views can be accepted. Using these check sets Blum, Luby and Rubinfeld [42] have shown that the

Hadamard code is efficiently checkable, although the theorem we present here is slightly sharper and due to

Theorem 5.28 ([42, 30]) *Let* $\mathbf{z} \in \mathbf{F}_2^{\mathbf{F}_2^n}$. *If the number of* $\mathbf{x}_1, \mathbf{x}_2 \in \mathbf{F}_2^n$ *ordered pairs for which* $\mathbf{v}_{\mathbf{x}_1} + \mathbf{v}_{\mathbf{x}_2} \neq \mathbf{v}_{\mathbf{x}_1+\mathbf{x}_2}$ *is at most* $\delta 2^{2n}$, *then* $\mathbf{v} \in \mathbf{F}_2^n$ *belongs to the* δ*-neighborhood of the Hadamard code.*

The most elegant proof of Theorem 5.28, which is due to M. Bellare, D. Coppersmith, J. Håstad, M. Kiwi and M. Sudan [30], uses Fourier-transform techniques. Let us map $\mathbf{F}_2 = \{0,1\}$ to $\{1,-1\} \subseteq \mathbf{R}$ such that 0 goes to 1 and 1 goes to -1. For a vector $\mathbf{v}$ with elements from $\mathbf{F}_2$ let v denote its real counterpart, that we obtain by applying the above map for all coordinates of $\mathbf{v}$. For $v \in \{1,-1\}^{\mathbf{F}_2^n}$ we denote the entry indexed by $\mathbf{x} \in \mathbf{F}_2^n$ with $v[\mathbf{x}]$. Thus for any $\mathbf{v} \in \mathbf{F}_2^{\mathbf{F}_2^n}$ the number of $\mathbf{x}_1, \mathbf{x}_2$ pairs such that $\mathbf{v}_{\mathbf{x}_1} + \mathbf{v}_{\mathbf{x}_2} \neq \mathbf{v}_{\mathbf{x}_1+\mathbf{x}_2}$ is exactly the number of $\mathbf{x}_1, \mathbf{x}_2$ pairs for which $v[\mathbf{x}_1]v[\mathbf{x}_2]v[\mathbf{x}_1 + \mathbf{x}_2] = -1$. Since this product is always either 1 or -1, the theorem we need to prove is equivalent to stating that

$$\delta = \frac{1}{2^{2n}}|\{(\mathbf{x}_1, \mathbf{x}_2) \mid v[\mathbf{x}_1]v[\mathbf{x}_2]v[\mathbf{x}_1 + \mathbf{x}_2] = -1\}| = \frac{1}{2} - \frac{1}{2^{1+2n}} \sum_{\mathbf{x}_1, \mathbf{x}_2 \in F_2^n} v[\mathbf{x}_1]v[\mathbf{x}_2]v[\mathbf{x}_1 + \mathbf{x}_2] \tag{33}$$

implies that the normalized Hamming distance of $\mathbf{v}$ from the closest member of the Hadamard code is at most δ. The normalized Hamming distance of $v_1 \in \{0,1\}^{\mathbf{F}_2^n}$ and $v_2 \in \{0,1\}^{\mathbf{F}_2^n}$ can be conveniently expressed as $dist(v_1, v_2) = \frac{1}{2} - \frac{1}{2^{n+1}}\langle v_1, v_2 \rangle$, where $\langle ., . \rangle$ denotes the usual scalar product of two real vectors. Let $\{w^{\mathbf{a}}\}_{\mathbf{a} \in \mathbf{F}_2^n}$ be the elements of the Hadamard code in the real representation, where $w^{\mathbf{a}}[\mathbf{x}] = 1$ if $\sum_{i=1}^n a_i x_i = 0$ in $\mathbf{F}_2$, and $w^{\mathbf{a}}[\mathbf{x}] = -1$ otherwise.

The vectors $\{w^{\mathbf{a}}\}_{\mathbf{a} \in \mathbf{F}^n}$ form an orthogonal basis for $\mathbf{R}^{\mathbf{F}_2^n}$. (Although most literature do, here we do not normalize the basis, nor the scalar product.) The orthogonality can be readily seen from the fact that two different linear forms over $\mathbf{F}_2$ with no constant term agree for exactly half of the replacement values.

The Fourier expansion for an arbitrary $v \in \mathbf{R}^{\mathbf{F}_2^n}$ is the expression $v = \sum_{\mathbf{a} \in \mathbf{F}_2^n} \lambda_{\mathbf{a}} w^{\mathbf{a}}$. The elements of the sequence $\{\lambda_{\mathbf{a}}\}_{\mathbf{a} \in \mathbf{F}_2^n}$ are called the Fourier coefficients of v. From $\langle w^{\mathbf{a}}, v \rangle = \lambda_{\mathbf{a}} \langle w^{\mathbf{a}}, w^{\mathbf{a}} \rangle = 2^n \lambda_{\mathbf{a}}$ we obtain:

$$dist(w^{\mathbf{a}}, v) = \frac{1}{2} - \frac{1}{2}\lambda_{\mathbf{a}}. \tag{34}$$

The orthogonality relations for the $w^{\mathbf{a}}$s also imply that for any $v \in \{-1,1\}^{\mathbf{F}_2^n}$

$$1 = \frac{1}{2^n}\langle v, v\rangle = \frac{1}{2^n}\sum_{\mathbf{a},\mathbf{b}\in\mathbf{F}_2^n} \lambda_{\mathbf{a}}\lambda_{\mathbf{b}}\langle w_a, w_b\rangle = \sum_{\mathbf{a}\in\mathbf{F}_2^n} \lambda_{\mathbf{a}}^2. \tag{35}$$

Let us now rewrite $\sum_{\mathbf{x}_1,\mathbf{x}_2\in F_2^n} v[\mathbf{x}_1]v[\mathbf{x}_2]v[\mathbf{x}_1+\mathbf{x}_2]$ using the Fourier expansion of v as

$$\sum_{\mathbf{x}_1,\mathbf{x}_2\in\mathbf{F}_2^n} (\sum_{\mathbf{a}\in\mathbf{F}_2^n} \lambda_{\mathbf{a}} w^{\mathbf{a}}[\mathbf{x}_1])(\sum_{\mathbf{b}\in\mathbf{F}_2^n} \lambda_{\mathbf{b}} w^{\mathbf{b}}[\mathbf{x}_2])(\sum_{\mathbf{c}\in\mathbf{F}_2^n} \lambda_{\mathbf{c}} w^{\mathbf{c}}[\mathbf{x}_1+\mathbf{x}_2]). \tag{36}$$

It will turn out that that the later formula simplifies to $2^{2n}\sum_{\mathbf{a}\in\mathbf{F}_2^n}\lambda_{\mathbf{a}}^3$. Indeed, rearranging the sum we get that (36) equals to

$$\sum_{\mathbf{a}\in\mathbf{F}_2^n}\sum_{\mathbf{b}\in\mathbf{F}_2^n}\sum_{\mathbf{c}\in\mathbf{F}_2^n} \lambda_{\mathbf{a}}\lambda_{\mathbf{b}}\lambda_{\mathbf{c}} \sum_{\mathbf{x}_1,\mathbf{x}_2\in\mathbf{F}_2^n} w^{\mathbf{a}}[\mathbf{x}_1]w^{\mathbf{b}}[\mathbf{x}_2]w^{\mathbf{c}}[\mathbf{x}_1+\mathbf{x}_2]. \tag{37}$$

Because of $w^{\mathbf{c}}[\mathbf{x}_1+\mathbf{x}_2] = w^{\mathbf{c}}[\mathbf{x}_1]w^{\mathbf{c}}[\mathbf{x}_2]$ for fixed **a, b** and **c,** the expression $\sum_{\mathbf{x}_1,\mathbf{x}_2\in\mathbf{F}_2^n} w^{\mathbf{a}}[\mathbf{x}_1]w^{\mathbf{b}}[\mathbf{x}_2]w^{\mathbf{c}}[\mathbf{x}_1+\mathbf{x}_2]$ becomes

$$\sum_{\mathbf{x}_1\in\mathbf{F}_2^n} w^{\mathbf{a}}[\mathbf{x}_1]w^{\mathbf{c}}[\mathbf{x}_1] \sum_{\mathbf{x}_2\in\mathbf{F}_2^n} w^{\mathbf{b}}[\mathbf{x}_2]w^{\mathbf{c}}[\mathbf{x}_2] = \langle w^{\mathbf{a}}, w^{\mathbf{c}}\rangle\langle w^{\mathbf{b}}, w^{\mathbf{c}}\rangle. \tag{38}$$

The right hand side of (38) is 2^{2n} if $\mathbf{a} = \mathbf{b} = \mathbf{c}$ and 0 otherwise. We get the simplified expression for (36) as claimed. From this (33) can be expressed as $\frac{1}{2} - \frac{1}{2}\sum_{\mathbf{a}\in\mathbf{F}_2^n}\lambda_{\mathbf{a}}^3$. Let $w^{\max}$ be an element of the Fourier basis whose coefficient, $\lambda_{\max}$, has maximum value among all Fourier coefficients of v (in formula $\lambda_{\max} = \max_{\mathbf{a}\in\mathbf{F}_2^n}\lambda_{\mathbf{a}}$). We claim that the distance in between v and $w^{\max}$ is at most δ. Indeed, from (34) and (35):

$$\delta = \frac{1}{2} - \frac{1}{2}\sum_{\mathbf{a}\in\mathbf{F}_2^n}\lambda_{\mathbf{a}}^3 \geq \frac{1}{2} - \frac{1}{2}\sum_{\mathbf{a}\in\mathbf{F}_2^n}\lambda_{\mathbf{a}}^2\lambda_{\max} = \frac{1}{2} - \frac{1}{2}\lambda_{\max} = dist(v, w^{\max}).$$

□

5.2.2 Checking Bivariate Low Degree Polynomials

We test low degree bivariate polynomials by casting a random line in a random direction or by casting a random line in one of the two coordinate

directions depending on whether we want to test the space of total degree d polynomials or the space of multi-degree d polynomials. The success of these tests relies on Theorems 5.29 and 5.30. In the sequel we denote the set of lines in $\mathbf{F}^2$ by Λ and the set of those lines that go through a fixed point $\mathbf{v}$ of $\mathbf{F}^2$ by $\Lambda_{\mathbf{v}}$.

Theorem 5.29 (Random Line Test, Bivariate [10]) *Let* $\mathbf{F}$ *be a finite field,* $F(x,y)$ *be an arbitrary function from* $\mathbf{F}^2$ *to* $\mathbf{F}$, $0 \leq d \leq |\mathbf{F}|/10$ *an integer, and* $0 \leq \delta \leq 1/6$. *For a line* $\mathcal{L}$ *of* $\mathbf{F}^2$ *let* $a(\mathcal{L})$ *denote the normalized Hamming distance of F from the closest degree* d *uni-variate polynomial on the line* $\mathcal{L}$. *If* $\mathbf{E}_{\mathcal{L}\in\Lambda}(a(\mathcal{L})) < \delta$ *then*

$$\min_{deg(G)\leq d} dist(F,G) < 2\delta.$$

The proof of Theorem 5.29 is based on the Bivariate Testing Theorem (BTT) below, which also serves as the basis for testing polynomials in higher dimensions. The BTT roughly states that testing for approximate membership in $\mathbf{F}^{d,d}(x,y)$ can be done by looking at the table of F over a random line in the first coordinate direction and a over a random line in the second coordinate direction. The theorem first appears in the paper of Arora and Safra [12], but there the size of the field depends quadratically on the degree. Although this would be sufficient for our purposes, we show a linear dependence following the lead of Polischuk and Spielman [120]. In this section we take the freedom of calling the multi-degree of a two-variate polynomial as "degree," as long as the term is followed by a pair of numbers.

Theorem 5.30 (Bivariate Testing Theorem [12, 120]) *Let* $\mathbf{F}$ *be a field and let* $X = \{a_1, \ldots, a_n\} \subseteq \mathbf{F}$, $Y = \{b_1, \ldots, b_n\} \subseteq \mathbf{F}$, $|X| = |Y| = n$. *Let* $C(x,y)$ *be a polynomial of degree* (d,n), *and* $D(x,y)$ *be a polynomial of degree* (n,d). *If*

$$Prob_{(x,y)\in X\times Y}\left(C(x,y) \neq D(x,y)\right) < \delta^2, \tag{39}$$

and $n > 3\delta n + 2d$, *then there exists a polynomial* $F(x,y)$ *of degree* (d,d) *such that*

$$Prob_{(x,y)\in X\times Y}\left(F(x,y) \neq C(x,y) \vee F(x,y) \neq D(x,y)\right) < 2\delta^2. \tag{40}$$

Proof. [Theorem 5.30] Polischuk and Spielman have a beautiful proof of this theorem in [120] using resultants, determinants and their derivatives. We take a different route and calculate dimensions of linear spaces.

Lemma 5.31 *Let V, $W_1, \ldots, W_m$ be linear spaces over a field* $\mathbf{F}$, $\varphi_i : V \to W_i$ $(1 \leq i \leq m)$ *be linear maps. If there are subspaces $V_1, \ldots, V_m \subseteq V$ that:*

1. $\varphi_i(V_i) = W_i$ *for* $1 \leq i \leq m$,
2. $V_i \subseteq \mathrm{Ker}(\varphi_j)$ *for* $1 \leq j < i$,

then $\dim \cap_{i=1}^m \mathrm{Ker}(\varphi_i) + \sum_{i=1}^m \dim W_i = \dim V$.

Proof. Define $\varphi : V \to \bigoplus_{i=1}^m W_i$ by $\varphi(\mathbf{v}) = (\varphi_1(\mathbf{v}), \ldots, \varphi_m(\mathbf{v}))$ for any $v \in V$. First we show that $\mathbf{Im}(\varphi) = \bigoplus_{i=1}^m W_i$.

We need to show that for any $\mathbf{w} = (\mathbf{w}_1, \ldots, \mathbf{w}_m) \in \bigoplus_{i=1}^m W_i$ there is a $\mathbf{v}$ such that $\varphi(\mathbf{v}) = \mathbf{w}$. To produce this $\mathbf{v}$ we recursively match the first i coordinates of $\mathbf{w}$ with $\varphi(\mathbf{v}_i)$, for some $\mathbf{v}_i \in V$ for $0 \leq i \leq m$. Let $\mathbf{v}_0 = 0$. Assume that we have already found a $\mathbf{v}_{i-1}$ such that $\varphi(\mathbf{v}_{i-1}) = (\mathbf{w}_1, \ldots, \mathbf{w}_{i-1}, \mathbf{w}'_i, \ldots, \mathbf{w}'_m)$. By Property 1. there is a $\mathbf{v}'_i \in V_i$ such that $\varphi_i(\mathbf{v}'_i) = \mathbf{w}_i - \mathbf{w}'_i$, so $\varphi(\mathbf{v}'_i + \mathbf{v}_{i-1}) = (\mathbf{w}_1, \ldots, \mathbf{w}_i, \mathbf{w}''_{i+1}, \ldots, \mathbf{w}''_m)$. We set $\mathbf{v}_i = \mathbf{v}'_i + \mathbf{v}_{i-1}$ and $\mathbf{v} = \mathbf{v}_m$. Note that adding $\mathbf{v}'_i$ to $\mathbf{v}_{i-1}$ did not change the first $i - 1$ coordinate of the image under φ by Property 2. We get that the image of φ is the entire $\bigoplus_{i=1}^m W_i$. Clearly, $\mathrm{Ker}(\varphi) = \cap_{i=1}^m \mathrm{Ker}(\varphi_i)$. Combining this with $\dim \mathbf{Im}(\varphi) + \dim \mathrm{Ker}(\varphi) = \dim V$ (see Lemma 5.7) we get:

$$\dim \cap_{i=1}^m \mathrm{Ker}(\varphi_i) + \sum_{i=1}^m \dim W_i = \dim \mathrm{Ker}(\varphi) + \dim \bigoplus_{i=1}^m W_i = \dim V.$$

□

Theorem 5.32 *Let* $\deg A = (a, b)$ *with* $b \geq 1$, $B \in \mathbf{F}^{c,d}[x, y]$, $B \neq 0$, *and assume that gcd(A,B) = 1. If* $A(\lambda_1, y) | B(\lambda_1, y), \ldots, A(\lambda_m, y) | B(\lambda_m, y)$ *for distinct* $\lambda_1, \ldots, \lambda_m$ *then* $m \leq a + c + ad/b$.

Proof. Let $V = \mathbf{F}^{m-1, b+d-1}[x, y]$ and $W_i = \mathbf{F}[y]/A(\lambda_i, y)$ for $1 \leq i \leq m$. Then $\dim W_i = \deg A(\lambda_i, y)$. The map $P(x, y) \to (P(\lambda_i, y) \bmod A(\lambda_i, y))$ is a linear map from V onto W_i. Let φ_i be this map. We show that the conditions of Lemma 5.31 hold for V, W_i, φ_i $(1 \leq i \leq m)$ with

$$V_i = \{(x - \lambda_1) \cdots (x - \lambda_{i-1}) Q \mid Q \in \mathbf{F}^{b-1}(y)\} \quad \text{for } 1 \leq i \leq m.$$

To show Condition 1. let $Q(y)$ be any element of W_i, i.e. some polynomial modulo $A(\lambda_i, y)$. We have $\deg Q \leq b - 1$ and so $Q_1 = Q(y) \prod_{j=1}^{i-1} \frac{x - \lambda_j}{\lambda_i - \lambda_j} \in V_i$.

Clearly, $Q = \varphi_i(Q_1)$. For Condition 2. notice that if $j < i$ then replacing λ_j in any polynomial in V_i gives 0. We now show that

$$W = \{CA + DB \mid C \in \mathbf{F}^{m-a-1,d-1}[x,y];\ D \in \mathbf{F}^{m-c-1,b-1}[x,y]\} \leq \cap_{i=1}^{m} \mathrm{Ker}(\varphi_i).$$

Indeed, $A(\lambda_j, y)$ divides $B(\lambda_j, y)$ for any $1 \leq j \leq m$, therefore it also divides $C(\lambda_j, y)A(\lambda_j, y) + D(\lambda_j, y)B(\lambda_j, y)$. Thus $\varphi_j(AC + BD) = 0$ for $1 \leq j \leq m$.

Let us now calculate the dimensions. Clearly, $\dim V = m(b+d)$, $\dim W_i = b$ for at least $m - a$ of $1 \leq i \leq m$, since in $A = \sum_{i=1}^{a} y^i q_i(x)$ the coefficient $q_a(x)$ cannot be 0 for more than a of the λ_is. Thus by Lemma 5.31 we have that $\dim W \leq m(b+d) - (m-a)b = md + ab$. On the other hand, A and B have no common factor. Because of the y-degree restrictions of C and D there are no (C, D) pairs in the definition of W other than the (0,0) pair for which $CA + DB$ is 0, and so the dimension of W is $(m-a)d+(m-c)b = md+ab+(mb-ab-ad-bc)$ thus $m \leq a+c+ad/b$.□

Remark 5.33 *While the above lemma is not sharp, it is sharp within a factor of 4. Indeed, we prove that for any fixed a, b, c, d for the minimal $m = m(a,b,c,d)$ such that the lemma holds we have $m \geq \max(a, c, a\lfloor d/b \rfloor)$. To show $m \geq a$ let $A(x,y) \doteq y^b(x^a - 1) + 1$, $B(x,y) = 1$. To show $m \geq c$ let $A(x,y) = y^b$, $B(x,y) = y^b + x^c - 1$. To show $m \geq ad/b$ let $A(x,y) = y^b + x^a$, $B(x,y) = \prod_{i=1}^{\lfloor d/b \rfloor}(y^b + \chi_i)$, where $\chi_1, \ldots, \chi_{\lfloor d/b \rfloor}$ are elements of $\mathbf{F}$ such that the polynomial $\prod_{i=1}^{\lfloor d/b \rfloor}(x^c - \chi_i)$ has $c\lfloor d/b \rfloor$ different roots. In these examples we assumed that $\mathbf{F}$ is large enough, and in the last two cases we also assumed that $d \geq b$. We can obviously sharpen the lemma when $d < b$.*

We continue now the proof of Theorem 5.30. This part is essentially the same as in [120]. Define

$$H = \{(a,b) \mid a \in X,\ b \in Y,\ C(a,b) \neq D(a,b)\}.$$

We claim there is a polynomial $E \in \mathbf{F}^{\delta n, \delta n}[x,y]$ such that $E \neq 0$, but $E =_H 0$. Indeed, consider the linear map $\phi : \mathbf{F}^{\delta n, \delta n}[x,y] \to \mathbf{F}^H$, which maps a polynomial into the vector of values that it takes on H. Since $\dim \mathbf{F}^{\delta n, \delta n}[x,y] = \delta^2 n^2$ and $\dim \mathbf{F}^H = |H| < \delta^2 n^2$, we have that $\mathrm{Ker}(\phi)$

contains a non zero element, E. This E is the polynomial we are looking for. Assume that E has degree (e_1, e_2). We have:

$$CE =_{X\times Y} DE =_{X\times Y} G,$$

where G is a bivariate polynomial that can be chosen to have degree at most $(d+e_1, d+e_2)$. To see this we need the following lemma:

Lemma 5.34 *Let P_1 and P_2 be two bivariate polynomials of degree (d_1, n) and degree (n, d_2) respectively, where $d_1, d_2 < n$. Assume that $P_1 =_{X\times Y} P_2$, where $X = \{a_1, \ldots, a_{n+1}\} \subseteq \mathbf{F}$, $Y = \{b_1, \ldots, b_{n+1}\} \subseteq \mathbf{F}$. Then there is a polynomial P_3 of degree (d_1, d_2) such that $P_3 =_{X\times Y} P_1$. Also, if P_1 is not the identically zero polynomial, than P_3 is not the identically zero polynomial either.*

Proof. Let $X_1 \subseteq X$, $|X_1| = d_1 + 1$ and $Y_1 \subseteq Y$, $|Y_1| = d_2 + 1$. There is a polynomial P_3 such that $P_3 =_{X_1\times Y_1} P_1$. But since every degree d_1 polynomial is determined uniquely by the values it takes over a set of cardinality $d_1 + 1$, and both P_1 and P_3 have degree d_1 in the x direction, we have that $P_3 =_{X\times Y_1} P_1$. We now use the fact that $P_1 =_{X\times Y} P_2$ and so $P_3 =_{X\times Y_1} P_2$. Since both P_1 and P_3 have degree d_2 in the y direction, we get that $P_3 =_{X\times Y} P_2$. Then $P_3 =_{X\times Y} P_1$. If P_1 is not identically zero, then it cannot be zero everywhere on $A_1 \times Y_1$, and so P_3 is not identically zero. follows. □

Since $CE \in \mathbf{F}^{d+e_1, n}$, $DE \in \mathbf{F}^{n, d+e_2}$, if we apply Lemma 5.34 to $P_1 = CE$ and $P_2 = DE$, we indeed find a polynomial G that agrees with CE and DE on $X \times Y$ and has degree at most $(d+e_1, d+e_2)$. Let J be the greatest common divisor of G and E and let $G = G_1 J$, $E = E_1 J$. Let us denote the degree of J by (d_1, d_2). Clearly, $G_1 \in \mathbf{F}^{d+e_1-d_1, d+e_2-d_2}[x, y]$, $\deg E_1 = (e_1 - d_1, e_2 - d_2)$. Since so far we did not break the symmetry between x and y, without loss of generality we can assume that $e_2 - d_2 \geq e_1 - d_1$.

If C is the identically zero polynomial, then the identically zero polynomial satisfies the conditions of the theorem. Otherwise CE is not the identically zero polynomial, and so by Lemma 5.34 G is not identically zero either. Hence J is not identically zero and it has the form $J(x, y) = y^{d_2} P(x) + \ldots$, where $P(x)$ is a degree d_1 polynomial. Let $\{\lambda_1, \ldots, \lambda_{n-d_1}\}$ be a subset of $n - d_1$ distinct elements of X which does not contain any of the roots of

$P(x)$. We have:

$$\begin{aligned} G_1(\lambda_1, y)J(\lambda_1, y) &=_Y E_1(\lambda_1, y)J(\lambda_1, y)D(\lambda_1, y); \\ &\cdots \\ G_1(\lambda_{n-d_1}, y)J(\lambda_{n-d_1}, y) &=_Y E_1(\lambda_{n-d_1}, y)J(\lambda_{n-d_1}, y)D(\lambda_{n-d_1}, y). \end{aligned}$$

Since $|Y| = n > d + \delta n \geq \deg_y G$, we can conclude that these identities hold as polynomial identities and $E_1(\lambda_i, y)|G_1(\lambda_i, y)$ for $1 \leq i \leq n - d_1$. Theorem 5.32 implies that either E_1 has degree $(0, 0)$ or $n - d_1 \leq e_1 + (d + e_1 - d_1) + ((e_1 - d_1)/(e_2 - d_2))(d + e_2 - d_2)$. We show that the later cannot be the case. Indeed, by our assumption $e_1 - d_1 \leq e_2 - d_2$, therefore:

$$e_1 + (d + e_1 - d_1) + ((e_1 - d_1)/(e_2 - d_2))(d + e_2 - d_2) \leq 2d + 3\delta n - d_1 < n - d_1.$$

Let us define $F = G_1$. Since $e_1 = d_1$ and $e_2 = d_2$, $F \in \mathbf{F}^{d,d}[x, y]$ follows from $G_1 \in \mathbf{F}^{d+e_1-d_1, d+e_2-d_2}[x, y]$.

We are left to show that the number of (a, b) pairs for which $F(a, b)$ is not equal to either $C(a, b)$ or $D(a, b)$ is at most $2\delta^2 n^2$.

Let $H_1 = \{(x, y) \in X \times Y \mid C(x, y) = D(x, y) \neq F(x, y)\}$. If $|H_1| \leq \delta^2 n^2$, we are done, since

$$\begin{aligned} \{(x, y) \in X \times Y \mid F(x, y) \neq C(x, y) \wedge F(x, y) \neq D(x, y)\} \quad \subseteq \\ H_1 \cup \{(x, y) \in X \times Y \mid C(x, y) \neq D(x, y)\} \end{aligned} \tag{41}$$

where the sizes of both sets on the right hand side of (41) are bounded by $\delta^2 n^2$. If $H_1 > \delta^2 n^2$ then either $X_1 = \{x \mid (x, y) \in H_1\}$ or $Y_1 = \{y \mid (x, y) \in H_1\}$ has size greater than δn. Assume this is X_1. Then for every fixed $a \in X_1$ the polynomial $F(a, y)$ is different from the polynomial $D(a, y)$. But since $F(a, y)J(a, y) =_Y D(a, y)J(a, y)$, and $\deg_y FJ < n$, this can happen only if $J(a, y)$ is identically 0. It is easy to see that then $(x - a)$ is a factor of J. Then $\prod_{a \in X_1}(x - a)$ is a factor of J, and we get a contradiction, since $|X_1| > \delta n$, but the x-degree of J is at most δn. □

Proof. [of Theorem 5.29] Let $DIR(\mathbf{F}^2) = \{dir_1, \ldots, dir_{|F|+1}\}$. Let $r_i = \mathbf{E}_{\mathcal{L} \in CLASS(dir_i)}(a(\lambda))$. By our assumption $\frac{1}{|\mathbf{F}|+1} \sum_{i=1}^{|\mathbf{F}|+1} r_i \leq \delta$. We may assume that $r_1 \leq \ldots \leq r_{|\mathbf{F}|+1}$. Then $r_1, r_2 \leq \frac{|\mathbf{F}|+1}{|\mathbf{F}|}\delta = \delta_0$. The conditions of the theorem imply that $3\delta_0|\mathbf{F}| + 2d < |F|$. Let $\mathbf{v}_1 = (x_1, y_1)$ a non-zero vector in direction dir_1 and $\mathbf{v}_2 = (x_2, y_2)$ a non-zero vector in direction dir_2.

If we choose the basis $\{\mathbf{v}_1, \mathbf{v}_2\}$ for $\mathbf{F}^2$, Theorem 5.29 implies (with the choice of $X = Y = \mathbf{F}$) that F is 2δ-close to a degree (d, d) polynomial P on $\mathbf{F}^2$ in the basis $\{\mathbf{v}_1, \mathbf{v}_2\}$.

Assume that the total degree of P is greater than d. Then by Theorem 5.14 there are at least $(|\mathbf{F}|-d)^4$ pairs $(\mathbf{w}_1, \mathbf{w}_2)$: $\mathbf{w}_1 \in \mathbf{F}^2, \mathbf{0} \neq \mathbf{w}_2 \in \mathbf{F}^2$ such that when we restrict P on the line $\mathcal{L}_{\mathbf{w}_1,\mathbf{w}_2} = \{\mathbf{w}_1 + \lambda \mathbf{w}_2\}$, the restriction is not a degree at most d polynomial. However these restrictions must always be degree at most $2d$ polynomials, since P has degree (d, d) in the basis $\{\mathbf{v}_1, \mathbf{v}_2\}$. For these "bad" lines the normalized Hamming distance of P from the closest degree d polynomial is at least $\frac{|\mathbf{F}|-2d}{|\mathbf{F}|}$. Let $a'(\mathcal{L}) = \min_{deg(G)\leq d} dist_{\mathcal{L}}(G, P)$. By the above:

$$\mathbf{E}_{\mathbf{w}_1,\mathbf{0}\neq\mathbf{w}_2}(a'(\mathcal{L}_{\mathbf{w}_1,\mathbf{w}_2})) = \mathbf{E}_{\mathcal{L}\in\Lambda}(a'(\mathcal{L})) \geq \left(\frac{|\mathbf{F}| - d}{|\mathbf{F}|}\right)^4 \frac{|\mathbf{F}| - 2d}{|\mathbf{F}|} > 0.5. \quad (42)$$

On the other hand

$$\mathbf{E}_{\mathcal{L}\in\Lambda}(a'(\lambda)) \leq \mathbf{E}_{\mathcal{L}\in\Lambda}(a(\lambda)) + dist(F, P) \leq 3\delta \leq 0.5, \quad (43)$$

a contradiction. □

An immediate corollary of Theorem 5.29 is the following:

Theorem 5.35 *Let* $\mathbf{F}$ *be a finite field,* $F(x, y)$ *be an arbitrary function from* $\mathbf{F}^2$ *to* $\mathbf{F}$, $d \leq |\mathbf{F}|/10$ *an integer, and* $a(\mathcal{L})$ *as in Theorem 5.29. Then*

$$\min_{deg(G)\leq d} dist(F, G) \leq 6\mathbf{E}_{\mathcal{L}\in\Lambda}(a(\mathcal{L})). \quad (44)$$

The following two lemmas represent the easy direction i.e. when we have an a priory knowledge that $\min_{deg(G)\leq d} dist(F, G)$ is small. In the first lemma we estimate the average of $a(\mathcal{L})$. A little twist is that we show that this average is small even if it is taken for a set of lines incident to a fixed point. The second lemma states that if P is a multi-degree d polynomial which is close to a function F, then in order to learn the value of $P(x, y)$ for some $(x, y) \in \mathbf{F}^2$ it is sufficient to cast a random line through (x, y) and extrapolate $P(x, y)$ only from values that F takes over this line. The value we get this way equals to $P(x, y)$ with high probability. In these lemmas $\mathbf{F}$ is a finite field.

Lemma 5.36 *Let $F(x,y)$ be an arbitrary function from $\mathbf{F}^2$ to $\mathbf{F}$, and let P be the closest total degree $\mathbf{d}$ polynomial to F. $\mathbf{v} = (x_0, y_0)$ an arbitrary fixed point in $\mathbf{F}^2$, and $a(\mathcal{L})$ as in Theorem 5.29. Then*

$$\mathbf{E}_{\mathcal{L} \in \Lambda_{\mathbf{v}}}(a(\mathcal{L})) \leq \frac{1}{|\mathbf{F}|} + dist(F,P) \tag{45}$$

Proof. Let $E = \{(x,y) \mid P(x,y) \neq F(x,y)\}$. The restriction of P on a line $\mathcal{L}$ is a degree d polynomial, so $a(\mathcal{L}) \leq dist_{\mathcal{L}}(P,F)$ for every $\mathcal{L} \in \Lambda$. Therefore $a(\mathcal{L}) \leq dist_{\mathcal{L}}(P,F)$. Thus it is enough to show that

$$\mathbf{E}_{\mathcal{L} \in \Lambda_{\mathbf{v}}}(dist_{\mathcal{L}}(P,F)) \leq \frac{1}{|\mathbf{F}|} + dist(F,P). \tag{46}$$

Since the lines incident to $\mathbf{v}$ are disjoint outside $\mathbf{v}$, those $(x,y) \neq \mathbf{v}$ for which $F(x,y) \neq P(x,y)$ contribute $\frac{1}{|\mathbf{F}|(|\mathbf{F}|+1)}$ to the left hand side of (46) which is less than $\frac{1}{|\mathbf{F}|^2}$ which they contribute to the right hand side. $F(\mathbf{v}) \neq P(\mathbf{v})$ can add an additional $\frac{1}{|\mathbf{F}|}$ to the left hand side. Inequalities (46) and (45) follow.

Lemma 5.37 *Let $F(x,y)$ be an arbitrary function from $\mathbf{F}^2$ to $\mathbf{F}$, Let P be the closest total degree $\mathbf{d}$ polynomial to F and $d \leq |\mathbf{F}|/10$. For $\mathcal{L} \in \Lambda$ we denote by $P_{\mathcal{L}}$ the uni-variate degree $\mathbf{d}$ polynomial on $\mathcal{L}$ which is closest to F on $\mathcal{L}$. Then for any fixed $\mathbf{v} = (x_0, y_0) \in \mathbf{F}^2$:*

$$Prob_{\mathcal{L} \in \Lambda_{\mathbf{v}}}(P(x_0,y_0) \neq P_{\mathcal{L}}(x_0,y_0)) \leq \frac{2}{|\mathbf{F}|} + 3dist(F,P).$$

Proof. If $P_{\mathcal{L}}(x_0,y_0) \neq P(x_0,y_0)$ then $P_{\mathcal{L}}$ is different from the restriction of P on $\mathcal{L}$. Since both $P_{\mathcal{L}}$ and the restriction of P on $\mathcal{L}$ are (uni-variate) polynomials of degree d, the number of points of $\mathcal{L} \setminus (x_0,y_0)$ where they do not equal is at least $|\mathbf{F}| - d$. By the triangle inequality the number of points of $\mathcal{L} \setminus (x_0,y_0)$ where P is not equal to F is at least $|\mathbf{F}| - d - |\mathbf{F}|a(\mathcal{L})$. Summing these up for $\mathcal{L} \in \Lambda_{\mathbf{v}}$:

$$|\mathbf{F}| \sum_{\mathcal{L} \in \Lambda_{\mathbf{v}}} a(\mathcal{L}) + |\{(x,y) \neq \mathbf{v} \mid F(x,y) \neq P(x,y)\}| \geq (|\mathbf{F}| - d)(|\mathbf{F}|+1)p, \tag{47}$$

where $p = Prob_{\mathcal{L} \in \Lambda_{\mathbf{v}}}(P(x_0,y_0) \neq P_{\mathcal{L}}(x_0,y_0))$. We get that

$$p \leq \frac{|\mathbf{F}|}{|\mathbf{F}|-d}\frac{\sum_{\mathcal{L}\in\Lambda_v} a(\mathcal{L})}{|\mathbf{F}|+1} + \frac{|\mathbf{F}|^2 dist(F,P)-1}{(|\mathbf{F}|-d)(|\mathbf{F}|+1)} \leq$$
$$\frac{|\mathbf{F}|}{|\mathbf{F}|-d}\left(\frac{1}{|\mathbf{F}|} + dist(F,P)\right) + \frac{|\mathbf{F}|^2 dist(F,P)-1}{(|\mathbf{F}|-d)(|\mathbf{F}|+1)} \leq$$
$$\frac{2}{|\mathbf{F}|} + 3dist(F,P)$$

The first term of the next to last expression comes from the first term of the previous expression by Lemma 5.37. The last expression follows from $|\mathbf{F}| \geq 2$ and $d \leq |\mathbf{F}|/10$. □

5.2.3 Checking General Low Degree Polynomials

We use the definitions of Section 5.1.6. Let us fix a space $\mathbf{F}^n$, a function $F : \mathbf{F}^n \to \mathbf{F}$, and a degree d. We denote the set of lines and planes in $\mathbf{F}^n$ by $\Lambda = \Lambda(\mathbf{F}^n)$ and $\Pi = \Pi(\mathbf{F}^n)$, respectively. For $\mathbf{x} \in \mathbf{F}^n$ we introduce $\Lambda_{\mathbf{x}} = \{\mathcal{L} \in \Lambda \mid \mathbf{x} \in \mathcal{L}\}$ and $\Pi_{\mathbf{x}} = \{\mathcal{P} \in \Pi \mid \mathbf{x} \in \mathcal{P}\}$.

Arora, Lund, Motwani, Sudan and Szegedy in [10] showed that the Random Line Test works in general in n dimension, and gives a constant error irrespective of the dimension. The test plays a central role in their PCP construction. Similarly to the two-dimensional case we introduce for a line $\mathcal{L}$ the quantity $a(\mathcal{L})$ and the polynomial $P_{\mathcal{L}}$ defined by:

$$a(\mathcal{L}) = dist_{\mathcal{L}}(P_{\mathcal{L}}, F) = \min_{deg(P)\leq d} dist_{\mathcal{L}}(P, F),$$

where $P_{\mathcal{L}}$ is a degree d (univariate) polynomial such that its Hamming distance from F on $\mathcal{L}$ is minimal. For $\mathcal{P} \in \Pi$ we define $b(\mathcal{P})$ and $P_{\mathcal{P}}$ by:

$$b(\mathcal{P}) = dist_{\mathcal{P}}(P_{\mathcal{P}}, F) = \min_{deg(P)\leq d} dist_{\mathcal{P}}(P, F),$$

where $P_{\mathcal{L}}$ is a multi-degree d (bivariate) polynomial such that its Hamming distance from F on $\mathcal{P}$ is minimal. The proof of the generalized Random Line Test relies on Theorems 5.29, 5.36 and 5.37. If we apply these theorems for a fixed plane $\mathcal{P}$ of $\mathbf{F}^n$, then we get:

$$b(\mathcal{P}) \leq 6\mathbf{E}_{\mathcal{L}\in\Lambda(\mathcal{P})}(a(\mathcal{L})), \qquad (48)$$

$$\mathbf{E}_{\mathcal{L}\in\Lambda_{\mathbf{x}_0}(\mathcal{P})}(a(\mathcal{L})) \leq \frac{1}{|\mathbf{F}|} + b(\mathcal{P}), \tag{49}$$

$$Prob_{\mathcal{L}\in\Lambda_{\mathbf{x}_0}(\mathcal{P})}(P_{\mathcal{P}}(\mathbf{x}_0) \neq P_{\mathcal{L}}(\mathbf{x}_0)) \leq \frac{2}{|\mathbf{F}|} + 3b(\mathcal{P}), \tag{50}$$

where $\mathbf{x}_0$ is a fixed point of $\mathcal{P}$ and $\Lambda_{\mathbf{x}_0}(P)$ is the set of lines in $\mathcal{P}$ that contain $\mathbf{x}_0$.

Theorem 5.38 (Random Line Test, Multi-Variate [10]) *Let* $\mathbf{F}$ *be a finite field,* $|\mathbf{F}| > 2d+288\delta|\mathbf{F}|+698$, $d \leq |\mathbf{F}|/10$ *and* $|\mathbf{F}| > nd$. *If* $\mathbf{E}_{\mathcal{L}\in\Lambda}(a(\mathcal{L})) \leq \delta$, *then*

$$\min_{deg(P)\leq d} dist(P, F) \leq 36\delta + \frac{77}{|\mathbf{F}|}. \tag{51}$$

Proof. If D is a probability distribution on a set A and f is a function from A to a set B, then we will say that f is constant with probability (at least) q with respect to D, if $\max_{b\in B} Prob_{a\in D}(f(a) = b) \geq q$. If $q \geq 1/2$ then the most popular value, b, is unique. If for a distribution D' we have $|D' - D|_1 \leq \epsilon$, then f is constant with probability $q - \epsilon$ with respect to D'. The following lemma is trivial to prove:

Lemma 5.39 *Let* D *be a probability distribution on a set* A, f *be a map from* A *to a set* B. *Let* $Prob_{a_1\neq a_2\in A}(f(a_1) \neq f(a_2)) \geq q$. *Then* f *is constant with probability* q *with respect to* D.

Lemma 5.40 *Let* $\mathcal{L}_0 \in \Lambda$, $x_0 \in \mathcal{L}_0$, *fixed. The maps below are constant with probability* $1 - 36\delta - \frac{76}{|\mathbf{F}|}$ *and* $1 - 144\delta - \frac{349}{|\mathbf{F}|}$, *respectively, and their unique most popular values coincide.*

1. $\mathcal{L} \to P_{\mathcal{L}}(\mathbf{x}_0)$, *when* $\mathcal{L}$ *is chosen uniformly from* $\Lambda_{\mathbf{x}_0}$.
2. $\mathcal{P} \to P_{\mathcal{P}}(\mathbf{x}_0)$, *when* $\mathcal{P}$ *is chosen uniformly from* $\Pi_{\mathcal{L}_0}$.

Before proving Lemma 5.40 we show that it implies the theorem. Let $G(\mathbf{x}_0)$ be the unique most popular value of 1. and 2. of Lemma 5.40. First we show that $deg(G) \leq d$. By Lemma 5.13 it is enough to show that if we restrict G to any line we get a degree at most d polynomial. Fix a line $\mathcal{L}$. For $\mathcal{P} \in \Pi_{\mathcal{L}}$ let $Q_{\mathcal{P}}$ be the restriction of $P_{\mathcal{P}}$ on $\mathcal{L}$. When $\mathcal{P}$ is selected randomly from $\Pi_{\mathcal{L}}$, the polynomial $Q_{\mathcal{P}}$ is a random variable with the property that for every $\mathbf{x} \in \mathcal{L}$ the probability that $Q_{\mathcal{P}}(\mathbf{x}) = G(\mathbf{x})$ is at least $1-144\delta-\frac{349}{|\mathbf{F}|}$. The desired property of G now follows from the lemma below.

Lemma 5.41 *Let D be a probability distribution on degree d polynomials over $\mathbf{F}$, and $f : \mathbf{F} \to \mathbf{F}$ such that for every $t \in \mathbf{F}$ we have $Prob_{p\in D}(p(t) = f(t)) > \frac{1}{2} + \frac{d}{|\mathbf{F}|}$. Then f is a degree d polynomial.*

Proof. Let X be the probability variable that takes the value $|\{t \mid p(t) = f(t)\}|$ on a polynomial $p \in D$. Clearly, $E(x) = \sum Prob_D(f(t) = p(t)) \geq \left(\frac{1}{2} + \frac{d}{|\mathbf{F}|}\right)|\mathbf{F}| > |\mathbf{F}|/2$, and so there is a $p_0 \in D$ such that $|\{t \mid p_0(t) = f(t)\}| > |\mathbf{F}|/2$. We will show that for every $t \in \mathbf{F}$ $f(t) = p_0(t)$.

Let us denote $\{t \mid p_0(t) = f(t)\}$ by H. Pick an arbitrary $t_0 \in \mathbf{F}$, and let X_{t_0} be the probability variable that takes the value $|\{t \in H \mid p(t) = f(t)\}|$ on $p \in D$ if $p(t_0) = f(t_0)$, and 0 otherwise. We have:

$$E(X_{t_0}) = \sum_{t\in H} Prob(p(t_0) = f(t_0) \wedge p(t) = f(t)) > |H|\frac{2d}{|\mathbf{F}|} > \frac{|\mathbf{F}|}{2}\frac{2d}{|\mathbf{F}|} = d.$$

Thus there is a $q \in D$ such that $q(t_0) = f(t_0)$ and q agrees with f on more that d values of H. But then q and p also coincide on more than d values, therefore they are equal, and so $p(t_0) = f(t_0)$. □

Next we show that $dist(F, G) \leq 36\delta + \frac{77}{|\mathbf{F}|}$. The premise of the theorem can be reformulated as

$$Prob_{\mathbf{x}\in\mathbf{F}^n, \mathcal{L}\in\Lambda_{\mathbf{x}}}(F(\mathbf{x}) \neq P_{\mathcal{L}}(\mathbf{x})) \tag{52}$$

for the reason that the left hand side equals to $Prob_{\mathcal{L}\in\Lambda,\mathbf{x}\in\mathcal{L}}(F(\mathbf{x}) \neq P_{\mathcal{L}}(\mathbf{x})) \leq \delta$. By 1. of Lemma 5.40 for any fixed $\mathbf{x}_0$:

$$Prob_{\mathcal{L}\in\Lambda_{\mathbf{x}_0}}(G(\mathbf{x}_0) \neq \mathcal{P}_{\mathcal{L}}(\mathbf{x}_0)) \leq 36\delta + \frac{76}{|\mathbf{F}|} \tag{53}$$

If we randomize Equation (53) over $\mathbf{x}_0$ and combine it with Equation (52) through the triangle inequality, we obtain that

$$Prob_{\mathcal{L},\mathbf{x}\in\mathcal{L}}(F(\mathbf{x}) \neq G(\mathbf{x})) \leq 36\delta + \frac{77}{|\mathbf{F}|}. \tag{54}$$

We are done, since $Prob_{\mathcal{L}\in\Lambda,\mathbf{x}\in\mathcal{L}}(F(\mathbf{x}) \neq G(\mathbf{x})) = Prob_{\mathbf{x}\in\mathbf{F}^n}(F(\mathbf{x}) \neq G(\mathbf{x})) = dist(F, G) \geq \min_{deg(P)\leq d} dist(F, P)$, since G is a total degree d polynomial. We are left to prove Lemma 5.40.

First we give an upper bound on the expectation of $b(\mathcal{P})$ for a random $\mathcal{P} \in \Pi_{\mathbf{x}_0}$. If we pick $\mathcal{P} \in \Pi_{\mathbf{x}_0}$ uniformly and $\mathcal{L} \in \Lambda(\mathcal{P})$ uniformly then by

Lemma 5.20 we get a distribution which is $\frac{2}{|\mathbf{F}|}$-close to the uniform distribution on Λ in the L_1 norm. Therefore:

$$\mathbf{E}_{\mathcal{P}\in\Pi_{\mathbf{x}_0}}\mathbf{E}_{\mathcal{L}\in\Lambda(\mathcal{P})}(a(\mathcal{L})) \leq \mathbf{E}_{\mathcal{L}\in\Lambda}(a(\mathcal{L})) + \frac{2}{|\mathbf{F}|} \leq \delta + \frac{2}{|\mathbf{F}|}. \tag{55}$$

From (48) and (55):

$$\mathbf{E}_{\mathcal{P}\in\Pi_{\mathbf{x}_0}}(b(\mathcal{P})) \leq 6\mathbf{E}_{\mathcal{P}\in\Pi_{\mathbf{x}_0}}\mathbf{E}_{\mathcal{L}\in\Lambda(\mathcal{P})}(a(\mathcal{L})) \leq 6\delta + \frac{12}{|\mathbf{F}|} \tag{56}$$

Pick now a random pair of lines $\mathcal{L}_1 \neq \mathcal{L}_2$ through $\mathbf{x}_0$ together with the plane $\mathcal{P}$ that contains both. We define the characteristic functions of three events on the probability space whose elements are the triplets $(\mathcal{L}_1, \mathcal{L}_2, \mathcal{P})$ as above.

1. X_1: the characteristic function of the event that $P_{\mathcal{L}_1}(\mathbf{x}_0) \neq P_{\mathcal{L}_2}(\mathbf{x}_0)$.
2. X_2: the characteristic function of the event that $P_{\mathcal{L}_1}(\mathbf{x}_0) \neq P_{\mathcal{P}}(\mathbf{x}_0)$.
3. X_3: the characteristic function of the event that $P_{\mathcal{L}_2}(\mathbf{x}_0) \neq P_{\mathcal{P}}(\mathbf{x}_0)$.

Clearly, $X_1 \leq X_2 + X_3$ point-wise. Thus $\mathbf{E}(X_1) \leq \mathbf{E}(X_2) + \mathbf{E}(X_3)$. The events underlying X_2 and X_3 have the same probability as the following one: First we pick a random plane $\mathcal{P}$ and then a random line $\mathcal{L}$ in $\mathcal{P}$ such that both go through $\mathbf{x}_0$ and consider the event that $P_{\mathcal{P}}(\mathbf{x}_0) \neq P_{\mathcal{L}}(\mathbf{x}_0)$. If we fix $\mathcal{P}$ then by Lemma 5.37 the probability that $P_{\mathcal{P}}(\mathbf{x}_0) \neq P_{\mathcal{L}}(\mathbf{x}_0)$ is bounded by $\frac{2}{|\mathbf{F}|} + 3b(\mathcal{P})$. Hence by (56):

$$\mathbf{E}(X_2) = \mathbf{E}(X_3) \leq \mathbf{E}_{\mathcal{P}\in\Pi_{\mathbf{x}_0}}\left(\frac{2}{|\mathbf{F}|} + 3b(\mathcal{P})\right) \leq 18\delta + \frac{38}{|\mathbf{F}|},$$

from which $\mathbf{E}(X_1) \leq 36\delta + \frac{76}{|\mathbf{F}|}$ follows. This implies by Lemma 5.39 that $P_{\mathcal{L}}(\mathbf{x}_0)$ is constant with probability $36\delta + \frac{76}{|\mathbf{F}|}$. Let us denote the most popular value of $P_{\mathcal{L}}(\mathbf{x}_0)$ by $G(\mathbf{x}_0)$.

Next we show that $P_{\mathcal{P}}(\mathbf{x}_0) = G(\mathbf{x}_0)$ with probability at least $144\delta + \frac{349}{|\mathbf{F}|}$, where $\mathcal{P}$ is randomly chosen from $\Pi_{\mathcal{L}_0}$. (Here $\mathcal{L}_0$ is the fixed line of Lemma 5.40.) Consider the following two distributions on the set of lines that intersect $\mathcal{L}_0$:

1. D_1: Pick a random point $\mathbf{x}$ of $\mathcal{L}_0$ and then pick a random line in $\Lambda_{\mathbf{x}}$.

2. D_2: Pick a random plane in $\Pi_{\mathcal{L}_0}$ and pick a random line in it.

It is easy to see that the L_1 norm of the difference of the two distributions is less than $\frac{2}{|\mathbf{F}|}$. Therefore from (48), (56) and (49):

$$\begin{aligned}
\mathbf{E}_{\mathcal{P}\in\Pi_{\mathcal{L}_0}}(b(\mathcal{P})) &\leq \\
6\mathbf{E}_{\mathcal{P}\in\Pi_{\mathcal{L}_0},\mathcal{L}\in\Lambda(\mathcal{P})}(a(\mathcal{L})) = 6\mathbf{E}_{\mathcal{L}\in D_2}(a(\mathcal{L})) &\leq \\
6\left(\mathbf{E}_{\mathcal{L}\in D_1}(a(\mathcal{L})) + \frac{2}{|\mathbf{F}|}\right) = 6\left(\mathbf{E}_{x\in\mathcal{L}_0}\mathbf{E}_{\mathcal{L}\in\Lambda_x}\left(a(\mathcal{L}) + \frac{2}{|\mathbf{F}|}\right)\right) &\leq \\
6\left(\mathbf{E}_{x\in\mathcal{L}_0}\mathbf{E}_{\mathcal{P}\in\Pi_x}\left(b(\mathcal{L}) + \frac{3}{|\mathbf{F}|}\right)\right) \leq 6\left(6\delta + \frac{12}{|\mathbf{F}|}\right) + \frac{18}{|\mathbf{F}|} = 36\delta + \frac{90}{|\mathbf{F}|}.&
\end{aligned}$$

From this using (50) and 1. of Lemma 5.40:

$$\begin{aligned}
Prob_{\mathcal{P}\in\Pi_{\mathcal{L}_0}}(P_{\mathcal{P}}(\mathbf{x}_0) \neq G(\mathbf{x}_0)) \leq Prob_{\mathcal{P}\in\Pi_{\mathcal{L}_0},\mathcal{L}\in\Lambda_{\mathbf{x}_0}(P)}(P_{\mathcal{P}}(\mathbf{x}_0)) \neq P_{\mathcal{L}}(\mathbf{x}_0)) &+ \\
Prob_{\mathcal{P}\in\Pi_{\mathcal{L}_0},\mathcal{L}\in\Lambda_{\mathbf{x}_0}(P)}(P_{\mathcal{L}}(\mathbf{x}_0) \neq G(\mathbf{x}_0)) &\leq \\
\mathbf{E}_{\mathcal{P}\in\Pi_{\mathcal{L}}}\left(\frac{2}{|\mathbf{F}|} + 3b(\mathcal{P})\right) + \left(\frac{1}{|\mathbf{F}|} + \left(36\delta + \frac{76}{|\mathbf{F}|}\right)\right) \leq 144\delta + \frac{349}{|\mathbf{F}|}.&
\end{aligned}$$

□

We get the following theorem as an immediate consequence.

Theorem 5.42 (Checkability of Total Degree) *Let* $\mathbf{F}$ *be a finite field,* $1 > \delta > \frac{3}{|\mathbf{F}|}$, $d \geq 0$, *such that*

$$|\mathbf{F}| \geq \max(nd, 10d, 2d + 288\delta|\mathbf{F}| + 698). \tag{57}$$

Then the δ*-neighborhood of* $\mathbf{F}^{deg<d}[\mathbf{x}]$ *is* $\delta/100$*-checkable with check sets of size* $|\mathbf{F}|$.

5.2.4 Checking Subspaces of Low Degree Polynomials

Although the space $\mathbf{F}^d[x_1, \ldots, x_n]$ is checkable for appropriate choices of parameters, a subspace of this space is not necessarily checkable. In this section we provide a lemma that shows that many of the important subspaces of $\mathbf{F}^d[\mathbf{x}]$ are checkable.

Consider, for instance $\mathbf{F}^{n\times\Delta}[\mathbf{x}]$, the space of multi-degree Δ polynomials. We would like to show that $\mathbf{F}^{n\times\Delta}[\mathbf{x}]$ is checkable when $|\mathbf{F}|$ is sufficiently large compared to n and Δ.

Assume that a black box function $F : \mathbf{F}^n \to \mathbf{F}$ is presented to us, that is guaranteed to be in $\mathbf{F}^{n\Delta}[\mathbf{x}]$ (a space that contains $\mathbf{F}^{n\times\Delta}[\mathbf{x}]$). We need to check that the degree of each individual variable is at most Δ. To this end select a random point $(y_1, \ldots, y_{n-1}) \in \mathbf{F}^{n-1}$ and check F over the *union* of the lines

$$\begin{aligned} \mathcal{L}_1(x) &= (x, y_1, \ldots, y_{n-1}), \\ \mathcal{L}_2(x) &= (y_1, x, \ldots, y_{n-1}), \\ &\vdots \\ \mathcal{L}_n(x) &= (y_1, \ldots, y_{n-1}, x). \end{aligned}$$

If with high probability over $\mathbf{y}$ for *all* lines above we get a degree Δ uni-variate polynomial, then it is easy to see, that F has multi-degree Δ. However, even with the promise that F belongs to $\mathbf{F}^{n\Delta}[\mathbf{x}]$, we cannot directly apply the above fact to check membership in $\mathbf{F}^{n\times\Delta}[\mathbf{x}]$ in the sense of Definition 5.27. The main problem is that if even over only 1% of $\mathcal{L}_1, \ldots, \mathcal{L}_n$ F has high degree (for an average $\mathbf{y}$), the low multi-degree of F is not guaranteed. To circumvent the above problem let

$$H_{\mathbf{y}} = \mathcal{L}_1 \cup \mathcal{L}_2 \cup \cdots \cup \mathcal{L}_n$$

for a fixed $\mathbf{y}$. We will embed H_ys into families of standard varieties in order to be able to perform the error-correction necessary. The method is spelled out in details in the next lemma in a more general form.

Lemma 5.43 (Subspace Checking Lemma) *Let $1 > \delta > 0$ and S be a subspace of $\mathbf{F}^{deg\leq d}[x_1, \ldots, x_n]$, where $\mathbf{F}$ is a finite field, and*

$$|\mathbf{F}| \geq \max(nd, 10d, 2d + 288\delta|\mathbf{F}| + 698)$$

Assume furthermore that there are $H_1, \ldots, H_m \subseteq \mathbf{F}^n$, each of size at most g, such that for any $P \in \mathbf{F}^{deg\leq d}[x_1, \ldots, x_n]$ the inequality

$$|\{i \mid \min_{Q\in S} dist_{H_i}(Q, P) = 0\}| \geq (1 - \delta)m \tag{58}$$

implies $P \in S$. Then the $\delta/4$-neighborhood of S is $\delta/800$-checkable. The check sets have size as most $g^2|\mathbf{F}|^2$, and we have at most $|\mathbf{F}|^n(m + |\mathbf{F}|^n)$ of them.

For the lemma to hold we also need the technical assumption that $\delta > 16\lceil \log_{|\mathbf{F}|} g \rceil / \sqrt{|F|}$.

Proof. A seeming checking procedure: Flip a coin. If head, check if F is a degree d polynomial by casting a random line inside $\mathbf{F}^n$, like in the previous section. If the coin-flip is tail, pick $1 \leq i \leq m$ randomly, and read F over H_i.

This straightforward procedure does not work for two reasons:

1. In the second part we cannot use that F is a degree d polynomial, only that it is close to one.
2. Even if we could assume that F is identical with a degree d polynomial, Equation (58) is not sufficiently strong to provide checkability in the sense of Definition 5.27.

We can overcome on both problems if we embed each H_i in members of a standard family of parameterized varieties defined in Section 5.1.7 (one family for each H_i). For the choices of $D = \lceil\sqrt{|\mathbf{F}|}\rceil$ (the degree of the varieties) and $k = 2\lceil\log_{|\mathbf{F}|} g\rceil$ (the dimension of the varieties) our varieties will have size no more than $|\mathbf{F}|^2 g^2$.

Remark 5.44 *In the case of the space of multi-degree Δ polynomials $g = |H_{\mathbf{y}}| = n|\mathbf{F}|$. In our applications n will be bounded by an $|\mathbf{F}|$ power, and k becomes a constant.*

For each H_i the associated family of varieties, $\{\mathcal{V}_{i,j}\}_{1\leq j\leq|\mathbf{F}|^n}$, contains exactly $|\mathbf{F}|^n$ members.

The correct way of checking membership in S (so that we comply with the constraints of Definition 5.27) is: flip a coin; if head, check a random line in $\Lambda(\mathbf{F}^n)$, and if tail check a random variety $\mathcal{V}_{i,j}$. Clearly, every check set has size at most $|\mathbf{F}|^2 g^2$, and the number of check sets is $|\Lambda(\mathbf{F}^n)| + m|\mathbf{F}|^n \leq |\mathbf{F}|^n(m + |\mathbf{F}|^n)$.

We are left to prove that the above procedure $\delta/800$-checks the $\delta/4$-neighborhood of S. If F is outside the $\delta/4$-neighborhood of S, it can be for one of the two reasons. We separate two cases accordingly.

The first case is if F is outside the $\delta/4$-neighborhood of $\mathbf{F}^{deg<d}[\mathbf{x}]$. In this case

$$\mathbf{E}_{\mathcal{L}\in\Lambda(\mathbf{F}^n)}(\min_{deg(P)\leq d} dist_{\mathcal{L}}(F, P)) \geq \delta/400$$

by Theorem 5.42. We suffer a factor of two loss, because the line check is performed only with probability 1/2, and hence the constant reduces to

$\delta/800$ in the checkability parameter. The second case is when there is a $P \in \mathbf{F}^{deg<d}[\mathbf{x}]$ such that $dist(P,F) \leq \delta/4$, but P is not in the space S. By Inequality (58) we can find $l = \delta m$ indices, $i_1, \ldots, i_l$, such that for $i \in I = \{i_1, \ldots, i_l\}$ $\min dist_{H_i}(P,S) \neq 0$. Since the variety $\mathcal{V}_{i,j}$ contains H_i for any $1 \leq j \leq |\mathbf{F}|^n$, we have that for any $i \in I$ and any $1 \leq j \leq |\mathbf{F}|^n$:

$$\min_{Q \in S} dist_{\mathcal{V}_{i,j}}(P,Q) \neq 0.$$

From this an easy application of Lemma 5.22 for $\mathcal{V}_{i,j}$, P and Q, with our parameters yields that

$$\min_{Q \in S} dist_{\mathcal{V}_{i,j}}(P,Q) \geq 1/2.$$

(Here is where we use the error correcting properties of low degree polynomials over low degree varieties.) In conclusion:

$$\mathbf{E}_{i,j}(\min_{Q \in S} dist_{\mathcal{V}_{i,j}}(P,Q)) \geq \delta/2.$$

We are not done yet since we need to get an inequality for F and not for P. Because of the triangle inequality it is sufficient to show that

$$\mathbf{E}_{i,j}(dist_{\mathcal{V}_{i,j}}(P,F)) \leq 3\delta/8.$$

We use Lemma 5.24:

$$\mathbf{E}_{i,j}(dist_{\mathcal{V}_{i,j}}(P,F)) \leq dist(P,F) + \frac{|\mathbf{F}|^k - (|\mathbf{F}| - D)^k}{|\mathbf{F}|^k} \leq \delta/4 + \delta/8.$$

The bound on the second term comes from the technical assumption we made in the end of the statement of the lemma. Again, the parameter, $\delta/8$, we obtain suffers a factor of two loss, because we choose to check the families of varieties only with probability 1/2, but it is still greater than $\delta/800$. □

5.2.5 A Subspace that Deserves Special Attention

We shall construct a checkable subspace of $\mathbf{F}^{deg \leq d}[\mathbf{x}, \mathbf{y}, \mathbf{z}]$ $(\mathbf{x} \in \mathbf{F}^n, \mathbf{y} \in \mathbf{F}^m, \mathbf{z} \in \mathbf{F}^n)$ such that each element P_f of this subspace encodes a function f from X^n to $\mathbf{F}$, where $X \subseteq \mathbf{F}$ with size $D + 1$, and $0 \in X$. Here D is an appropriately small, but fixed root of $|\mathbf{F}|$. The encoding will be associated

with an arbitrarily chosen fixed function $g : X^n \times X^m \to \mathbf{F}$, and will have the property that for every $\mathbf{y} \in X^m$:

$$\sum_{\mathbf{x} \in X^n} f(\mathbf{x}) g(\mathbf{x}, \mathbf{y}) = P_f(\mathbf{0}, \mathbf{y}, \mathbf{0}). \tag{59}$$

The degree d of P_f will be $d = nD + (n+m)D + n$. This encoding will serve the basis for the holographic code in section 5.3.4, which is fundamental in the basic PCP construction.

Let $b_i = \sum_{a \in X} a^i$ and

$$Q = \sum_{\alpha \leq n \times D} \sum_{\beta \leq m \times D} c_{\alpha,\beta} y^\beta x^\alpha \in \mathbf{F}[\mathbf{x}, \mathbf{y}].$$

Define:

$$P_Q = \sum_{\alpha \leq n \times D} \sum_{\beta \leq m \times D} c_{\alpha,\beta} \mathbf{y}^\beta \prod_{i=1}^{n} (z_i x_i^{\alpha_i} + (1 - z_i) b_{\alpha_i}),$$

Lemma 5.45 *For every* $\mathbf{x}_0 \in \mathbf{F}^n$ *and* $\mathbf{y} \in \mathbf{F}^m$*:*

$$\sum_{\mathbf{x} \in X^n} Q(\mathbf{x}, \mathbf{y}) = P_Q(\mathbf{x}_0, \mathbf{y}, \mathbf{0}) \tag{60}$$

Also:

$$Q(\mathbf{x}, \mathbf{y}) = P_Q(\mathbf{x}, \mathbf{y}, \mathbf{1}). \tag{61}$$

Proof. Equation (61) is immediate. For Equation (60) notice that it is enough to prove that for a fixed α:

$$\sum_{\mathbf{x} \in X^n} \prod_{i=1}^{n} x_i^{\alpha_i} = \prod_{i=1}^{n} b_{\alpha_i}, \tag{62}$$

since (60) can be obtained as a linear combination of equations as in (62). Equation (62) immediately comes from $b_{\alpha_i} = \sum_{x_i \in X} x_i^{\alpha_i}$. □

Fix $g(\mathbf{x}, \mathbf{y}) : X^n \times X^m \to \mathbf{F}$. We define a map from the space of functions of the type $f(\mathbf{x}) : X^n \to \mathbf{F}$ into the space of $2n + m$ variate polynomials as follows:

1. Extend both f and g into polynomials of multi-degree $n \times D$ and $(m + n) \times D$, respectively (by Lemma 5.11). By Lemma 5.10 these extensions are unique. Let they be f' and g', respectively.

2. Let $Q = f'g'$. We map f to P_Q.

It is easy to see that the encoding is linear and that

$$\sum_{\mathbf{x}\in X^n} f(\mathbf{x})g(\mathbf{x},\mathbf{y}) = P_Q(\mathbf{0},\mathbf{y},\mathbf{0}). \tag{63}$$

(The equation follows from Lemma 5.45.)

Theorem 5.46 *Let* $|\mathbf{F}| > (nmD)^{c_0}$ *for some sufficiently large fixed* c_0. *Then the* 10^{-3}*-neighborhood of the space*

$$S_g = \{P_Q \mid Q = f'g',\ f : X^n \to \mathbf{F}\}$$

is 10^{-6}*-checkable with check sets of size at most* $|\mathbf{F}|^8$.

We give a sketch of the proof. The first thing to notice is that S_g is a subspace of $\mathbf{F}^d[\mathbf{x},\mathbf{y},\mathbf{z}]$, so we may use the subspace-checking lemma of the previous section if appropriate H_is are found. The problem is strongly related to checking the space of multi-degree D polynomials, but a little bit more complicated. For constructing H_i:

1. For every $1 \le i \le m$ select a random line such that all coordinates but y_i are fixed.

2. For every $1 \le i \le n$ select a random plane such that all coordinates, but X_i and z_i are fixed.

3. Select a random line in general direction in the subspace defined by $\mathbf{z} = \mathbf{1}$.

4. Select a random line in a direction independent of $\mathbf{x}$ (such that only the $\mathbf{y}$ coordinates can vary in the subspace defined by $\mathbf{z} = \mathbf{1}$.

Take the union of these $m+2$ lines and n planes to obtain a check set. To reduce randomness, (i.e. the number of check sets) we notice that the random selections of lines and planes need not be independent selections (we had a similar way of recycling random bits in the example of the previous section). It is easy to see that it is sufficient if i goes from 1 to $|\mathbf{F}|^{2n+2m}$. The size of the check set, g, is $(m+2)|\mathbf{F}| + n|\mathbf{F}|^2$.

In order to show that these check sets satisfy condition (58) of Lemma 5.43, we need to prove that if $|\mathbf{F}|$ is large enough compared to m and n

(meaning that it is a fixed power of them), and $P(\mathbf{x}, \mathbf{y}, \mathbf{z})$ is a polynomial of (total) degree d such that on 99% of the check sets the restriction (offset) of P belongs to S_g, then P belongs to S_g. Informally, items 1. and 2. ensure that P is of the form P_Q, and 3. ensures that Q is of the form $f'g'$. To see this we need to combine Equation (61) of Lemma 5.45 and the following lemma:

Lemma 5.47 *Let Q and G be a fixed degree d polynomials of n variables over a field $\mathbf{F}$. If $|\mathbf{F}|$ is a sufficiently large polynomial of n and d, and over 99% of randomly chosen lines the restriction of Q is divisible with the restriction of G, then there is a polynomial F such that $Q = FG$.*

We leave out the exact details. Finally, item 4. ensures that f' depends only on $\mathbf{x}$ and not on $\mathbf{y}$.

5.3 Holographic Codes

Definition 5.48 *A linear encoding E of $\mathbf{F}^k$ into $\mathbf{F}^n$ is δ-holographic with check-size g with respect to a set of linear functions $L_1, \ldots, L_p : \mathbf{F}^k \to \mathbf{F}$, if there are $W_{1,1}, \ldots, W_{pm} \in DISTR(\{1, \ldots, n\})$, $supp(W_{i,j}) \leq g$ $(1 \leq i \leq p$, $1 \leq j \leq m)$, $Q_1, \ldots, Q_i \in DISTR(\{1, \ldots, m\})$, and a decoding function $D : \mathbf{F}^n \to \mathbf{F}^k$ such that:*

1. *For every $\mathbf{y} \in \mathbf{F}^k$ $D(E(\mathbf{y})) = \mathbf{y}$.*
2. *For every $\mathbf{v} \in \mathbf{F}^n$ and every $1 \leq i \leq p$ if $a = L_i(D(\mathbf{v}))$, then*

$$\mathbf{E}_{j \in Q_i} \left(\min_{\mathbf{z} = E(\mathbf{y}) : L_i(\mathbf{y}) \neq a} dist_{W_{i,j}}(\mathbf{z}, \mathbf{v}) \right) \geq \delta \tag{64}$$

We say that in the above definition E is a δ-holographic encoding of $L_1, \ldots, L_p$ with parameters g (the check size), m (the check number) and n (the code length). Our goal is to construct codes with constant check-size and δ while keeping n and m polynomial in p.

Holographic codes for a set of linear functions $L_1, \ldots, L_p : \mathbf{F}^k \to \mathbf{F}$ are interesting when there are dependencies among the L_is. Indeed, assume there are no dependencies. Define E by $E(\mathbf{y}) = (L_1(\mathbf{y}), \ldots, L_p(\mathbf{y}))$ for every $\mathbf{y} \in \mathbf{F}^k$. Let W_i $(1 \leq i \leq p)$ be the distribution $I(i)$ which is concentrated entirely on the i^{th} coordinate of $E(\mathbf{y})$. Let D be any function which takes a vector $\mathbf{v} \in \mathbf{F}^p$ to some $\mathbf{y} \in \mathbf{F}^k$ such that $E(\mathbf{y}) = \mathbf{v}$ (there exists such a

decoding function, since by our assumption $L_1, \ldots, L_p$ are linearly independent). Then E is a 1-holographic code with the best possible parameters $(g = 1, m = n = p)$.

Let us also notice that if a system of linear functions $L_1, \ldots, L_p : \mathbf{F}^k \to \mathbf{F}$ has the exact same set of linear dependencies as another system $L'_1, \ldots, L'_p : \mathbf{F}^{k'} \to \mathbf{F}$ and there is a δ-holographic code for the first system, then there is a δ-holographic code for the second with exactly the same parameters g, n and m (in fact the two codes are the same, only the encoding and decoding functions differ). The later remark shows the independence of the construction on k.

5.3.1 The Holographic Property of the Hadamard Code

Lemma 5.49 *The Hadamard code over* $\mathbf{F}_2$ *is 1/12-holographic with respect to the set of* all *linear functions over* $\mathbf{F}_2^n$.

Proof. For a linear form $L_{\mathbf{w}} = \langle x, \mathbf{w} \rangle$ let $W_{\mathbf{w},\mathbf{r}_1,\mathbf{r}_2}$ $\mathbf{r}_1, \mathbf{r}_2 \in \mathbf{F}_2^n$ be the weight function which allocates weight 1/4 to each element of

$$\{\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_1 + \mathbf{w}, \mathbf{r}_1 + \mathbf{r}_2\}$$

(recall that the index set of the Hadamard code is $\mathbf{F}_2^n$). Our decoding function D will simply be the minimum distance decoding function.

D clearly satisfies Property 1 of Definition 5.48. We show that it also satisfies Property 2. Let $\mathbf{v} \in \mathbf{F}_2^{2^n}$ be an arbitrary word such that $D(\mathbf{v}) = \mathbf{y}^{(0)}$. We separate two cases:

1. $dist(E(\mathbf{y}^{(0)}), \mathbf{v}) > 1/3$. By the definition of D (one of) the closest code word of the Hadamard code to $\mathbf{v}$ is $\mathbf{y}^{(0)}$, so by Theorem 5.28 for at least 1/3 of the $\mathbf{r}_1, \mathbf{r}_2 \in \mathbf{F}_2^n$ pairs $\mathbf{v}_{\mathbf{r}_1} + \mathbf{v}_{\mathbf{r}_2} + \mathbf{v}_{\mathbf{r}_1+\mathbf{r}_2} \neq 0$. For each such pair $\mathbf{v}$ looks different from any code word on the set $\{\mathbf{r}_1, \mathbf{r}_2, \mathbf{r}_1 + \mathbf{r}_2\}$. For the triplets $\mathbf{w}, \mathbf{r}_1, \mathbf{r}_2$ where $\mathbf{r}_1, \mathbf{r}_2$ as above we have:

$$\min_{\mathbf{z}=E(\mathbf{y})} dist_{W_{\mathbf{w},\mathbf{r}_1,\mathbf{r}_2}}(\mathbf{z}, \mathbf{v}) > 1/4.$$

   The 1/12 lower bound for $\mathbf{E}_{\mathbf{w},\mathbf{r}_1,\mathbf{r}_2} \min_{\mathbf{z}=E(\mathbf{y})} dist_{W_{\mathbf{w},\mathbf{r}_1,\mathbf{r}_2}}(\mathbf{z}, \mathbf{v})$ then follows.

2. $dist(E(\mathbf{y}^{(0)}), \mathbf{v}) \leq 1/3$. Let $\mathbf{z}^{(0)} = E(\mathbf{y}^{(0)})$ Clearly, the probability over a random $\mathbf{r}_1$ that one of $\mathbf{v}_{\mathbf{r}_1} \neq \mathbf{z}^{(0)}_{\mathbf{r}_1}$ and $\mathbf{v}_{\mathbf{r}_1+\mathbf{w}} \neq \mathbf{z}^{(0)}_{\mathbf{r}_1+\mathbf{w}}$ holds

is at most 2/3, so with probability 1/3 we have that $\mathbf{v}_{\mathbf{r}_1} = \mathbf{z}^{(0)}_{\mathbf{r}_1}$ and $\mathbf{v}_{\mathbf{r}_1+\mathbf{w}} = \mathbf{z}^{(0)}_{\mathbf{r}_1+\mathbf{w}}$. Let $\mathbf{y} \in \mathbf{F}_2^n$ be such that $\langle \mathbf{y}^{(0)}, \mathbf{w} \rangle \neq \langle \mathbf{y}, \mathbf{w} \rangle$. Then for $\mathbf{z} = E(\mathbf{y})$ either $\mathbf{z}_{\mathbf{r}_1} \neq \mathbf{z}^{(0)}_{\mathbf{r}_1}$ or $\mathbf{z}_{\mathbf{r}_1+\mathbf{w}} \neq \mathbf{z}^{(0)}_{\mathbf{r}_1+\mathbf{w}}$. Thus for all $\mathbf{w}, \mathbf{r}_1, \mathbf{r}_2$ triplets where the pair $\mathbf{w}, \mathbf{r}_1$ is as above:

$$\min_{\mathbf{z}=E(\mathbf{y}), \langle \mathbf{y}^{(0)}, \mathbf{w} \rangle \neq \langle \mathbf{y}, \mathbf{w} \rangle} dist_{W_{\mathbf{w},\mathbf{r}_1,\mathbf{r}_2}}(\mathbf{z}, \mathbf{v}) > 1/4.$$

Therefore the expectation of these minimum distances is again 1/12.

□

5.3.2 Composition of holographic codes over a field F.

For a linear function $E : \mathbf{F}^k \to \mathbf{F}^n$ every coordinate function $E_i : \mathbf{F}^k \to \mathbf{F}$ which maps a $\mathbf{y} \in \mathbf{F}^k$ to the i^{th} coordinate of $E(\mathbf{y})$ is a linear function. Since holographic decoding takes the advantage of linear dependencies in between small subsets of coordinate functions, it is natural to try to encode these subsets again in the hope of reducing the check-size.

Theorem 5.50 (Composition of holographic codes) *Let E be a δ-holographic code as in Definition 5.48 with parameters g, m and n. Let $E_{i,j}$ be a δ_1-holographic encoding of*

$$\{L_i\} \cup \{E_{j_1} \mid j_1 \in supp(W_{i,j})\}$$

with check-size g_1, check number m_1 and code length n_1. The code $E'(\mathbf{y}) = E(\mathbf{y}) \oplus \bigoplus_{1 \leq i \leq m, 1 \leq j \leq p} E_{i,j}(\mathbf{y})$ is a $\delta\delta_1/4$-holographic code with dimension $n' = n + mpn_1$, check size $g' = g_1 + 1$ and check number $m' = 2mgm_1$.

Proof: Let $W_{i,j,0,l}$ $(1 \leq l \leq m_1)$ be the checks for $E_{i,j}$ belonging to L_i with distribution $Q_{i,j,0}$, and let $W_{i,j,j_1,l}$ $(j_1 \in supp(W_{i,j}); 1 \leq l \leq m_1)$ be the checks for $E_{i,j}$ belonging to E_{j_1} with distribution Q_{i,j,j_1}.

The checks that belong to the compound code E' are defined by

$$W'_{i,j,j_1,l} = 1/2I(j_1) + 1/2W_{i,j,j_1,l}, \quad (65)$$

$$W'_{i,j,0,l} = W_{i,j,0,l} \quad (66)$$

for $1 \leq i \leq p, 1 \leq j \leq m, j_1 \in W_{i,j}$ and $1 \leq l_1 \leq m_1$. The distribution Q'_i on the checks indexed by the tuples (i, j, j_1, l) and $(i, j, 0, l)$ for the compound

code is defined by:

$$Prob(i,j,j_1,l) = \frac{1}{2}Prob(j \in Q_i)Prob(j_1 \in W_{i,j})Prob(l \in Q_{i,j,j_1}), \quad (67)$$

$$Prob(i,j,0,l) = \frac{1}{2}Prob(j \in Q_i)Prob(l \in Q_{i,j,0}). \quad (68)$$

Clearly, the support size of each check is at most $g_1 + 1$ and Q_i' is a probability distribution. The decoding D' of E' on a vector $\mathbf{v}$ is defined as $D(\mathbf{v}|\{1,\ldots,n\})$, that is we decode on the first n coordinates exactly in the same way as we decode for E (we disregard all the other coordinates).

Property 1. of Definition 5.48 obviously holds for E'. In order to show Property 2. let us fix $1 \leq i \leq p$ and $\mathbf{v} \in \mathbf{F}^{n+mpn_1}$. Let $D'(\mathbf{v}) = \mathbf{y}$, $L_i(\mathbf{y}) = a$ and

$$A = \mathbf{E}_{j \in Q_i, j_1 \in W_{i,j}, l \in Q_{i,j,j_1}} \left(\min_{\mathbf{z}=E'(\mathbf{y}):L_i(\mathbf{y}) \neq a} dist_{W'_{i,j,j_1,l}}(\mathbf{z},\mathbf{v}) \right)$$

$$B = \mathbf{E}_{j \in Q_i, l \in Q_{i,j,0}} \left(\min_{\mathbf{z}=E'(\mathbf{y}):L_i(\mathbf{y}) \neq a} dist_{W'_{i,j,0,l}}(\mathbf{z},\mathbf{v}) \right)$$

What we have to show is that $A/2 + B/2 \geq \delta\delta_1/4$. Assume the opposite. We prove that in this case Equation 64 of Definition 5.48 for the code E does not hold thus arriving at a contradiction.

Since $dist_{W'_{i,j,j_1,l}}(\mathbf{z},\mathbf{v}) = 1/2 dist_{I(j_1)}(\mathbf{z},\mathbf{v}) + 1/2 dist_{W_{i,j,j_1,l}}(\mathbf{z},\mathbf{v})$, we have that for a fixed $\mathbf{z} = E(\mathbf{y})$ either $dist_{W'_{i,j,j_1,l}}(\mathbf{z},\mathbf{v}) \geq 1/2$ or $E_{j_1}(\mathbf{y}) = \mathbf{v}_{j_1}$. Thus

$$\frac{1}{2} \min_{\mathbf{z}=E'(\mathbf{y}):E_{j_1}(\mathbf{y})=\mathbf{v}_{j_1}} dist_{W_{i,j,j_1,l}}(\mathbf{z},\mathbf{v}) \leq \min_{\mathbf{z}=E'(\mathbf{y})} dist_{W'_{i,j,j_1,l}}(\mathbf{z},\mathbf{v}) \leq \min_{\mathbf{z}=E'(\mathbf{y}):L_i(\mathbf{y}) \neq a} dist_{W'_{i,j,j_1,l}}(\mathbf{z},\mathbf{v}).$$

From the above we obtain that the probability of the event

$$S_1^i = \left\{ (j,j_1) \mid \mathbf{E}_{l \in Q_{i,j,j_1}} \left(\min_{\mathbf{z}=E'(\mathbf{y}):E_{j_1}(\mathbf{y})=\mathbf{v}_{j_1}} dist_{W_{i,j,j_1,l}}(\mathbf{z},\mathbf{v}) \right) \geq \delta_1 \right\} \quad (69)$$

is at most $2A/\delta_1$. The probability of the event

$$S_2^i = \left\{ j \mid \mathbf{E}_{l \in Q_{i,j,0}} \left(\min_{\mathbf{z}=E'(\mathbf{y}):L_i(\mathbf{y}) \neq a} dist_{W_{i,j,0,l}}(\mathbf{z},\mathbf{v}) \right) \geq \delta_1 \right\} \quad (70)$$

is at most B/δ_1. Let $\chi_{S_2^i}$ be the characteristic function of S_2^i. We now look at $\mathbf{y}^{(i,j)}$, the decoding of $\mathbf{v}$ through $D_{i,j}$! By the holographic property of $E_{i,j}$ with respect to L_i we get that for any $j \notin S_2^i$ it holds that $L_i(\mathbf{y}^{(i,j)}) \neq a$. Thus for every $1 \leq i \leq p$ we have that for every $1 \leq j \leq m$:

$$\min_{\mathbf{z}=E(\mathbf{y}):L(\mathbf{y})\neq a} dist_{W_{i,j}}(\mathbf{z},\mathbf{y}) \leq (1-\chi_{S_2^i}(j))dist_{W_{i,j}}(E'(\mathbf{y}^{(i,j)}),\mathbf{v}) + \chi_{S_2^i}(j) \tag{71}$$

Using the holographic property of $E_{i,j}$ again, this time with respect to E_{j_1}, we get that for any $(j,j_1) \notin S_1^i$ we have $E_{j_1}(\mathbf{y}^{(i,j)}) = \mathbf{v}_{j_1}$, since $\mathbf{E}_{l\in Q_{i,j,j_1}}\left(\min_{\mathbf{z}=E'(\mathbf{y}):E_{j_1}(\mathbf{y})=\mathbf{v}_{j_1}} dist_{W_{i,j,j_1,l}}(\mathbf{z},\mathbf{v})\right) < \delta_1$ excludes $E_{j_1}(\mathbf{y}^{(i,j)}) = b$ for any $b \neq \mathbf{v}_{j_1}$. If for a fixed i we take the expectation of both sides or Equation 71 over $j \in Q_i$ we obtain:

$$\mathbf{E}_{j\in Q_i}\left(\min_{\mathbf{z}=E(\mathbf{y}):L(\mathbf{y})\neq a} dist_{W_{i,j}}(\mathbf{z},\mathbf{y})\right) \leq 2A/\delta_1 + B/\delta_1 < \delta,$$

a contradiction.

5.3.3 Changing the Field

In this section we give a construction for a holographic code over a field $\mathbf{F}$ if we already have a holographic code for an extension $\mathbf{K}$ of $\mathbf{F}$. The reverse is rather straightforward. If E is a δ-holographic encoding over $\mathbf{F}$ with respect to a set of linear functions then it is also a δ-holographic encoding over $\mathbf{K}$ for the same set of linear functions (a linear encoding extends from $\mathbf{F}$ to $\mathbf{K}$ via the matrix that defines it). If $d = \dim_{\mathbf{F}} \mathbf{K}$ then the new code can be looked at as d copies of the original code, and the decoding is done for each copy separately.

In the reverse direction we would like to encode every element of $\mathbf{K}$ as a vector over $\mathbf{F}$, but the natural correspondence $\phi : \mathbf{K} \to \mathbf{F}^d$ (provided by Lemma 5.4) may lead us to a substantial shrinkage of normalized Hamming distances. When we encode a vector $(x_1,\ldots,x_l)$ over $\mathbf{K}$ as a vector $(\phi(x_1),\ldots,\phi(x_l))$ over $\mathbf{F}$, the later vector has length ld, but may have only as many non-zero coordinates as the former. Therefore the normalized Hamming distance of $(\phi(x_1),\ldots,\phi(x_l))$ from $\mathbf{0}$ can be d times smaller than the normalized Hamming distance of $(x_1,\ldots,x_l)$ from $\mathbf{0}$.

To remedy this problem we turn to error correcting codes: Let us compose ϕ with an error correcting encoding $\psi : \mathbf{F}^d \to \mathbf{F}^{d'}$ such that the resulting code is a (d',d,cd') code for some constant c. Let $\phi' = \phi \circ \psi$. If we encode

coordinates of vectors via ϕ' instead of ϕ then distances may shrink with a factor at most c. However, what we really need to worry about is decoding, not encoding.

Definition 5.51 *Let E be a δ-holographic encoding of $\mathbf{K}^k$ to $\mathbf{K}^n$ with respect to a set of linear functions whose coefficients come from $\mathbf{F} \subseteq \mathbf{K}$. From E we define a holographic encoding E' over $\mathbf{F}$ (with respect to the same set of linear functions) such that we encode a vector $\mathbf{y} \in \mathbf{F}^k$ by applying E on $\mathbf{y}$ and then ϕ' on the coordinates of $E(\mathbf{y})$ separately.*

We need to provide a decoding function for E'. The first attempt works: we first decode the blocks of the new code with the shortest distance decoding. This way we get a vector whose elements are from $\mathbf{K}$ and has just the right length to apply the decoding function of the original code on it. The resulting vector, $\mathbf{y}^*$ is over $\mathbf{K}$, and not necessary over $\mathbf{F}$. For the entire decoding procedure we arbitrarily decide at an $\mathbf{F}$-linear $\pi : \mathbf{K} \to \mathbf{F}$, which fixes the elements of $\mathbf{F}$ (such a map is called a projection). We map the coordinates of $\mathbf{y}^*$ using π to obtain $\mathbf{y}$, the end result of our three step decoding procedure. We also need to define the new $W_{i,j}$s. Let $W'_{i,j}$ be the direct product of $W_{i,j}$ with the uniform distribution on $[1, d']$. The holographic property of the new code hinges upon the following lemma:

Lemma 5.52 *Let $\mathbf{K}$ be a d dimensional extension of $\mathbf{F}$, which we view as $\mathbf{F}^d$. Let C be a (d', d, cd') code over $\mathbf{F}$, and $\mathbf{z}, \mathbf{v} \in \mathbf{K}^l$. Let $\mathbf{z}' = (\mathbf{z}^1, \ldots, \mathbf{z}^l)$ be the coordinate-wise encoding of $\mathbf{z}$ with the above code, and let $\mathbf{v}' = (\mathbf{v}^1, \ldots, \mathbf{v}^l)$, be another vector in $\mathbf{F}^{ld'}$ ($\mathbf{z}^i, \mathbf{v}^i \in \mathbf{F}^{d'}$ for $1 \leq i \leq l$). Assume furthermore that the coordinate-wise shortest distance decoding of $\mathbf{z}'$ results in $\mathbf{z}$. Let p be an arbitrary probability distribution on $[1, l]$ and p' be the the product of p with the uniform distribution on $[1, d']$. Then:*

$$dist_{p'}(\mathbf{z}', \mathbf{v}') \geq \frac{c}{2} dist_p(\mathbf{z}, \mathbf{v}). \tag{72}$$

Proof. (Sketch) Because of the shift-invariance of the shortest degree decoding with respect to code words, it is sufficient to prove the above lemma when $\mathbf{v} = \mathbf{0}$. Then the proof then comes almost immediately from the definitions. □

Let $\mathbf{v}' \in \mathbf{F}^{nd'}$ be arbitrary, and let $\mathbf{v}$ be its block-wise shortest distance decoding. Let $a = L_i(D(\mathbf{v}))$. By the δ-holographic property of E:

$$\mathbf{E}_{j\in Q_i}\left(\min_{\mathbf{z}=E(\mathbf{y}^*):L_i(\mathbf{y}^*)\neq a} dist_{W_{i,j}}(\mathbf{z},\mathbf{v})\right)\geq\delta \tag{73}$$

Lemma 5.52 together with Inequality (73) imply that:

$$\mathbf{E}_{j\in Q_i}\left(\min_{\mathbf{z}'=E'(\mathbf{y}):L_i(\mathbf{y})\neq\pi(a)} dist_{W'_{i,j}}(\mathbf{z}',\mathbf{v}')\right)\geq\frac{c}{2}\delta. \tag{74}$$

Observe that in (73) $\mathbf{y}^*$ was taken from $\mathbf{K}^k$ and in (74) $\mathbf{y}$ was taken from $\mathbf{F}^k$, a smaller domain, and that $L_i(\mathbf{y})\neq\pi(a)$ is a stricter condition than $L_i(\mathbf{y})\neq a$ for $\mathbf{y}\in\mathbf{F}^k$. These remarks are all needed to see that Equation (74) holds. This equation means that we have just constructed a new holographic code, whose properties we summarize in the following lemma:

Lemma 5.53 (Field Size Changing Lemma) *Let E be a δ-holographic code over a field $\mathbf{K}$ with respect to a set of linear functions $L_1,\ldots,L_p$ whose coefficients come from a subfield $\mathbf{F}$ of $\mathbf{K}$. Let d be the dimension of $\mathbf{K}$ over $\mathbf{F}$. Suppose, there exists a (d',d,cd') linear error correcting code over $\mathbf{F}$. Then we can construct a $c\delta/2$-holographic code E' with respect to $L_1,\ldots,L_p$ over $\mathbf{F}$. If the parameters of E' are g, m, and n, the parameters of E are gd', m, and nd'.*

5.3.4 The Holographic Property of Polynomial Codes

We will say that a code is δ-holographic without specifying the set of linear functions for which the code is built if it is δ-holographic with respect to the coordinate functions (i.e. the individual bits) of the code. This holds for instance in the case of the Hadamard code (See Section 5.3.1), and many other codes. In this section we show that this is the case for all checkable subspaces of the code formed by the tables of all total degree at most d polynomials.

Lemma 5.54 *Let $S\leq\mathbf{F}^{deg\leq d}[x_1,\ldots,x_n]$, where $\mathbf{F}$ is a finite field, and assume that the ϵ-neighborhood of S is δ-checkable with check-size g_1. Assume that $\epsilon\leq 1-\frac{d+1}{|\mathbf{F}|}-\delta$. Then S is $\delta/2$-holographic with check size $g+|\mathbf{F}|$.*

Proof. We define the decoding function D as the minimum-distance decoding. Clearly Property 1 of the definition for holographic codes holds. We need to prove Property 2. Let $V\in\mathbf{F}^{\mathbf{F}^n}$, and $\mathbf{x}\in\mathbf{F}^n$. With probability

1/2 we check if V is ϵ-close to any member of S and with probability 1/2 we cast a random line through $\mathbf{x}$. If V is not ϵ-close to any member of S, then the first part of the check will have expected distance at least δ, but since we put only half measure on these check points, the expectation reduces to $\delta/2$. If V is ϵ-close to some $P \in S$ then let $V = P + E$, where $dist(E, \mathbf{0}) \leq \epsilon$. In particular, the correct decoding of V "on $\mathbf{x}$" is $P(\mathbf{x})$. Hence:

$$\begin{aligned} \mathbf{E}_{\mathcal{L} \in \Lambda_x} \left(\min_{Q \in S, Q(\mathbf{x}) \neq V(\mathbf{x})} dist_{\mathcal{L}}(V, Q) \right) &\geq \\ \mathbf{E}_{\mathcal{L} \in \Lambda_x} \left(\min_{Q \in S, Q(\mathbf{x}) \neq P(\mathbf{x})} dist_{\mathcal{L}}(P, Q) - dist_{\mathcal{L}}(V, P) \right) &\geq \\ 1 - \frac{d}{|\mathbf{F}|} - \epsilon - \frac{1}{|\mathbf{F}|}. & \end{aligned}$$

Since this part of the distance also get weight 1/2 we get a $(1 - \frac{d+1}{|\mathbf{F}|} - \epsilon)/2$ lower bound in the second case, and the lemma follows. □

To encode an arbitrary set $\{L_1 \ldots, L_p\}$ of linear functions we shall use the checkable subspace in Section 5.2.5 constructed exactly for this purpose. Consider the space S_g in than section. It encodes an arbitrary function $f : X^n \to \mathbf{F}$ (whose values are viewed as variables over $\mathbf{F}$) in such a way that every sum

$$\sum_{\mathbf{x} \in X^n} f(\mathbf{x}) g(\mathbf{x}, \mathbf{y})$$

occurs as a coordinate function in $P_{f'g'}$, namely it equals to $P_{f'g'}(\mathbf{0}, \mathbf{y}, \mathbf{0})$. Thus, if we combine Theorem 5.46 with Lemma 5.54 and scale down the parameters appropriately, then we get the following:

Corollary 5.55 *There are $\delta > 0$, $c, c_1 > 0$ that the following holds. Let $\{L_1 \ldots, L_p\}$ be linear functions from $\mathbf{F}^k$ to $\mathbf{F}$ $(p \geq k)$, and let $\mathbf{F} \geq c_1 \log^c p$. then There is a δ-holographic code for $\{L_1 \ldots, L_p\}$ with parameters n, m and g, where $n, m \leq p^c$ and $g \leq |\mathbf{F}|^c$.*

5.3.5 Compound Holographic Codes

The composition theorem (Theorem 5.50) for holographic codes allows us to construct holographic codes with good parameters. Recall that our goal is to achieve constant check-size while keeping all other parameters polynomial.

Lemma 5.56 *There are $\delta > 0$, $c > 0$ that the following holds. Let $\{L_1 \ldots, L_p\}$ be linear functions from $\mathbf{F}_2^k$ to $\mathbf{F}_2$ ($p \geq k$). There is a δ-holographic code for $\{L_1 \ldots, L_p\}$ with parameters n, m and g, where $n, m \leq p^c$ and $g \leq c$.*

Proof. We apply Corollary 5.55 of the previous section to build a code for $\{L_1 \ldots, L_p\}$ over an extension $\mathbf{K}$ of $\mathbf{F_2}$ with parameters $|\mathbf{K}| = \log^{c_1} p$ $n_1, m_1 \leq p^{c_1}$ and $g_1 \leq \log^{c_1} p$. After changing the field we apply a polynomial code construction over the smaller field to encode the check sets, and apply the composition lemma again. We need this step to make the check size logarithmic (actually it can be made as small as $\log^{c_2} \log p$ for some constant c_2). The other parameters remain polynomial. Then we change the field to $\mathbf{F_2}$ (note that $\mathbf{F_2}$ is a subfield of any subfield of $\mathbf{K}$). In a new application of the code composition we reduce the check size to constant using the Hadamard encoding for each check set of the large code. All parameters remain polynomial in p, and the holographic parameter remains a constant.□

5.3.6 Decoding Holographic Codes

In this section we will rely on the notations of Definition 5.48 in Section 5.3. Let E be a holographic code for a set of linear functions $\{L_1 \ldots, L_p\}$, D be the decoding function associated with it, $\mathbf{w} \in \mathbf{F}^n$ be an arbitrary vector of length n, which we view as an oracle with random access to its entries. Let $\mathbf{y}_0 = D(\mathbf{w})$. Our goal is to design a randomized decoding procedure P that with some positive probability either outputs the value of $L_i(\mathbf{y}_0)$ for a desired $1 \leq i \leq k$ or outputs an error message. However, when $\mathbf{w} = E(\mathbf{y}_0)$ it always outputs $L_i(\mathbf{y}_0)$ correctly. We show this is possible to achieve with a procedure which looks only at most $2g$ bits of $\mathbf{w}$.

For the first attempt we may design a procedure like this: Fix the desired i. Pick $1 \leq j \leq m$ according to distribution Q_i, and try to recover $L_i(\mathbf{y}_0)$ from $supp(W_{i,j})$. Let $\mathbf{x}|supp(W_{i,j})$ denote the restriction of a vector $\mathbf{x} \in \mathbf{F}^n$ to $supp(W_{i,j})$. This restriction we call a view. The procedure rejects any view that is not of the form $E(\mathbf{y})|supp(W_{i,j})$ for some $\mathbf{y} \in \mathbf{F}^k$. If the view is of this form the procedure selects an arbitrary $\mathbf{y}^j$ (following an arbitrarily fixed rule) such that $E(\mathbf{y}^j)|supp(W_{i,j})$, coincides with $E(\mathbf{w}|supp(W_{i,j})$ and outputs $L_i(\mathbf{y}^j)$. Let

$$A = Prob_{j \in Q_i}(P \text{ outputs "Reject" when viewing } W_{i,j}), \quad (75)$$

$$B = Prob_{j \in Q_i}(P \text{ outputs } L_i(\mathbf{y}_0) \text{ when viewing } W_{i,j}) \quad (76)$$

We claim that $A + B \geq \delta$. Indeed, if for some j the procedure does not output "Reject" and $L_i(\mathbf{y}^j)$ is not equal to $L_i(\mathbf{y}_0) = a$ then

$$\min_{\mathbf{z}=E(\mathbf{y}):L_i(\mathbf{y})\neq a} dist_{W_{i,j}}(\mathbf{z}, \mathbf{w}) \leq dist_{W_{i,j}}(E(\mathbf{y}^j), \mathbf{w}) = 0,$$

and our claim follows from the δ-holographic property of E, which requires that

$$\mathbf{E}_{j\in Q_i}\left(\min_{\mathbf{z}=E(\mathbf{y}):L_i(\mathbf{y})\neq a} dist_{W_{i,j}}(\mathbf{z}, \mathbf{w})\right) \geq \delta.$$

Consider now the case when $\mathbf{w} = E(\mathbf{y}_0)$. It may occur that $\mathbf{w}|supp(W_{i,j})$ coincides with $E(\mathbf{y}^j)|supp(W_{i,j})$, but $L_i(\mathbf{y}^j) \neq L_i(\mathbf{y}_0)$. This is a problem as long as we would like to decode $L_i(\mathbf{y}_0)$ from $E(\mathbf{y}_0)$ with zero error. We can remedy this problem using the next lemma.

Lemma 5.57 *If $\delta > 0$ there is at least one $1 \leq j \leq m$ for which $L_i(\mathbf{y}_0)$ is uniquely determined by $E(\mathbf{y}_0)|supp(W_{i,j})$.*

Proof. Since the map $\mathbf{y} \to E(\mathbf{y})|supp(W_{i,j})$ is linear, there are only two possibilities: either $L_i(\mathbf{y})$ is a linear function from the linear space formed by all views of $E(\mathbf{y})$ ($\mathbf{y} \in \mathbf{F}^k$) on $supp(W_{i,j})$ to $\mathbf{F}$ or for all $\mathbf{y} \in \mathbf{F}^k$ and $b \in \mathbf{F}$ there is a $\mathbf{y}' \in \mathbf{F}^k$ such that $E(\mathbf{y})|supp(W_{i,j}) = E(\mathbf{y}')|supp(W_{i,j})$ and $L_i(\mathbf{y}') = b$. The later cannot be the case for all j, otherwise the holographic property of E with respect to L_i would be violated for every codeword. □

Let $1 \leq j_0 \leq m$ be a number for which Lemma 5.57 holds. We can modify our decoding procedure such that for any $j \in Q_i$ it views $supp(W_{i,j_0}) \cup supp(W_{i,j})$. This way the decoding is unique, since $L_i(\mathbf{y})$ is a linear function from the linear space formed by all possible views of $E(\mathbf{y})$ when $\mathbf{y}$ runs through $\mathbf{F}^k$ on $supp(W_{i,j_0}) \cup supp(W_{i,j})$ to $\mathbf{F}$. If the code has an explicit construction then this map is computable in polynomial time. We can now state the decoding lemma for holographic codes:

Lemma 5.58 (Holographic Decoding) *For any δ-holographic code E with parameters g, m, and n with respect to $\{L_1 \ldots, L_p\}$ with an associated decoding function D there is a randomized decoding procedure $P^{\mathbf{w}}(r, i)$, which inputs a number $1 \leq i \leq k$, selects a random number $r \in Q$, reads $2g$ entries of $\mathbf{w} \in \mathbf{F}^n$ and outputs an element of $\mathbf{F}$ (supposed to be, but not always equals to $L_i(D(\mathbf{w}))$) or a symbol* "Reject," *such that:*

1. *If $\mathbf{w} = E(\mathbf{y})$ then the probability that P on input i outputs $L_i(\mathbf{y})$ is 1.*

2. *For any* $\mathbf{w} \in \mathbf{F}^n$ *and for every* $1 \leq i \leq k$*:*

$$\begin{aligned} &Prob_{j \in Q_i}(P \text{ outputs "Reject" when viewing } W_{i,j}) \quad + \\ &Prob_{j \in Q_i}(P \text{ outputs } L_i(D(\mathbf{w})) \text{ when viewing } W_{i,j}) \quad \geq \quad \delta \end{aligned}$$

Moreover, if the code is explicitly given, the procedure runs in time polynomial in $n \log mp$.

References

[1] M. Ajtai, J. Komlós, E. Szemeredi, Deterministic simulation in logspace, *Proceedings of 19th Annual Symp. on the Theory of Computing, ACM,* pp 132-140, (1987)

[2] N. Alon, U. Feige, A. Wigderson, D. Zuckerman, Derandomized graph products, *Computational Complexity* (1995) pp. 60-75.

[3] N. Alon, E. Fischer and M. Krivelevich, M. Szegedy "Logical Property Testing", *FOCS 1999*

[4] N. Alon, M. Krivelevich, I. Newman, M. Szegedy "Regularity Testing", *FOCS 1999*

[5] S. Arora, *Probabilistic Checking of Proofs and Hardness of Approximation Problems,* PhD thesis, U.C. Berkeley, 1994. Available from `http://www.cs.princeton.edu/~arora` .

[6] S. Arora, Polynomial-time approximation schemes for Euclidean TSP and other geometric problems. *Proceedings of 37th IEEE Symp. on Foundations of Computer Science,* pp 2-12, 1996.

[7] S. Arora, L. Babai, J. Stern, and Z. Sweedyk, The hardness of approximate optima in lattices, codes, and systems of linear equations. *Journal of Computer and System Sciences,* 54(2):317-331, April 1997.

[8] S. Arora, D. Karger, M. Karpinski, Polynomial Time Approximation Schemes for Dense Instances of NP-hard Problems, *J. Comput. System Sci.* 58 (1999), pp. 193-210.

[9] S. Arora and C. Lund, Hardness of approximations. In *Approximation Algorithms for NP-hard problems,* D. Hochbaum, ed. PWS Publishing, 1996.

[10] S. Arora, C, Lund, R. Motwani, M. Sudan, and M. Szegedy, Proof verification and the intractability of approximation problems. *Journal of the ACM,* 45(3):501–555, 1998. Preliminary version in *Proc. of FOCS'92.*

[11] S. Arora, R. Motwani, S. Safra, M. Sudan, and M. Szegedy, PCP and approximation problems. *Unpublished note,* 1992.

[12] S. Arora and S. Safra. Probabilistic checking of proofs: A new characterization of NP. *Journal of the ACM,* 45(1):70–122, 1998. Preliminary version in *Proc. of FOCS'92.*

[13] S. Arora and M. Sudan. Improved low degree testing and its applications. In *Proceedings of the 29th ACM Symposium on Theory of Computing,* pages 485–495, 1997.

[14] G. Ausiello, A. D'Atri, and M. Protasi, Structure Preserving Reductions among Convex Optimization Problems. *Journal of Computer and Systems Sciences,* 21:136-153, 1980.

[15] G. Ausiello, P. Crescenzi, G. Gambosi, V. Kann, A. Marchetti-Spaccamela, M. Protasi, *Complexity and Approximation,* Springer Verlag, ISBN 3-540-65431-3.

[16] G. Ausiello, A. Marchetti-Spaccamela and M. Protasi, Towards a Unified Approach for the Classification of NP-complete Optimization Problems. *Theoretical Computer Science,* 12:83-96, 1980.

[17] L. Babai, Trading group theory for randomness. *Proceedings of the Seventeenth Annual Symposium on the Theory of Computing,* ACM, 1985.

[18] L. Babai E-mail and the unexpected power of interaction, *Proc. 5th IEEE Structure in Complexity Theory conf.*, Barcelona 1990, pp. 30-44.

[19] L. Babai, Transparent (holographic) proofs. *Proceedings of the Tenth Annual Symposium on Theoretical Aspects of Computer Science,* Lecture Notes in Computer Science Vol. 665, Springer Verlag, 1993.

[20] L. Babai, Transparent proofs and limits to approximation. *Proc. First European Congress of Mathematics* (1992), Vol. I, Birkhäuser Verlag, 1994, pp. 31–91.

[21] L. Babai and L. Fortnow, Arithmetization: a new method in structural complexity theory. *Computational Complexity,* 1:41-66, 1991.

[22] L. Babai, L. Fortnow, and C. Lund, Non-deterministic exponential time has two-prover interactive protocols. *Computational Complexity,* 1:3-40, 1991.

[23] L. Babai, L. Fortnow, L. Levin, and M. Szegedy, Checking computations in polylogarithmic time. *Proceedings of the Twenty Third Annual Symposium on the Theory of Computing,* ACM, 1991.

[24] L. Babai and K. Friedl, On slightly superlinear transparent proofs. *Univ. Chicago Tech. Report,* CS-93-13, 1993.

[25] L. Babai and S. Moran, Arthur-Merlin games: a randomized proof system, and a hierarchy of complexity classes. *Journal of Computer and System Sciences,* 36:254-276, 1988.

[26] Bar-Yehuda, R., and Even, S. (1985), A local-ratio theorem for approximating the weighted vertex cover problem, Analysis and Design of Algorithms for Combinatorial Problems, *volume 25 of Annals of Disc. Math.,* Elsevier science publishing company, Amsterdam, 27-46.

[27] D. Beaver and J. Feigenbaum, Hiding instances in multioracle queries. *Proceedings of the Seventh Annual Symposium on Theoretical Aspects of Computer Science,* Lecture Notes in Computer Science Vol. 415, Springer Verlag, 1990.

[28] M. Bellare. Interactive proofs and approximation: reductions from two provers in one round. *Proceedings of the Second Israel Symposium on Theory and Computing Systems,* 1993.

[29] M. Bellare. Proof checking and approximation: Towards tight results. *Sigact News,* 27(1), 1996.

[30] M. Bellare, D. Coppersmith, J. Håstad, M. Kiwi and M. Sudan, Linearity testing in characteristic two. *IEEE Transactions on Information Theory* 42(6):1781-1795, November 1996.

[31] M. Bellare, O. Goldreich and M. Sudan, Free bits, PCPs and non-approximability — towards tight results. To appear *SIAM Journal on*

Computing, SIAM Journal on Computing, 27(3):804–915, 1998. Preliminary version in *Proc. of FOCS'95.* Full version available as TR95-024 of ECCC, the *Electronic Colloquium on Computational Complexity,* `http://www.eccc.uni-trier.de/eccc/`.

[32] M. Bellare, S. Goldwasser, C. Lund, and A. Russell, Efficient probabilistically checkable proofs. *Proceedings of the Twenty Fifth Annual Symposium on the Theory of Computing,* ACM, 1993. (See also Errata sheet in *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing,* ACM, 1994).

[33] M. Bellare and P. Rogaway, The complexity of approximating a non-linear program. *Journal of Mathematical Programming B,* 69(3):429-441, September 1995. Also in *Complexity of Numerical Optimization,* Ed. P. M. Pardalos, World Scientific, 1993.

[34] M. Bellare and M. Sudan, Improved non-approximability results. *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing,* ACM, 1994, pages 184–193.

[35] M. Ben-Or, S. Goldwasser, J. Kilian, and A. Wigderson, Multi-prover interactive proofs: How to remove intractability assumptions. *Proceedings of the Twentieth Annual Symposium on the Theory of Computing,* ACM, 1988.

[36] Berman, P., and Karpinski, M., On some tighter inapproximability results, *Proc. 26th ICALP,* pages 200-209 (1999); also available as Technical Report TR98-029, ECCC.

[37] P. Berman and G. Schnitger. On the complexity of approximating the independent set problem. *Information and Computation* 96:77–94, 1992.

[38] M. Bern and P. Plassmann, The steiner problem with edge lengths 1 and 2. *Information Processing Letters,* 32:171–176, 1989.

[B] A. Blum, Algorithms for Approximate Graph Coloring, Ph.D. Thesis, *Massachusetts Institute of Technology,* 1991 (MIT/LCS/TR-506, June 1991)

[39] A. Blum, T. Jiang, M. Li, J. Tromp, and M. Yannakakis, Linear approximation of shortest superstrings. *Journal of the ACM,* 41(4):630-647, July 1994.

[40] M. Blum, Program checking. *Proc. FST&TCS,* Springer L.N.C.S. **560**, pp. 1–9.

[41] M. Blum and S. Kannan, Designing Programs that Check Their Work. *Proceedings of the Twenty First Annual Symposium on the Theory of Computing,* ACM, 1989.

[42] M. Blum, M. Luby, and R. Rubinfeld, Self-testing/correcting with applications to numerical problems. *Journal of Computer and System Sciences,* 47(3):549-595, December 1993.

[43] R. Boppana, and M. M. Halldórsson, Approximating maximum independent sets by excluding subgraphs, *Bit* 32, 180-196, 1992.

[44] J. Cai, A. Condon, and R. Lipton. PSPACE is provable by two provers in one round. Journal of Computer and System Sciences, 48(1):183-193, February 1994.

[45] CT M. Cesati and L. Trevisan. On the Efficiency of Polynomial Time Approximation Schemes. Information Processing Letters, 64(4):165-171, 1997.

[46] V. Chvatal, A greedy heuristic for the set covering problem, *Math. Oper. Res.,* 4, 1979, 233-235.

[47] A. Cohen and A. Wigderson. Dispersers, deterministic amplification, and weak random sources. *Proceedings of the Thirtieth Annual Symposium on the Foundations of Computer Science,* IEEE, 1989.

[48] A. Condon, The complexity of the max word problem, or the power of one-way interactive proof systems. *Proceedings of the Eighth Annual Symposium on Theoretical Aspects of Computer Science,* Lecture Notes in Computer Science Vol. 480, Springer Verlag, 1991.

[49] A. Condon, J. Feigenbaum, C. Lund and P. Shor, Probabilistically Checkable Debate Systems and Approximation Algorithms for PSPACE-Hard Functions. *Proceedings of the Twenty Fifth Annual Symposium on the Theory of Computing,* ACM, 1993.

[50] A. Condon, J. Feigenbaum, C. Lund and P. Shor, Random debaters and the hardness of approximating stochastic functions. *SIAM Journal on Computing,* 26(2):369-400, April 1997.

[51] S. Cook, The complexity of theorem-proving procedures. *Proceedings of the Third Annual Symposium on the Theory of Computing,* ACM, 1971.

[52] P. Crescenzi and V. Kann, A compendium of NP optimization problems. Technical Report, Dipartimento di Scienze dell'Informazione, Università di Roma "La Sapienza", SI/RR-95/02, 1995. The list is updated continuously. The latest version is available by anonymous ftp from `nada.kth.se` as `Theory/Viggo-Kann/compendium.ps.Z`. Web address: `http://www.nada.kth.se/~viggo/problemlist/compendium.html`

[53] Pierluigi Crescenzi, Viggo Kann, Riccardo Silvestri, Luca Trevisan, *Structure in Approximation Classes,* Electronic Colloquium on Computational Complexity (ECCC) (066): (1996); SIAM J. on Computing, 28(5):1759-1782, 1999.

[54] P. Crescenzi and L. Trevisan, *MAXNP-completeness Made Easy,* Theoretical Computer Science, 225(1-2):65-79, 1999.

[55] Pierluigi Crescenzi, Luca Trevisan, *On Approximation Scheme Preserving Reducibility and Its Applications.* Theory of Computing Systems 33(1): 1-16 (2000)

[56] E. Dahlhaus, D. Johnson, C. Papadimitriou, P. Seymour, and M. Yannakakis, The complexity of multiway cuts. *SIAM Journal on Computing,* 23:4, pp. 864–894, 1994.

[57] W. De la Vega and G. Lueker, Bin Packing can be solved within $1 + \epsilon$ in Linear Time. *Combinatorica,* vol. 1, pages 349–355, 1981.

[58] I. Dinur, E. Fischer, G. Kindler, R. Raz, and S. Safra. PCP characterizations of NP: Towards a polynomially-small error-probability. In *Proceedings of the 31st ACM Symposium on Theory of Computing,* pages 29–40, 1999.

[59] L. Engebretsen and J. Holmerin, Clique is hard to Approximate within $n^{1-o(1)}$, *Manuscript*

[60] R. Fagin, Generalized first-order spectra and polynomial-time recognizable sets. In Richard Karp (ed.), *Complexity of Computation,* AMS, 1974.

[61] U. Feige, A threshold of ln n for Set Cover. In *Proceedings of the Twenty Eighth Annual Symposium on the Theory of Computing,* ACM, 1996. Journal Version: *Journal of the ACM,* 45(4):634-652, July 1998

[62] U. Feige, S. Goldwasser, L. Lovász, S. Safra, and M. Szegedy, Interactive proofs and the hardness of approximating cliques. *Journal of the ACM,* 43(2):268-292, March 1996.

[63] U. Feige, M. Karpinski, M. Langberg, Improved Approximation of Max-Cut on Graphs of Bounded Degree, `http://www.wisdom.weizmann.ac.il/~mikel/`

[64] U. Feige and J. Kilian. Two prover protocols – Low error at affordable rates. *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing,* ACM, 1994.

[FeKi2] U. Feige and J. Kilian. Zero knowledge and chromatic number. *Proceedings of the Eleventh Annual Conference on Complexity Theory,* IEEE, 1996, pages 172–183.

[65] U. Feige and L. Lovász, Two-prover one-round proof systems: Their power and their problems. *Proceedings of the Twenty Fourth Annual Symposium on the Theory of Computing,* ACM, 1992.

[66] L. Fortnow, J. Rompel, and M. Sipser, On the power of multi-prover interactive protocols. *Theoretical Computer Science,* 134(2):545-557, November 1994.

[67] R. Freivalds, Fast Probabilistic Algorithms. *Proceedings of Symposium on Mathematical Foundations of Computer Science,* Springer-Verlag Lecture Notes in Computer Science, v. 74, pages 57–69, 1979.

[68] K. Friedl, Zs. Hátsági and A. Shen, Low-degree testing. *Proceedings of the Fifth Symposium on Discrete Algorithms,* ACM, 1994.

[69] K. Friedl and M. Sudan, Some improvements to low-degree tests. *Proceedings of the Third Israel Symposium on Theory and Computing Systems,* 1995.

[70] M. Fürer, Improved hardness results for approximating the chromatic number. *Proceedings of the Thirty Sixth Annual Symposium on the Foundations of Computer Science,* IEEE, 1995.

[71] M.R. Garey, R.L. Graham, and J.D. Ullman, *Worst case analysis of memory allocation algorithms,* Proceedings in the 4th Annual ACM Symposium on the Theory of Computing, pp. 143-150, 1972

[72] M. Garey and D. Johnson, The complexity of near-optimal graph coloring. *Journal of the ACM,* 23:43-49, 1976.

[73] M. Garey and D. Johnson, "Strong" NP-completeness results: motivation, examples and implications. *Journal of the ACM,* 25:499-508, 1978.

[74] M. Garey and D. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness.* W. H. Freeman, 1979.

[75] M. Garey, D. Johnson and L. Stockmeyer, Some simplified NP-complete graph problems. *Theoretical Computer Science* 1:237-267, 1976.

[76] P. Gemmell and M. Sudan, Highly resilient correctors for polynomials. *Information Processing Letters,* 43(4):169-174, September 1992.

[77] P. Gemmell, R. Lipton, R. Rubinfeld, M. Sudan, and A. Wigderson, Self-testing/correcting for polynomials and for approximate functions. Proceedings *of the Twenty Third Annual Symposium on the Theory of Computing,* ACM, 1991.

[78] M. Goemans and D. Williamson, Improved approximation algorithms for maximum cut and satisfiability problems using semidefinite programming. *Journal of the ACM,* 42(6):1115-1145, November 1995.

[79] O. Goldreich, A Taxonomy of Proof Systems. In *Complexity Theory Retrospective II,* L.A. Hemaspaandra and A. Selman (eds.), Springer-Verlag, New York, 1997.

[80] S. Goldwasser, S. Micali, and C. Rackoff, The knowledge complexity of interactive proof-systems. *SIAM J. on Computing,* 18(1):186-208, February 1989.

[81] R. Graham, Bounds for certain multiprocessing anomalies, *Bell Systems Technical Journal,* 45:1563-1581, 1966.

[82] R. Graham, Bounds for certain multiprocessing anomalies and related packing algorithms, *Proceedings of the Spring Joint Conference,,* pp: 205-217, 1972.

[83] M. Halldórsson, A still better performance guarantee for approximate graph coloring, *Inform. Process. Lett. 46,* pages 169-172.

[84] J. Håstad, Testing of the long code and hardness for clique. *Proceedings of the Twenty Eighth Annual Symposium on the Theory of Computing,* ACM, 1996.

[85] J. Håstad, Clique is hard to approximate within $n^{1-\epsilon}$. *Proceedings of the Thirty Seventh Annual Symposium on the Foundations of Computer Science,* IEEE, 1996, pages 627–636

[86] J. Håstad, Some optimal inapproximability results. *Proceedings of the Twenty Eighth Annual Symposium on the Theory of Computing,* ACM, 1997, pages 1–10.

[87] J. Håstad, A Rewriting of Feige's Proof for the Setcover, Unpublished manuscript.

[88] O. Ibarra and C. Kim, Fast approximation algorithms for the knapsack and sum of subset problems, *Journal of the ACM,* 22:463-468, 1975.

[89] R. Impagliazzo and D. Zuckerman. How to Recycle Random Bits. *Proceedings of the Thirtieth Annual Symposium on the Foundations of Computer Science,* IEEE, 1989.

[90] D. Johnson, Approximation algorithms for combinatorial problems. *J. Computer and Systems Sci,* 9:256-278, 1974.

[91] V. Kann, Maximum bounded 3-dimensional matching is MAX SNP-complete. *Information Processing Letters,* 37:27-35, 1991.

[92] H. Karloff and U. Zwick A 7/8-approximation for MAX 3SAT?, *Proc. 38th Ann. IEEE Symp. on Foundations of Comput. Sci.,* IEEE Computer Society, 406-415. (1997)

[93] N. Karmakar and R. Karp, An Efficient Approximation Scheme For The One-Dimensional Bin Packing Problem. *Proceedings of the Twenty Third Annual Symposium on the Foundations of Computer Science,* IEEE, 1982.

[94] R. Karp, Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations,* Advances in Computing Research, pages 85–103. Plenum Press, 1972.

[95] D. Karger, R. Motwani, and G. Ramkumar, On approximating the longest path in a graph. *Algorithmica,* 18(1):82-98, May 1997.

[96] M. Karpinski, A. Zelikovsky, Approximating Dense Cases of Covering Problems, *Proc. DIMACS Workshop on Network Design: Connectivity and Facilities Location,* Princeton, 1997, pp. 169-178. Available also as ECCC TR97-004

[97] S. Khanna, N. Linial, and S. Safra, On the hardness of approximating the chromatic number. *Proceedings of the Second Israel Symposium on Theory and Computing Systems,* 1993.

[98] S. Khanna, R. Motwani, M. Sudan, and U. Vazirani, On syntactic versus computational views of approximability. *SIAM Journal on Computing,* 28 (1998) 164–191. Preliminary Version: Proceedings of the 35th Annual IEEE Symposium on Foundations of Computer Science, 1994, pp. 819-830. (with S. Khanna, M. Sudan, and U. Vazirani) Also available as: ECCC Report No. TR95-023, Electronic Colloquim on Computational Complexity, http://www.eccc.uni-trier.de/eccc/, 1995.

[99] S. Khanna, M. Sudan, and L. Trevisan, *Constraint Satisfaction: The Approximability of Minimization Problems,* In Proceedings of the 12th IEEE Conference on Computational Complexity. IEEE, pages 282-296, 1997.

[100] P. Kolaitis and M. N. Thakur, Approximation properties of of *NP* minimization classes. *Proc. Sixth Ann. Structure in Complexity Theory Conf., IEEE Computer Society, 353-366.*

[101] P. Kolaitis and M. Vardi, The decision problem for the probabilities of higher-order properties. *Proceedings of the Nineteenth Annual Symposium on the Theory of Computing,* ACM, 1987.

[102] D. Lapidot and A. Shamir. Fully parallelized multi-prover protocols for NEXP-time. *Journal of Computer and System Sciences,* 54(2):215-220, April 1997.

[103] L. Levin, **Universal'nyĭe perebornyĭe zadachi** (Universal search problems : in Russian). *Problemy Peredachi Informatsii,* 9(3):265–266, 1973. A corrected English translation appears in an appendix to Trakhtenbrot [135].

[104] N. Linial and U. Vazirani, Graph products and chromatic numbers, *Proc. 30th IEEE Symp. on Foundations of Computer Science,* pages 124-128, 1989

[105] R. Lipton, New directions in testing. In J. Feigenbaum and M. Merritt, editors, *Distributed Computing and Cryptography,* volume 2 of *DIMACS Series in Discrete Mathematics and Theoretical Computer Science,* pages 191–202. American Mathematical Society, 1991.

[106] L Lovász On the ratio of optimal integral and fractional covers, *Discrete Mathematics,* 13, pages 383-390, 1975

[107] C. Lund, L. Fortnow, H. Karloff, and N. Nisan, Algebraic Methods for Interactive Proof Systems. *J. ACM,* **39**, 859–868, 1992.

[108] C. Lund and M. Yannakakis, On the hardness of approximating minimization problems. *Journal of the ACM,* 41(5):960-981, September 1994.

[109] C. Lund and M. Yannakakis, The approximation of maximum subgraph problems. *Proceedings of ICALP 93,* Lecture Notes in Computer Science Vol. 700, Springer Verlag, 1993.

[110] D.Micciancio, The Shortest Vector Problem is NP-hard to Approximate within some Constant. *39th IEEE Symposium on Foundations of Computer Science (FOCS'98)*

[111] J. Mitchell. Guillotine subdivisions approximate polygonal subdivisions: Part II- A simple PTAS for geometric k-MST, TSP, and related problems. Preliminary manuscript, April 30, 1996. *To appear in SIAM J. Computing,*

[112] Monien, B., and Speckenmeyer, E. Ramsey numbers and an approximation algorithm for the vertex cover problem, *Acta Inf.* 22, 115-123, (1985).

[113] R. Motwani, Lecture Notes on Approximation Algorithms. Technical Report, Dept. of Computer Science, Stanford University (1992).

[114] P. Orponen and H. Manilla, On approximation preserving reductions: Complete problems and roboust measures, *Technical Report C-1987-28, Department of Computer Science, University of Helsinki. (1987)*

[115] C. Papadimitriou and M. Yannakakis, Optimization, approximation and complexity classes. *Journal of Computer and System Sciences* 43:425-440, 1991.

[116] C. Papadimitriou and M. Yannakakis, The traveling salesman problem with distances one and two. *Mathematics of Operations Research,* 1992.

[117] A. Paz and S. Moran, Non-deterministic polynomial optimization problems and their approximation, *Theoretical Computer Science,* 15:251-277, 1981.

[118] E. Petrank, The Hardness of Approximation: Gap Location. *Computational Complexity,* Vol. 4, pages 133-157, 1994.

[119] S. Phillips and S. Safra, PCP and tighter bounds for approximating MAXSNP. *Manuscript,* Stanford University, 1992.

[120] A. Polishchuk and D. Spielman, Nearly Linear Sized Holographic Proofs. *Proceedings of the Twenty Sixth Annual Symposium on the Theory of Computing,* ACM, 1994.

[121] R. Raz, A parallel repetition theorem. Proceeding of the 27th STOC, 1995, pp. 447-456. Journal version in: *SIAM journal of computing* 27(3) (1998) pp. 763-803

[122] R. Raz and S. Safra. A sub-constant error-probability low-degree test, and a sub-constant error-probability PCP characterization of NP. *Proceedings of the Twenty Eighth Annual Symposium on the Theory of Computing,* ACM, 1997. pages 475-484.

[123] R. Rubinfeld, *A Mathematical Theory of Self-Checking, Self-Testing and Self-Correcting Programs.* Ph.D. thesis, U.C. Berkeley, 1990.

[124] R. Rubinfeld and M. Sudan, Robust characterizations of polynomials with applications to program testing. *SIAM Journal on Computing* 25(2):252-271, April 1996.

[125] S. Sahni, Approximate algorithms for the 0/1 knapsack problem, *Journal of the ACM,* 22:115-124, 1975.

[126] S. Sahni and T. Gonzales, P-complete approximation problems. *Journal of the ACM,* 23:555-565, 1976.

[127] J. Schwartz, Probabilistic algorithms for verification of polynomial identities. *Journal of the ACM,* 27:701-717, 1980.

[128] M. Serna, L. Trevisan, and F. Xhafa. The parallel approximability of non-boolean constraint satisfaction and restricted integer linear programming. In *Proceedings of the 15th Symposium on Theoretical Aspects of Computer Science,* pages 488–498. LNCS 1373, Springer-Verlag, 1998.

[129] A. Shamir, IP = PSPACE. *Journal of the ACM,* 39(4):869-877, October 1992.

[130] Alex Samorodnitsky and Luca Trevisan, A PCP Characterization of NP with Optimal Amortized Query Complexity, *Manuscript*

[131] M. Sudan, *Efficient Checking of Polynomials and Proofs and the Hardness of Approximation Problems.* Ph.D. Thesis, U.C. Berkeley, 1992. Also appears as ACM Distinguished Theses, Lecture Notes in Computer Science, no. 1001, Springer, 1996.

[132] M. Sudan and L. Trevisan. Probabilistically checkable proofs with low amortized query complexity. In *Proceedings of the 39th IEEE Symposium on Foundations of Computer Science,* 1998.

[133] M. Szegedy, Many Valued Logics and Holographic Proofs, ICALP 99

[134] G. Tardos, Multi-prover encoding schemes and three prover proof systems. *Proceedings of the Ninth Annual Conference on Structure in Complexity Theory,* IEEE, 1994.

[135] B. Trakhtenbrot, A survey of Russian approaches to *Perebor* (brute-force search) algorithms. *Annals of the History of Computing* 6:384-400, 1984.

[136] Luca Trevisan, *Reductions and (Non-)Approximability,* PhD Thesis, Department of Computer Science, University of Rome 'La Sapienza', 1997," http://www.cs.columbia.edu/ luca/thesis

[137] L. Trevisan. Approximating satisfiable satisfiability problems. In *Proceedings of the 5th European Symposium on Algorithms,* pages 472–485. LNCS 1284, Springer-Verlag, 1997.

[138] L. Trevisan. Recycling queries in PCPs and in linearity tests. In *Proceedings of the 30th ACM Symposium on Theory of Computing,* 1998.

[139] L. Trevisan, G.B. Sorkin, M. Sudan, D.P. Williamson. Gadgets, Approximation, and Linear Programming. In Proceedings of the 37th Symposium on Foundations of Computer Science. IEEE, pp. 617-626, 1996. Full version submitted to SIAM J. on Computing.

[140] L. Welch and E. Berlekamp, Error correction of algebraic block codes. *US Patent* Number 4,633,470 (filed: 1986).

[141] M. Yannakakis. On the approximation of maximum satisfiability. *Journal of Algorithms,* 17(3):475-502, November 1994.

[142] D. Zuckerman. On unapproximable versions of NP-complete problems. *SIAM Journal on Computing,* 25(6):1293-1304, December 1996

[143] U. Zwick. Finding almost satisfying assignment. In *Proceedings of the 30th ACM Symposium on Theory of Computing,* 1998.

HANDBOOK OF COMBINATORIAL OPTIMIZATION (VOL 5)
D.-Z. Du and P.M. Pardalos (Eds.) pp. 193-216

Steiner Trees in Industry

Xiuzhen Cheng
Department of Computer Science
George Washington University, DC
E-mail: cheng@gwu.edu

Yingshu Li and Ding-Zhu Du
Department of Computer Science and Engineering
University of Minnesota, Minneapolis, MN 55455
E-mail: {cheng,dzd}@cs.umn.edu

Hung Q. Ngo
Department of Computer Science and Engineering
State University of New York, Buffalo, NY 14260
E-mail: hungngo@cse.buffalo.edu

Contents

References

1 Introduction

The Steiner tree problem is a classic intractable problem with many applications in the design of computer circuits, long-distance telephone lines, or mail rout ing, etc. The computational nature of the problem makes it a traditional research subject in theory of computing.

Given a set of points in the Euclidean plane, the shortest network interconnecting the points in the set is called the *Steiner minimum tree.* The Steiner minimum tree may contain some vertices which are not the given points. Those vertices are called *Steiner points* while the given points are called *terminals.* The shortest network for three terminals was first studied by Fermat (1601-1665). Fermat proposed the problem of finding a point to minimize the total distance from it to three terminals in the Euclidean plane. The direct generalization is to find a point to minimize the total distance from it to n terminals, which is still called the Fermat problem. The Steiner minimum tree problem is an indirect generalization. Recent research on mathematical history showed that this generalization (i.e., the Steiner minimum tree) was first studied by Gauss.

On March 19, 1836, Schumacher wrote a letter to Gauss and mentioned a paradox about Fermat problem on terminals: For four vertices of a convex quadrilateral, the solution of the Fermat problem is the intersection point of two diagonals. When two of the four vertices move to the same position, the intersection point of two diagonal would also move to this position which is not the solution of the Fermat problem for the three points resulting from the four vertices. Schumacher could not understand why this would happen. On March 21, 1836, Gauss wrote back to Schumacher and explained the paradox. In this letter, he mentioned another generation of Fermat problem, that is, aim on the network structure instead of a point position. Gauss already discussed all possible topologies of Steiner minimum trees for four points. (See Schreiber [1986].)

In the last centenary, the Steiner tree problem has been extended to various metric spaces. Among them, the Euclidean Steiner tree (i.e., the Steiner tree in Euclidean plane), the rectilinear Steiner tree (i.e., the Steiner tree in the rectilinear plane), and the network Steiner tree (i.e., the Steiner tree in an undirected network) were recognized as most important ones and

received much more attentions. We call them *classic* Steiner tree problems.

It is well-known that Steiner tree problems usually are NP-hard (see Karp [1972], Garey and Johnso [1978], Garey, Graham, and Johnson [1978], Foulds and Graham [1982]), i.e., the exact optimal solutions are unlikely computable in polynomial-time. Therefore, one has to put efforts on looking for good approximation solutions. During the last ten years, great progress has been made in the study of Steiner trees, including solution of the Gilbert-Pollak conjecture on the Steiner ratio, solution of the better approximation problem, and designs of polynomial-time approximation schemes for Euclidean and rectilinear Steiner Trees.

The Gilbert-Pollak conjecture was made in 1968. There is a folklore about it (see Hwang [1990]). A large company usually has a private telephone network interconnecting its branchs. For example, University of California has nine campuses. If you call from one campus to another one through private network, it would be counted as a long distance call. The private network is not really built by the company privately. It is rent from telephone company, realized by a special phone number. For example, when I graduated from University of California at Santa Barbara and went to work in Berkeley, my advisor gave me a special phone number for calling him from Berkeley. This number is the private network of University of California. Before 1967, the charge of a private network was determined by the length of the minimum spanning tree for the destinations. The minimum spanning tree for a set of terminals is the shortest tree with edges between terminals. It is different from the Steiner tree by disallowing the existence of Steiner points. This restriction causes the minimum spanning tree possibly longer than the Steiner minimum tree for the same set of terminals. In 1967, a flight company found this fact. Therefore, they requested some new services at those Steiner points, so that the minimum spanning tree for the new set of destinations is the Steiner minimum tree for the original set of destinations, which is shorter than the minimum spanning tree for the original set of destinations. Therefore, those requests increased the service of telephone company and decreased the charge from the telephone company. With this situation, the telephone company had to change the billing base from the minimum spanning tree to the Steiner minimum tree. Therefore, the telephone company faced a problem: With this change, how much should be increased on the rate of the unit length? This motivated the study of the Steiner ratio, the ratio of lengths of the Steiner minimum tree and the minimum spanning tree for the same set of terminals. Gilbert and Pollak [1968] conjectured that the Steine ratio in the Euclidean plane is

at least $\sqrt{3}/2$ which is achieved by three vertices of an equilateral triangle.

The work of Gilbert and Pollak [1968] is a turning point in the study of Steiner trees. Before this work, the Steiner tree was studied mainly due to mathematical interests and hence progress was made very slowly. Gilbert-Pollak's work brought the Steiner tree into model industries. Since then, the Steiner tree attracts more and more attentions and the number of research publications in the Steiner tree grows very fast. Through many efforts made by Graham and Hwang [1976], Pollak [1978], Chung and Hwang [1978], Du and Hwang [1983], Du, Hwang and Yao [1985], Chung and Graham [1985], Friedel and Widmayer [1989], Booth [1991], and Rubinstein and Thomas [1991], the Gilbert-Pollak conjecture was finally proved by Du and Hwang [1990,1992]. The significance of their proof stems also from the potential applications of the new approach included in the proof and hence received a lot of public recognitions (see Stewart [1991], Kolata [1990], Campbell [1992], Cipra [1990, 1991], Peterson [1990], and Sangalli [1991]).

While the Steiner minimum tree is an NP-hard problem in many metric spaces, the minimum spanning tree can be computed in at most $O(n^2)$ time. Therefore, the inverse of the Steiner ratio is actually the performance ratio of the minimum spanning tree when it is considered as a polynomial-time approximation of the Steiner minimum tree. Is there polynomial-time approximation better than the minimum spanning tree? For more than twenty year (1968-1990), many polynomial-time approximations were discovered (see Chang [1972], Korthonen [1979], Kou and Makki [1987], Smith and Liebman [1979], Smith, Lee, and Liebman [1981], Waxman and M. Imase [1988], Smith and Shor [1992]), however none of them has a performance ratio which can be proved to be better than the inverse of the Steiner ratio. This situation exists not only in the Euclidean plane, but also in any interested metric space. Therefore, a long-standing open problem was generated. In general, it was called the *better approximation problem* whether there exists a polynomial-time approximation for Steiner minimum trees in each metric space with performance ratio smaller than the inverse of the Steiner ratio. Zelikovsky [1993] made the first breakthrough. He found a polynomial-time 11/6-approximation for network Steiner minimum trees which beats the inverse of the Steiner ratio in networks, $\rho_2^{-1} = 2$. Soon later, Berman and Ramaiye [1994] gave a polynomial-time 92/72-approximation for the Steiner minimum tree in the rectilinear plane which beats the inverse of the rectilinear Steiner ratio 3/2 (see Hwang [1976]), and Du, Zhang, and Feng [1991] showed a general solution for the open problem. They showed that in any metric space, there exists a polynomial-time approxi-

mation with performance ratio better than the inverse of the Steiner ratio provided that for any set of a fixed number of points, the Steiner minimum tree is polynomial-time computable.

After the better approximation problem is settled, it is naive to ask how small performance ratio a polynomial-time approximation can achieve. Bern and Plassmann [1989] showed that the network Steiner minimum tree problem is MAX-SNP hard. Namely, it is unlikely to have a polynomial-time approximation scheme. For Euclidean and rectilinear Steiner minimum trees, Arora [1996] and Mitchell [1996] independently discovered a surprising result that there exist polynomial-time approximation schemes. Their approaches work not only in Steiner trees but also in a family of geometric optimization problems. This fact shows clearly that the Steiner tree is not an isolated research topic. It always influences and reflects the progress in the general theory of computing, especially in algorithm design and analysis.

"What could be the next major development?" Jeff Ullman asked when one of authors visited at Stanford University ten years ago. Now, many researchers think about the same problem in the area of Steiner trees. After powerful techniques have been discovered for studying Steiner trees, classic problems on approximation of Steiner trees in the Euclidean and rectilinear plane have been settled. Should we put our research interests only in network Steiner trees? If we use yahoo.com to search subject Steiner-tree, then we may found 1720 web-pages on this subject. Many of them come from industries. This suggests a wide field in Steiner trees. In fact, most major theoretical open problems in Steiner trees were initiated from industry applications. Now, it is time to look back, to find out the impact of previous theoretical development in industries, and to obtain source from industries to suppose new developments in theory. Therefore, we review some variations of Steiner tree problems and related research problems.

2 Important Techniques Discovered Previously

All three major developments described as above result from discovery of new techniques in analysis and designs of approximation algorithms.

The Gilbert-Pollak conjecture was proved with a so-called minimax approach. Indeed, when an approximation is obtained from adding restriction to the original optimization problem, the determination of the performance ratio can be transformed to a minimax problem. In this approach, the central part is a new minimax theorem about minimizing the maximum value

of several concave functions over a polytope as follows:

Minimax Theorem. Let $f(x) = \max_{i \in I} g_i(x)$ where I is a finite set and $g_i(x)$ is a continuous, concave function in a polytope X. Then the minimum value of $f(x)$ over the polytope X is achieved at some critical point, namely, a point satisfying the following property:

(*) There exists an extreme subset Y of X such that $x \in Y$ and the index set $M(x)$ $(= \{i \mid f(x) = g_i(x)\})$ is maximal over Y.

The Steiner ratio problem is first transformed to such a minimax problem ($g_i(x)$= (the length of a Steiner tree) - (the Steiner ratio)·(the length of a spanning tree with graph structure i) where x is a vector whose components are edge-lengths of the Steiner tree) and the minimax theorem reduces the minimax problem to the problem of finding the minimax value of the concave functions at critical points. Then each critical point is transformed back to an input set of points with special geometric structure; it is a subset of a lattice formed by equilateral triangles. This special structure is called *critical structure* which enables us to verify the conjecture corresponding to the non-negativeness of minimax value of the concave functions.

Using the same approach, Gao, Du, and Graham [1995] proved that in any Minkowski plane (or 2-dimensional Banach space), the Steiner ratio is at least 2/3. This settles a conjecture made independently by Cieslik [1990] and Du *et al.* [1993]. The main contribution of Gao, Du, and Graham is in the study of Steiner trees for points in equilateral triangle lattice. In fact, Du and Hwang [1992] already showed that the Steiner ration in any Minkowski plane is achieved by points in an equilateral triangle lattice. Recently, Brazil et al [1996] showed a very interesting result on Steiner trees for square-lattice points. As an intermediate result, they found that every full Steiner tree for square-lattice points has linear topology, i.e. all Steiner points lay on a path in the tree. If a similar result can be proved for points on every equilateral triangle lattice (we intend to do it), then it is possible to give a new proof for the above conjecture and to solve other open problems, including one of conjectures in Minkowski planes that the Steiner ratio in any Minkowski plane equals the Steiner ratio in its dual plane. Wan, Du, and Graham [1997] showed that this conjecture is true for five points.

The excellent idea of Zelikovsky [1993] for establishing better approximations consists of two parts. The first part is design of approximation with greedy algorithm. This greedy algorithm is not directly applied to the Steiner minimum tree. It is applied to the **k-size** Steiner minimum tree, a restriction of the Steiner minimum tree. Let us explain it as follows.

A tree interconnecting a terminal set is called a *Steiner tree* if every leaf

is a terminal. However, a terminal in a Steiner tree may not be a leaf. A Steiner tree is *full* if every terminal is a leaf. When a terminal is not a leaf, the tree can be decomposed into several small trees at this terminal. In this way, every Steiner tree can be decomposed into smaller trees in each of which every terminal is a leaf. These smaller trees are called *full components* of the tree. The *size* of a full component is the number of terminals in the full component. A k-size Steiner tree is a Steiner tree with all full components of size at most k. The k-size Steiner minimum tree is the shortest one among all k-size Steiner trees. The 2-size Steiner minimum tree is the minimum spanning tree. The k-size Steiner minimum tree for $k \geq 3$ is certainly an approximation better than the minimum spanning tree. However, the k-size Steiner minimum tree for $k \geq 4$ is NP-hard and no polynomial-time algorithm has been found for the 3-size Steiner minimum tree. Therefore, a greedy approximation is employed instead of themselves.

The second part of Zelikovsky's excellent idea is to connect the performance ratio of the greedy approximation to the k-Steiner ratio. The k-Steiner ratio in a metric space is the least ratio of lengths between the Steiner minimum tree and the k-Steiner minimum tree for the same set of terminals in the metric space. The 2-Steiner ratio is exactly the Steiner ratio. A better lower bound for the k-Steiner ratio will give a better performance ratio for approximations of Zelikovsky's type. Zelikovsky [1993] showed that the 3-Steiner ratio in graphs is at least 3/5. Du, Zhang, and Feng [1991] showed that the k-Steiner ratio in graphs is at least $\lfloor \log_2 k \rfloor / (1 + \lfloor \log_2 k \rfloor)$. Recently, Borchers and Du [1995] completely determined the exact value of the k-Steiner ratio in graphs for all k and Borchers, Du, Gao, and Wan [1998] determined the exact value of the k-Steiner ratio in the rectilinear plane for all k. The techniques discovered in these works may improve the currently best-known approximation performance ratio for network Steiner tree problem provided by Karpiski and Zelikovsky [1997] (we intend to do it).

In 1995, S. Arora and J. Mitchell independently discovered powerful techniques to establish polynomial-time approximation schemes for geometric optimization problems, including Euclidean and rectilinear Steiner tree problems. It is quite interesting to notice that Arora [1996] appeared only a few days before Mitchell [1999]. Any way, they use very different techniques to reach the same goal. Therefore, both are very interesting.

Arora discovered a new technique about dynamic partition. The dynamic partition was first introduced to the area of Steiner trees by Jiang and Wang [1994]. They designed a polynomial-time approximation scheme

for the Euclidean and rectilinear Steiner minimum tree under restriction that the ratio of lengths between the longest edge and the shortest edge in a minimum spanning tree is bounded by a constant. Arora's technique is based on recursive partition. In Jiang and Wang [1994], although partition can be moved parallelly, the size of each cell is fixed. It cannot be varied according to local information about distribution of terminals. Therefore, only in case that terminals are distributed almost evenly, the partition could work well. This is why such a condition that the ratio of lengths between the longest edge and the shortest edge in a minimum spanning tree is bounded by a constant is required. However, in Arora's recursive partition, each big cell is partitioned into small cells independently from other big cells. How to cut only depends on the situation inside of itself. This advantage enables him to discard the condition in Jiang and Wang.

Mitchell's technique was initiated from studying a minimum length rectangular partition problem. Given a rectilinear region R surrounded by a rectilinear polygon and some rectilinear holes, a *rectangular partition* of R is a set of segments in R, which divide R into small rectangles each of which does not contain any hole in its interior. The problem is to find such a rectangular partition with the minimum total length. This problem is NP-hard.

Du et al. [1986] introduced a concept of guillotine subdivision. A guillotine subdivision is a sequence of cuts performed recursively such that each cut partitions a piece ito at least two. Du et al. [1986] showed that the minimum length guillotine rectangular partition can be computed in polynomial-time. However, they were only able to show that this guillotine subdivision is a 2-approximation of the minimum length rectangular partition in a special case. Mitchell [1996] showed that this is actually true in general. He also successfully utilized this technique to obtain constant approximations for other geometric optimization problems.

Inspired by this success, Mitchell [1999] extended guillotine subdivision to m-guillotine subdivision, a rectangular polygonal subdivision such that there exists a cut whose intersection with the subdivision edges consists of a small number $(O(m))$ of connected components and the subdivisions on either side of the cut are also m-guillotine. With a minor change of the proof of Mitchell [1996], Mitchell established a polynomial-time approximation scheme for minimum length rectangular partition. Mitchell [1997] and Mitchell et al [1999] further extended this m-guillotine subdivision technique to other geometric optimization problems, including Euclidean and rectilinear Steiner tree problems.

3 Some Problems Related to Industries

Successful researches on classical Steiner tree problems encourage extensive study on variations of Steiner trees with various application in industries. Currently, they form a quite active research direction in Steiner trees.

3.1 Class Steiner Trees

In physical VLSI designs, "after the placement of components on a chip, set of pins on the component boundaries are to be connected within the remaining free chip space. For each set of pins sharing the same electrical signal (a net), a Steiner minimal tree is sought, with the pins as required vertices, and the vertices of a grid-like graph, defined by the positions of pins and component boundaries, as Steiner vertices." (See Ihler et al [1999].) Motivated from the flexibility of pin positions, Ihler et al proposed a variation of Steiner tree problem, called *class Steiner tree* problem as follows: given a connected graph with required classes of terminals, find a shortest connected subgraph that contains at least one terminal from each class.

Ihler et al showed that the class Steiner tree problem is harder than set-covering problem and hence unlikely to have a constant-bounded approximation. Therefore, they were satisfied with two approximations with pretty large performance ratio.

However, we feel that there are some important informations lost from the real world problem to the mathematical formulation:

- The Steiner tree lays in the rectilinear plane rather than an arbitrary graph.
- Each class of terminals lay on the boundary of a connected component.

The lack of the above information made the computational complexity increasing. We believe that if we consider the class Steiner tree problem with these two conditions, then there may exist much better approximation, even polynomial-time approximation schemes. We intend to study along this direction.

A similar problem appeared in designs of highway intersections: It is to construct roads of minimum total length to interconnect several highways under the constraint that the roads can intersect each highway only at one point in a designated interval which is a line-segment. Du, Hwang, and Xue [1999] presented a set of optimality conditions for the problem and

showed how to construct a solution to meet this set of optimality conditions. Howeever, no bounded polynomial-time approximation has been found for this problem. We intend to study its approximation together with the above problem. We intend to study its approximation together with the above problem.

3.2 On-line and Dynamic Steiner Trees

When a new customer is out of original telephone network, the company has to build a new line to connect the customer into the network. This situation brings us an on-line Steiner tree problem as follow: Assume that a sequence of points in a metric space are given step by step. In the ith step, only locations of the first n_i points in the sequence are known. The problem is to construct a shorter network at each step based on the network constructed in previous steps. The study of on-line problems was initiated from Sleator and Tarjan [1985] and Manase, McGeoch, and Sleator [1988]. A criterion for the performance of an on-line algorithm is to compare the solution generated by the on-line algorithm with the solution of corresponding off-line problem. In the Euclidean plane, it has been known that the worst-case ratio of lengths between on-line solution and off-line solution is between $O(n \log n / \log \log n)$ and $O(n \log n)$ (see Alon and Azar [1993], Westbrook and Yan [1995], and Tsai et al [1996]).

When customers are allowed to drop from the network, it called the *dynamic Steiner tree* problem. The dynamic Steiner tree has application in network routing (see Aharoni and Cohen [1998]). For online and dynamic optimization problems, the running time is very important issue in designs of approximation algorithms. Therefore, reducing running time with preserved performance ratio is an interesting issue in the study of those problems.

3.3 Steiner Tree Packing

Given m sets of terminals, find m Steiner trees each interconnecting one set of terminals such that no cross lines exists and the total length reaches the minimum. This is called the *Steiner tree packing* problem. This problem came from VLSI design at the earlier year (see Hwang et al [1992]). Recently, the development of new technologies requires to solve some variations of Steiner tree packing problems (see Pulleyblank [1995]). For example, The edges of the Steiner trees are required to lie in channels between cells. Each channel has a capacity which tells at most how many edges can run through

it. This problem is widely open.

3.4 High Dimensional Steiner Trees

Recently, the satellite communication promotes studying on Steiner trees in three-dimensional Euclidean space and the multilayer chips initiates interests in Steiner trees in three-dimensional rectilinear space. Actually, Steiner trees in high-dimensional space have been studied for many year due to some theoretical interest and its application in constructing phylogenetic trees (Cavalli-Sforza and Edwards [1967]). There are two long-standing conjectures about them.

Conjecture 3.1 (Chung-Gilbert [1976) *] The Steiner ratio in n-dimensional Euclidean space is at least $\sqrt{3}/(4-\sqrt{2})$.*

Conjecture 3.2 (Graham-Hwang [1976) *] The Steiner ratio in n-dimensional rectilinear space is $d/(2d-1)$.*

Proving these conjectures is quite challenge and would certainly need to make a great progress in analysis techniques.

It was also conjectured by Gilbert and Pollak [1968] that in any Euclidean space the Steiner ratio is achieved by the vertex set of a regular simplex. Chung and Gilbert [1976] constructed a sequence of Steiner trees on regular simplexes. The lengths of constructed Steiner trees goes decreasingly to $\sqrt{3}/(4-\sqrt{2})$. Although the constructed trees are not known to be Steiner minimum trees, Chung and Gilbert conjectured that $\sqrt{3}/(4-\sqrt{2})$ is the best lower bound for Steiner ratios in Euclidean spaces. Clearly, if $\sqrt{3}/(4-\sqrt{2})$ is the limiting Steiner ratio in d-dimensional Euclidean space as d goes to infinity, then Chung-Gilbert's conjecture is a corollary of Gilbert and Pollak's general conjecture. However, this general conjecture of Gilbert and Pollak has been disproved by Smith [1992] for dimension from three to nine and by Du and Smith [1996] for dimension larger than two. Now, interesting questions which arise in this situation are about Chung and Gilbert's conjecture. Could Chung-Gilbert's conjecture also be false? If the conjecture is not false, can we prove it by the minimax approach?

First, we claim that Chung-Gilbert's conjecture could be true. In fact, we could get rid of Gilbert-Pollak's general conjecture, and use another way to reach the conclusion that the limiting Steiner ratio for regular simplex is the best lower bound for Steiner ratios in Euclidean spaces. To support our viewpoint, let us analyze a possible proof of such a conclusion as follows.

Consider n points in $(n-1)$-dimensional Euclidean space. Then all of $n(n-1)/2$ distances between the n points are independent. Suppose that we could do a similar transformation and the minimax theorem could apply to these n points to obtain a similar result in the proof of Gilbert-Pollak's conjecture for Euclidean plane, i.e. a point set with critical geometric structure has the property that the union of all minimum spanning trees contains as many equilateral triangles as possible. Then such a critical structure must be a regular simplex.

The above observation tells us two facts:

(a) Chung-Gilbert's conjecture can follow from the following two conjectures.

- The Steiner ratio for n points in an Euclidean space is not smaller than the Steiner ratio for the vertex set of $(n-1)$-dimensional regular simplex.
- (Smith[1992]) $\sqrt{3}/(4-\sqrt{2})$ is the limiting Steiner ratio for simplex.

(b) It may be possible to prove Conjecture 1 by the minimax approach if we could find a right transformation.

One may wonder why we need to find a right transformation. What happens to the transformation used in proof of Gilbert-Pollak's conjecture in the Euclidean plane? Here, we remark that such a transformation does not work for Conjecture 1. In fact, in the Euclidean plane, with a fixed graph structure, all edge-lengths of a full Steiner tree can determine the set of original points and furthermore the length of a spanning tree for a fixed graph structure is a convex function of the edges-lengths of the Steiner tree. However, in Euclidean spaces of dimension more than two , edge-lengths of a full Steiner tree are not enough to determine the set of original points. Moreover, adding other parameters may destroy the convexity of the length of a spanning tree as a function of the parameters.

From the above, we see that proving Chung-Gilbert's conjecture requires a further development of the minimax approach.

Graham-Hwang's conjecture can be easily transferred to a minimax problem requested by our minimax approach. For example, choose lengths of all straight segments of a Steiner tree. When connection pattern of the Steiner tree is fixed, the set of original points can be determined by such segments-lengths, the length of the Steiner tree is a linear function and the length of a spanning tree is a convex function of such segment-lengths, so that g_i is a concave function of such segment-lengths. However, for this transformation,

it is hard to determine the critical structure. To explain the difficulty, we notice that in general the critical points could exist in both the boundary and interior of the polytope. (See the minimax theorem.) In the proof of Gilbert-Pollak's conjecture in plane, a crucial fact is that only interior critical points need to be considered in a contradiction argument. The critical structure of interior critical points are relatively easy to be determined. However, for the current transformation on Graham-Hwang's conjecture, we have to consider some critical points on the boundary. It requires a new technique, either determine critical structure for such critical points or eliminate them from our consideration.

One possible idea is to combine the minimax approach and Hwang's method. In fact, by the minimax approach, we may get useful condition on the set of original points. With such a condition, the point set can have only certain type of full Steiner trees. This may reduce the difficulty of extending Hwang's method to high dimension.

The techniques developed for solving problems about Steiner trees in the Euclidean and rectilinear space can usually be extended to Minkowski-Banach spaces. The following open problems are also our target:

Conjecture 3.3 (Cieslik [1990) , *Du et al [1993]] In any Minkowski plane, the Steiner ratio is between 2/3 and* $\sqrt{3}/2$.

Conjecture 3.4 (Du et al [1993) *] The Steiner ratio in a Minkowski plane equals that in its dual plane.*

Conjecture 3.5 *In any infinite dimensional Banach space, the Steiner ratio is between* 1/2 *and* $\sqrt{3}/(2-\sqrt{2})$.

Conjecture 3.6 *The Steiner ratio in any Banach space equals that in its dual space.*

3.5 Multi-weight Steiner Trees

A complicated computer network may consist of nets of different speeds. The following problem was proposed based on such a background: Consider an undirected network with multiple edge weights $(c_1(e), c_2(e), ..., c_k(e))$ $(c_1(e) > c_2(e) > \cdots > c_k(e))$. Given a subset N of vertices and a partition $\{N_1, N_2, ..., N_k\}$ of N with $|N_1| \geq 2$, find a subnetwork interconnecting N with minimum total weight such that the length of any edge e on a path between a pair of vertices in N_j is at least $c_j(e)$ (see Iwainsky [1985] and

Duin [1991]). We found that this problem can be transformed to multiphase Steiner network problem. Thus, we intend to combine researches on these two problems.

What is the multiphase Steiner network problem? Given an edge-weighed graph B with vertex set X and subsets $X_1, Y_1, \cdots, X_m, Y_m$ of X with $X_i \cap Y_i = \emptyset$, the problem is to find a minimum weighed subgraph G such that for every $i = 1, \cdots, m$, G contains a Steiner tree for X_i without using vertices not in Y_i.

A accompany of the multiphase Steiner tree problem is the multiphase spanning network problem: Given an edge-weighed complete graph with vertex set X ($|X| = n$) and subsets $X_1, \cdots, X_m$ of vertices, the problem is to find a minimum weighed subgraph G such that for every $i = 1, \cdots, m$, G contains a spanning tree for X_i. If NP $\neq$ P, then the best performance ratio of polynomial-time approximation for this problem is $O(n \log n)$. But, in unit weigh case (this case has more applications), it is unknown whether a constant-bounded polynomial-time approximation exists or not.

Both multiphase spanning network and Steiner network problems arose in communication network design (Prisner [1992]) and vacuum system design (Du and Miller [1988]). For the former one, when the solution is a forest, the system $(X_1, \cdots, X_m)$ is called *subtree hypergraph.* Such a system has various applications in computer database schemes (Beeri et al [1983]) and statistics. It is also related to chordal graphs (Duchet [1978]). Tarjan and Yannakakis [1984] gave a $O(m + n)$-time algorithm to tell whether a set system is a subtree hypergraph or not.

3.6 Steiner Arborescence

Given a weighted directed graph G, a vertex r, and a subset P of n vertices, a *Steiner arborescence* is a directed tree with root r such that for each $x \in P$ there exists a path from r to x. The shortest Steiner arborescence is also called a *minimum Steiner arborescence.* Computing minimum Steiner arborescence is an NP-hard problem. Also, one knows that if NP $\neq$ P, then the best possible performance ratio of polynomial-time approximation for this problem is $O(\log n)$. This means that although, like the minimum spanning tree, the minimum arborescence as a shortest arborescence tree without Steiner points can be computed in polynomial-time, the Steiner ratio (the maximum lower bound for the ratio of lengths between the minimum Steiner arborescence and the minimum arborescence for the same set of given points) in directed graphs is zero. Charikar et al [1999] apply

Arora's techniques to this problem and obtained the best known result that for any $\varepsilon > 0$ there exists a polynomial-time approximation with performance ratio $O(n^{\varepsilon})$. An open problem remains for closing the gap between the lower bound and the upper bound for the performance ratio.

A version of this problem in the rectilinear plane has a great interest in VLSI designs and an interesting story in the literature. Given a set P of n points in the first quadrant of the rectilinear plane, a *rectilinear Steiner arborescence tree* is a directed tree rooted at the origin, consisting of all paths from the root to points in P with horizontal edges oriented in left-to-right direction and vertical edges oriented in bottom-up direction. What is the complexity of computing the minimum rectilinear arborescence? First, it was claimed that a polynomial-time algorithm was found. However, Rao, Sadayappan, Hwang, and Shor [1992] found a serious flow in this algorithm. Although they could not show the NP-completeness of the problem, they pointed out the difficulties of computing the minimum rectilinear arborescence in polynomial-time. They also showed that while the ratio of lengths between a minimum arborescence tree and a minimum Steiner tree for the same set of points tends to infinity, there is a polynomial-time approximation with performance two. Recently, Shi and Su [2000] showed that computing the minimum rectilinear arborescence is NP-hard. Lu and Ruan [2000] showed, by employing Arora's techniques, that there is a polynomial-time approximation scheme for the problem. We intend to implement this algorithm to obtain an efficient software in the real world.

3.7 Bottleneck Steiner Trees and Related Problem

In wavelength-division multiplexing (WDM) optical network design (Li et al [1994] and Ramamurthy et al [1997]), suppose we need to connect n sites located at $p_1, p_2, \cdots, p_n$ with WDM optical network. Due to the limit in transmission power, signals can only travel a limited distance (say R) for guaranteed correct transmission. If some of the inter-site distances are greater than R, we need to provide some amplifiers or receivers/transmitters at some locations in order to break it into shorter pieces. This situation requires us to consider the problem of minimizing the maximum edge-length and the number of Steiner points in design of WDM optical network. To do so, two variations of Steiner trees have been studied.

The first is to minimize the maximum edge-length under an upper bound on the number of Steiner points. That is, given a set $P = \{p_1, p_2, \ldots, p_n\}$ of n terminals and an positive integer k, we want to find a Steiner tree with at

most k Steiner points such that the length of the longest edges in the tree is minimized. This is one of the bottleneck Steiner tree problems. Wang and Du [2000] showed that (a) if $NP \neq P$, then the performance ratio of any polynomial-time approximation for the problem in the Euclidean plane is at least $\sqrt{2}$; (b) if $NP \neq P$, then the performance ratio of any polynomial-time approximation for the problem in the rectilinear plane is at least two; (c) there exists a polynomial-time approximation with performance ratio two for the problem in both rectilinear and Euclidean planes.

The second is to minimize the number of Steiner points under upper bound for edge-length. That is, given a set of n terminals $X = \{p_1, p_2, \cdots, p_n\}$ in the Euclidean plane $\mathcal{R}^2$, and a positive constant R, the problem is to compute a tree T spanning a superset of X such that each edge in the tree has a length no more than R and with the minimum number $C(T)$ of points other than those in X, called *Steiner points.* This problem is called *Steiner tree problem with minimum number of Steiner points,* denoted by *STP-MSP* for short. Lin and Xue [1998] showed that the STP-MSP problem is NP-hard. They also showed that the approximation obtained from the minimum spanning tree by simply breaking each edge into small pieces within the upper bound (called steinerized spanning tree) has a worst-case performance ratio at most five. Chen et al [2000] showed that this approximation has a performance ratio exactly four. They also presented a new polynomial-time approximation with a performance ratio at most three and a polynomial-time approximation scheme under certain conditions. Lu et al [2000] studied the STP-MSP in rectilinear plane. They showed that in the rectilinear plane, the steinerized spanning tree has performance ratio exactly three and there exists a polynomial-time approximation two.

4 Conclusion

The following problems are worth studying:

- Open problems on the Steiner ratio, such as Chung-Gilbert's conjecture, Graham-Hwang's conjecture, and Cielick's conjecture, etc..
- Find better approximation for network Steiner trees and establish an explicit lower bound for the approximation performance ratio of network Steiner trees.
- Close the gap between the lower bound and the upper bound for the approximation performance ratio of Steiner minimum arborescence.

- Find more efficient approximation algorithms for on-line and dynamic Steiner minimum trees and various Steiner tree packing problems.
- Find good approximation algorithms for multi-weighted Steiner trees and multiphase Steiner trees and study close relationship between multi-weighted Steiner trees, multiphase Steiner trees, and phylogenetic trees.
- Make clear whether there exists a polynomial-time approximation scheme for class Steiner tree in the special case with the real world background and highway interconnection problem.
- Close the gap between the lower bound and the upper bound for the approximation performance ratio of bottleneck Steiner tree in the Euclidean plane and make clear whether there exists a polynomial-time approximation scheme for the Steiner tree with minimum number of Steiner points and bounded edge-length.
- Implement efficient approximation algorithms to meet the requests from industries.

We believe that to attack these new and old open problems new techniques are still required and the Steiner tree is still an attractive topic for researchers in combinatorial optimization and computer science.

References

E. Aharoni and R. Cohen [1998], Restricted dynamic Steiner trees for scalable multicast in datagram networks, *IEEE/ACM Transactions on Networking,* 6, no.3, 286-297.

N. Alon and Y. Azar [1993], On-line Steiner trees in the Euclidean plane, *Discrete Comput. Geom.* 10, no. 2, 113–121.

S. Arora [1996], Polynomial time approximation schemes for Euclidean TSP and other geometric problems, *Proceedings of 37th FOCS,* 1996, pp. 2-12.

C. Beeri, R. Fagin, D. Maier, M. Yannakakis [1983], On the desirability of acyclic database schemes, *J. ACM* 30*,* 479-513.

P. Berman and V. Ramaiyer [1994], Improved approximations for the Steiner tree problem, *J. of Algorithm* 17, 381-408.

M. Bern and P. Plassmann [1989], The Steiner problem with edge lengths 1 and 2, *Information Processing Letters* 32, 171-176.

R.S. Booth [1991], The Steiner ratio for five points, *Discrete and Computational Geometry.*

A. Borchers and D.-Z. Du [1995], The k-Steiner ratio in graphs, in *Proceedings of 27th STOC.*

A. Borchers, D.-Z. Du, B. Gao and P.-J. Wan [1998], The k-Steiner ratio in the rectilinear plane, *Journal of Algorithms,* 29, 1-7.

M. Brazil, T. Cole, J.H. Rubinstein, D.A. Thomas, J.F. Weng, and N.C. Wormald [1996], Minimal Steiner trees for $2^k \times 2^k$ square lattices, *Journal of Combinatorial Theory, Series A* 73, 91-110.

P.J. Campbell [1992], Mathematics, *Science and the Future, 1992 Year Book,* Encyclopaedia, Britannica, Inc., London, 373-376.

L.L. Cavalli-Sforza and A.W. Edwards [1967], Phylogenetic analysis: models and estimation procedures, *Am. J. Human Genetics,* 19, 233-257.

S.-K. Chang [1972], The generation of minimal trees with a Steiner topology, *J. ACM,* 19, 699-711.

M. Charikar, C. Chekuri, T. Cheung, Z. Dai, A. Goel, S. Guha, and M. Li [1999], Approximation algorithms for directed Steiner problems. *J. Algorithms* 33, no. 1, 73–91.

D. Chen, D.-Z. Du, X. Hu, G.-H. Lin, L. Wang, and G. Xue [2000], Approximations for Steiner trees with minimum number of Steiner points, accepted by *Journal of Global Optimization.*

F.R.K. Chung and E.N. Gilbert [1976], Steiner trees for the regular simplex, *Bull. Inst. Math. Acad. Sinica,* 4, 313-325.

F.R.K. Chung and R.L. Graham [1985], A new bound for euclidean Steiner minimum trees, *Ann. N.Y. Acad. Sci.,* 440, 328-346.

F.R.K. Chung and F.K. Hwang [1978], A lower bound for the Steiner tree problem, *SIAM J.Appl.Math.,* 34, 27-36.

D. Cieslik [1990], The Steiner ratio of Banach-Minkowski planes, *Contemporary Methods in Graph Theory* (ed. R. Bodendiek), BI-Wissenschatteverlag, Mannheim, 231-248.

D. Cieslik [1998], *Steiner Minimal Trees,* Kluwer.

B.A. Cipra [1990], In math, less is more - up to a point, *Science,* 1081-1082.

B.A. Cipra [1991], Euclidean geometry alive and well in the computer age, *SIAM News,* 24:1.

D.-Z. Du, B. Gao, R.L. Graham, Z.-C. Liu, and P.-J. Wan [1993], Minimum Steiner trees in normed planes, *Discrete and Computational Geometry,* 9, 351-370.

D.Z. Du and F.K. Hwang [1983], A new bound for the Steiner ratio, *Trans. Amer. Math. Soc.* 278, 137-148.

D.Z. Du, F.K. Hwang, and E.Y. Yao [1985], The Steiner ratio conjecture is true for five points, *J. Combinatorial Theory, Series A,* 38, 230-240.

D.Z. Du and F.K. Hwang [1990], The Steiner ratio conjecture of Gilbert-Pollak is true, *Proceedings of National Academy of Sciences,* 87, 9464-9466.

D.-Z. Du, F.K. Hwang [1999], and G. Xue: Interconnecting highways, *SIAM Discrete Mathematics* 12, 252-261.

D.-Z. Du and Z. Miller [1988], Matroids and subset interconnection design, *SIAM J. Disc. Math.,* 1, 416-424.

D.-Z. Du, L.-Q. Pan, and M.-T. Shing [1986], Minimum edge length guillotine rectangular partition, *Report 02418-86, Math. Sci. Res. Inst.,* Univ. California, Berkeley, CA 1986.

D.-Z.Du and P.M. Pardolas [1994], A continuous version of a result of Du and Hwang, *Journal of Global Optimization,* 5, 127-129.

D.-Z. Du and W.D. Smith [1996], Three disproofs for Gilbert-Pollak conjecture in high dimensional spaces, *Journal of Combinatorial Theory,* 74, 115-130.

D.Z. Du, Y. Zhang, and Q. Feng [1991], On better heuristic for euclidean Steiner minimum trees, *Proceedings 32nd FOCS,* 431-439.

P. Duchet [1978], Propriete de Helly et problemes de representation, in: *Colloque International Paris-Orsay* 260, 117-118.

C. Duin [1991], T. Volgenant, The multi-weighted Steiner tree problem, *Annals of Operations Research* 33, 451-469.

J. Friedel and P. Widmayer [1989], A simple proof of the Steiner ratio conjecture for five points, *SIAM J. Appl. Math.* 49, 960-967.

L.R. Foulds and R.L. Graham [1982], The Steiner problem in Phylogeny is NP-complete, *Advanced Applied Mathematics,* 3, 43-49.

B. Gao, D.-Z. Du, and R.L. Graham [1995], A tight lower bound for the Steiner ratio in Minkowski planes, *Discrete Mathematics,* 142, 49-63.

M.R. Garey, R.L. Graham and D.S. Johnson [1977], The complexity of computing Steiner minimal trees, *SIAM J. Appl. Math.,* 32, 835-859.

M.R. Garey and D.S. Johnson [1977], The rectilinear Steiner tree is NP-complete, *SIAM J. Appl. Math.,* 32, 826-834.

E.N. Gilbert and H.O. Pollak [1968], Steiner minimal trees, *SIAM J. Appl. Math.,* 16, 1-29.

R.L. Graham and F.K. Hwang [1976], Remarks on Steiner minimal trees, *Bull. Inst. Math. Acad. Sinica,* 4, 177-182.

F.K. Hwang [1976], On Steiner minimal trees with rectilinear distance, *SIAM J. Appl. Math.,* 30, 104-114.

F.K. Hwang [1990], Flag down but sail up, *Mathematical Dissemination* 12: 2, 5-7.

F.K. Hwang, D.S. Rechards, and P. Winter [1992], *Steiner tree problems* North-Holland, Amsterdam.

A. Iwainsky [1985], Optimal trees - a short overview on problem formulations, in A. Iwainsky (ed.) *Optimization of Connections Structures in Graphs,* CICIP, East Berlin, GDR, 121-133.

E. Ihler, G. Reich, and P. Widmayer [1999], Class Steiner trees and VLSI-design, *Discrete Applied Mathematics* 90, 173-194.

T. Jiang, E.L. Lawler, and L. Wang [1994], Aligning sequences via an evolutionary tree: complexity and approximation, *Proceedings of 26th STOC.*

T. Jiang and L. Wang [1994], An approximation scheme for some Steiner tree problems in the plane, *LNCS* 834, 414-422.

R.M. Karp [1972], Reducibility among combinatorial problems, in R.E. Miller and J.W. Tatcher (ed.), *Complexity of Computer Computation,* Plenum Press, New York, 85-103.

M. Karpinski and A.Z. Zelikovsky [1997], New approximation algorithms for the Steiner tree problems, *Journal of Combinatorial Optimization,* 1, 47-65.

G. Kolata [1990], Solution to old puzzle: how short a shortcut? *New York Times* October 30.

P. Korthonen [1979], An algorithm for transforming a spanning tree into a Steiner tree, *Survey of Mathematical Programming (2),* North-Holland, 349-357.

L. Kou and K. Makki [1987], An even faster approximation algorithm for the Steiner tree problem in graph, *Congressus Numerantium* 59, 147-154.

C.-S. Li, F.F. Tong, C.J. Georgiou and M. Chen [1994], Gain equalization in metropolitan and wide area optical networks using optical amplifiers, *Proc. IEEE INFOCOM'94,* pp. 130–137, June.

G.-H. Lin and G.L. Xue [1999], Steiner tree problem with minimum number of Steiner points and bounded edge-length, *Information Processing Letters,* 69, 53–57.

B. Lu, J. Gu, X. Hu, and E. Shragowitz [2000], Wire segmenting for buffer insertion based on RSTP-MSP, accepted by *Theoretical Computer Science.*

B. Lu and L. Ruan [2000], Polynomial-time approximation scheme for rectilinear Steiner arborescence problem, *Journal of Combinatorial Optimization,* 357-363.

M.S. Manase, L.A. McGeoch, and D.D. Sleator [1988], Competitive algorithms for on-line problems, Proc. of 20th STOC, Chacago.

J. S. B. Mitchell [1996], Guillotine subdivisions approximate polygonal subdivisions: A simple new method for the geometric k-MST problem, *Proc. 7th ACM-SIAM Sympos. Discrete Algorithms,* 402-408.

J. S. B. Mitchell [1997], Guillotine subdivisions approximate polygonal subdivisions: Part III - Faster polynomial-time approximation scheme for geometric network optimization," *Proc. ninth Canadian conference on computational geometry,* 229-232.

J. S. B. Mitchell [1999], Guillotine subdivisions approximate polygonal subdivisions: Part II - A simple polynomial-time approximation scheme for geometric k-MST, TSP, and related problems, *SIAM J. Comp.* 28, 1298-1309.

J. S. B. Mitchell, A. Blum, P. Chalasani, and S. Vempala [1999], A constant-factor approximation for the geometric k-MST problem in the plane, *SIAM J. Comp.* 28, 771-781.

I. Peterson [1990], Proven path for limiting shortest shortcut, *Science News,* December 22 & 29, 389.

H.O. Pollak [1978], Some remarks on the Steiner problem, *J. Combinatorial Theory, Ser.A,* 24, 278-295.

E. Prisner [1992], Two algorithms for the subset interconnection design, *Networks* 22, 385-395.

W.R. Pulleyblank [1995], Two Steiner tree packing problems, 27th STOC, pp.383-387.

B. Ramamurthy, J. Iness and B. Mukherjee [1997], Minimizing the number of optical amplifiers needed to support a multi-wavelength optical LAN/MAN, *Proc. IEEE INFOCOM'97,* pp. 261–268, April.

S.K. Rao, P. Sadayappan, F.K. Hwang, and P.W. Shor [1992], The rectilinear Steiner arborescence problem, *Algorithmica* 7, 277-288.

J.H. Rubinstein and D.A. Thomas [1991], The Steiner ratio conjecture for six points, *J. Combinatoria Theory, Ser.A,* 58, 54-77.

A. Sangalli [1991], Mathematicians learn how to join the dots, *New Scientists* April, 22.

P. Schreiber [1986], On the history of the so-called Steiner weber problem, *Wiss. Z. Ernst-Moritz-Arndt-Univ. Greifswald, Math.-nat.wiss. Reihe,* 35, no.3.

W. Shi and C. Su [2000], The rectilinear Steiner arborescence problem is NP-complete, *Proc. 11th ACM-SIAM Symposium on Discrete Algorithms (SODA).*

D.D. Sleator and R.E. Tarjan [1985], Amortized efficiency of list uptate and paging rules, *Commications of ACM* 28, 202-208.

J.M. Smith and J.S. Liebman [1979], Steiner trees, Steiner circuits and interference problem in building design, *Engineering Optimization,* 4, 15-36.

J.M. Smith, D.T. Lee, and J.S. Liebman [1981], An $O(N \log N)$ heuristic for Steiner minimal tree problems in the Euclidean metric, *Networks,* 11, 23-39.

W.D. Smith [1992], How to find Steiner minimal trees in euclidean d-space, *Algorithmica,* 7.

W.D. Smith and P.W. Shor [1992], Steiner tree problems, *Algorithmica,* 7, 329-332.

I. Stewart [1991], Trees, telephones and tiles, *New Scientist* 16 (1991) 26-29.

R.E. Tarjan and M. Yannakakis [1984], Simple linear-time algorithms to test chordality of graphs, test acyclicity of hypergraphs, and selectively reduce acyclic hypergraphs, *SIAM J. Comput.* 13, 566-579.

Y.T. Tsai, C.Y. Tang, and Y.Y. Chen [1996], An average case analysis of a greedy algorithm for the on-line Steiner tree problem, *Comput. Math. Appl.* 31, no. 11, 121–131.

P.-J. Wan, D.-Z. Du, and R.L. Graham [1997], The Steiner ratio on the dual normed plane, *Discrete Mathematics* 171, 261-275.

L. Wang and D.-Z. Du [2000], Approximations for a Bottleneck Steiner Tree Problem, submitted for publication.

B.M. Waxman and M. Imase [1988], Worst-case performance of Rayward-Smith's Steiner tree heuristic, *Inform. Process. Lett.* 29, 283-287.

J. Westbrook and D.C.K. Yan [1995], The performance of greedy algorithms for the on-line Steiner tree and related problems, *Math. Systems Theory* 28, no. 5, 451–468.

Y.F. Wu, P. Widmayer, and C.K. Wong [1986], A faster approximation algorithm for the Steiner problem in graphs, *Acta Informatica,* 23, 223-229.

A.Z. Zelikovsky [1993], The 11/6-approximation algorithm for the Steiner problem on networks, *Algorithmica* 9, 463-470.

HANDBOOK OF COMBINATORIAL OPTIMIZATION (VOL 5)
D.-Z. Du and P.M. Pardalos (Eds.) pp. 217-258

Network-based Models and Algorithms in Data Mining and Knowledge Discovery

Vladimir Boginski
Center for Applied Optimization, ISE Department
University of Florida, Gainesville, FL 32611
E-mail: vb@ufl.edu

Panos M. Pardalos
Center for Applied Optimization, ISE Department
University of Florida, Gainesville, FL 32611
E-mail: pardalos@ufl.edu

Alkis Vazacopoulos
Dash Optimization, Inc.
Englewood Cliffs, NJ 07632
E-mail: av@dashoptimization.com

Contents

1 Introduction

Nowadays, scientists, engineers and businessmen working in diverse fields have one thing in common: they all have to deal with large datasets which arise in a broad spectrum of areas, including finance, banking, manufacturing, supply chain, medicine and biotechnology, telecommunications, military systems, etc [2]. In many cases, practitioners collect historical data of a certain type and try to use it for solving problems they encounter. Among the examples of such historical data, one can mention recordings of medical devices that are used for disease diagnoses and therapy planning, prices and returns of financial instruments traded in the stock market, history of telephone calls made between different numbers, credit card transactions, etc. Intuitively, one can say that large amounts of information can potentially be very helpful in making decisions. With the advances in computational equipment technology and the development of efficient external memory algorithms for processing massive amounts of data that cannot fit in the computer memory [3], the issue of storage and formatting massive datasets can be resolved, however, availability of a large set of historical data does not necessarily imply that this data will be efficiently used in making practical decisions. A crucial problem that arises here is choosing an appropriate methodology for processing and analyzing the data in order to extract valuable information from it. In other words, one should find analytical models and algorithms that would reveal internal structure and patterns of the data.

The broad research area that deals with this type of problems is referred to as data mining.

In many cases, in order to discover useful information from a certain dataset, mathematical programming approaches are successfully applied. In the framework of different types of data mining problems, the techniques of formulating and solving optimization problems and developing heuristic algorithms proved to be rather efficient in practice. Optimization techniques are traditionally widely used in the analysis of financial data, especially in the problems of choosing an investment portfolio based on the historical data of the stock prices. Also, mathematical programming approaches have been recently applied to the classification of securities. Besides finance, probably the most fast-growing application area of these techniques is medicine. There is a strong tendency to use optimization models in the analysis of different kinds of data obtained from the patients. Important applications include investigating brain function disorders (in particular, epilepsy), cancer diagnosis and therapy planning, etc.

In this chapter, we discuss one of the promising research directions in data mining – using *network-based* mathematical programming models for data analysis and decision making. In many practical situations, a real-life dataset can be represented as a large *graph (network)* - a structure that can be easily understood and visualized [16]. A graph is a set of vertices (dots) and edges (links) connecting them. When a dataset is represented as a graph, certain attributes are associated with the vertices and edges of the corresponding graph. These attributes may contain specific information characterizing the given application, which often provides a new insight into the internal structure and patterns of the data. In particular, one can locate certain special formations in these graphs: *cliques* and *independent sets,* which have a clear data mining interpretation. The analysis of these structures leads to the consideration of standard graph-based combinatorial optimization problems: maximum clique and independent set, clique partitioning, and coloring. Considered examples of applying network-based approaches to the analysis of real-life datasets include telecommunications, medicine, and finance.

We also describe another network-based approach in data mining, so-called *Artificial Neural Networks,* which is extensively applied in predictive modeling. Although the term "neural network" in this case should not be understood literally, this approach was originally motivated by the principles of the functioning of the brain, where a network is formed by the connections between neurons.

The purpose of this chapter is to describe how mathematical program-

ming (and, in particular, network-based) techniques can assist practitioners representing different branches of science and industry in analyzing the datasets they deal with, and making efficient decisions.

Although this review is not intended to be exhaustive, we will describe several examples that we believe can used as a basis for the development of similar models in various areas.

The remainder of the chapter is organized as follows. In Section 2, we briefly describe the general setup of major data mining problems: predictive modeling and clustering, and mention several standard mathematical programming techniques for solving these problems. Section 3 presents an overview of the neural networks approach applied in predictive modeling. Section 4 deals with network representation of real-world massive datasets and describes the applications of standard graph-theoretical concepts to extracting useful information of these datasets. Finally, Section 5 summarizes the discussion.

2 Major Types of Data Mining Problems: Supervised vs. Unsupervised Learning

We first give a brief review of major types of problems considered in the area of data mining. Two large classes of problems that we discuss here are so-called *predictive modeling* and *clustering*. These problems are often associated with the concepts *"supervised learning"* and *"unsupervised learning"*, respectively. The purpose of this section is to describe the general setup of these problems and to draw conceptual distinctions between them. We will also mention some mathematical programming techniques commonly used for solving these problems. For more detailed description of these and other classes of data mining problems, the reader is referred to [19].

2.1 Predictive modeling: Classification and Regression

Predictive modeling is a class of data mining problems that essentially deals with predicting a certain attribute of an element in a dataset based on the known information about its other attributes (or features). The common setup of these problems assumes the availability of a certain initial dataset containing the elements with all known attributes, and this information is then utilized for constructing a model that predicts unknown attributes of new data elements. One can distinguish two major types of problems considered in this area: *classification* and *regression*. Below, we briefly

summarize the general setup of these problems.

2.1.1 Classification: General Setup

The first subject of our discussion is *classification* – one of the most important data mining problems arising in a great variety of applications. The general setup of this class of problems is as follows. Suppose that we have a dataset of N elements, and each of these elements has a finite number of certain attributes. If we denote the number of attributes as n, then every element of the given dataset can be represented as a pair (x_i, y_i), $i = 1, ..., N$, where $x_i \in R^n$ is an n-dimensional vector:

$$x_i = (x_{i1}, x_{i2}, ..., x_{in})^T,$$

and y_i is the *class attribute.* The value of y defines to which class a given element belongs, and this value is known *a priori* for each element of the initial dataset. It should be also mentioned that in this case y_i can take integer values, and the number of these values (i.e., the number of classes) is pre-defined.

Now suppose that a new element with the known attributes vector x, but unknown class attribute y, is added to the dataset. The essence of classification problems is to predict the unknown value of y. This is accomplished by identifying a criterion of placing the element into a certain class based on the information about the known attributes x of this element. An initial question that naturally arises is how to find a criterion that will correctly classify data elements with a sufficiently high confidence level. The intuitive answer is that the construction of such a criterion should be based on the available information about the elements whose attributes are all known. However, another fundamental question is how to create a formal model that would take the available dataset as the input and perform the classification procedure. Several approaches aimed to utilize this idea have been developed. The main idea of these approaches is to adjust (or, "train") the classification model using the existing information about the elements in the available dataset (which is usually referred to as the "training dataset") and then apply this model to classifying new elements.

One of the techniques widely used in practice deals with the geometrical approach. Recall that since all the data elements can be represented as n-dimensional vectors (or points in the n-dimensional space), then these elements can be separated *geometrically* by constructing the *surfaces* that serve as the "borders" between different groups of points. One of the common approaches is to use linear surfaces (planes) for this purpose, however,

different types of nonlinear (e.g., quadratic) separating surfaces can be considered in certain applications.

It is also important to note that usually it is not possible to find a surface that would "perfectly" separate the points according to the value of some attribute, i.e. points with different values of the given attribute may not necessarily lie at the different sides of the surface, however, in general, the number these errors should be small enough.

So, according to this approach, the classification problem is represented as the problem of finding geometrical parameters of the separating surface(s). As it will be described below, these parameters can be found by solving the optimization problem of minimizing the misclassification error for the elements in the training dataset (so-called "in-sample error"). After determining these parameters, every new data element will be automatically assigned to a certain class, according to its geometrical location in the elements space.

The procedure of using the existing dataset for classifying new elements is often called "training the classifier" (and the corresponding dataset is referred to as the "training dataset"). It means that the parameters of separating surfaces are "tuned" (or, "trained") to fit the attributes of the existing elements to minimize the number of errors in their classification. However, a crucial issue in this procedure is not to "overtrain" the model, so that it would have enough flexibility to classify *new* elements, which is the primal purpose of constructing the classifier.

To illustrate the basic principles of applying the geometrical classification approach described above, we consider one of the first practical applications of mathematical programming in classification problems developed by Mangasarian *et al.* [49]. This study deals with the diagnosis of breast cancer cases. Note that different kinds of disease diagnoses are very common in medical practice, and any diagnosis implies some kind of classification, therefore, applications of classification problems in medicine naturally arise, and the interdisciplinary research area of utilizing mathematical programming in the analysis of medical data is developing very rapidly nowadays.

The essence of the breast cancer diagnosis system developed in [49] is as follows. The authors considered the dataset consisting of 569 30-dimensional feature vector corresponding to each patient. Each case could be classified as malignant or benign, and the actual diagnosis was known for all the elements in the dataset. These 569 elements were used for "training" the classifier, which was developed based on linear programming (LP) techniques. The procedure of constructing this classifier is relatively simple. The vectors corresponding to malignant and benign cases are stored in two matrices.

The matrix $A(m \times n)$ contains m malignant vectors (n is the dimension of each vector), and the matrix $B(k \times n)$ represents k benign cases.

The goal of the constructed model is to find a plane which would separate all the vectors (points in the n-dimensional space) in A from the vectors in B. If a plane is defined by the standard equation

$$x^T\omega = \gamma,$$

where $\omega = (\omega_1, \ldots, \omega_n)^T$ is an n-dimensional vector of real numbers, and γ is a scalar, then this plane will separate all the elements from A and B if the following conditions are satisfied:

$$A\omega \geq e\gamma + e, \quad B\omega \leq e\gamma - e. \tag{1}$$

Here $e = (1, 1, \ldots, 1)^T$ is the vector of ones with appropriate dimension (m for the matrix A and k for the matrix B).

However, as it was pointed out above, in practice it is usually not possible to perfectly separate two sets of elements by a plane. So, one should try to minimize the average measure of misclassifications, i.e., in the case when the constraints (1) are violated the average sum of violations should be as small as possible. The violations of these constraints are modeled by introducing nonnegative variables u and v as follows:

$$A\omega + u \geq e\gamma + e, \quad B\omega - v \leq e\gamma - e. \tag{2}$$

Now we are ready to write down the optimization model that will minimize the total average measure of misclassification errors as follows:

$$\min_{\omega,\gamma,u,v} \quad \frac{1}{m}\sum_{i=1}^{m} u_i + \frac{1}{k}\sum_{j=1}^{k} v_j \tag{3}$$

subject to

$$A\omega + u \geq e\gamma + e \tag{4}$$

$$B\omega - v \leq e\gamma - e \tag{5}$$

$$u \geq 0, \quad v \geq 0. \tag{6}$$

As one can see, this is a linear programming problem, and the decision variables here are the geometrical parameters of the separating plane ω and γ, as well as the variables representing misclassification error u and v. Although in many cases this type of problems may involve high dimensionality of data, they can be efficiently solved by available LP solvers, for instance Xpress-MP or CPLEX.

The misclassification error that is minimized here is usually referred to as the *in-sample error,* since it is measured for the training sample dataset. Note that if the in-sample error is unacceptably high, the classifying procedure can be repeated for each of the subsets of elements in the halfspaces generated by the separating plane. As a result of such a procedure several planes dividing the elements space into subspaces will be created, which is illustrated by Figure 1. Then every new element will be classified according to its location in a certain subspace. If we consider the case of only one separating plane, then after solving the above problem, each new cancer case is automatically classified into either malignant or benign class as follows: if the vector x corresponding to this case satisfies the condition $x^T\omega > \gamma$ it is considered to be malignant, otherwise it is assumed to be benign.

The approach described here is rather general and can be applied to any real-life dataset whose elements are classified into two classes. It should be noted that this technique can be generalized for the case of multiple nonlinear separating surfaces. For instance, the LP problems corresponding to classification using quadratic separating surfaces are presented in [24].

As it was mentioned above, the procedure of using the existing dataset for classifying new elements is often called "training the classifier". It means that the parameters of separating surfaces are "tuned" (or, "trained") to fit the attributes of the existing elements to minimize the number of errors in their classification. However, a crucial issue in this procedure is not to "overtrain" the model, so that it would have enough flexibility to classify *new* elements, which is the primal purpose of constructing the classifier. If the training sample is too large, then it is possible that the classifier will adjust very well to the details of this data, and it won't have enough flexibility to classify the unknown elements, which will increase the generalization (or, "out-of-sample") error, i.e., the error in classifying new elements. In [49], the authors indicate that even one separating plane can be an overtrained classifier if the number of attributes in each vector is too large. They point out that the best out-of-sample results were achieved when only three attributes of each vector were taken into account, and one separating plane

was used.

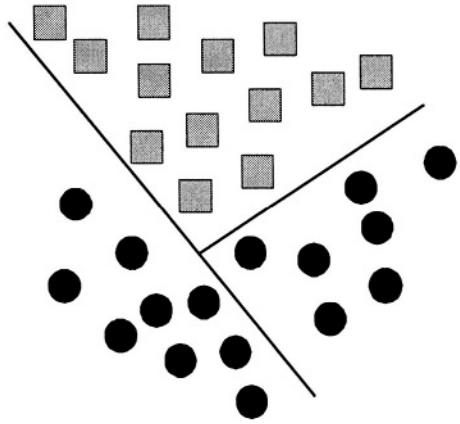


Figure 1: An example of binary classification using linear separating surfaces.

These arguments lead to introducing the following concepts closely related to classification: *feature selection* and *support vector machines (SVMs).*

The main idea of *feature selection* is choosing a *minimal number of attributes* (i.e., components of the vector $\boldsymbol{x}$ corresponding to a data element) that are used in the construction of separating surfaces [20]. This procedure is often important in practice, since it may produce a better classification in the sense of the out-of-sample error. Moreover, in the case of high dimensionality of the initial problem, choosing the minimal number of features will significantly simplify the model.

The essence of *support vector machines* is to construct separating surfaces that will minimize the *upper bound on the out-of-sample error.* In the case of one linear surface (plane) separating the elements from two classes, this approach will choose the plane that maximizes the sum of the distances between the plane and the closest elements from each class, i.e., the "gap" between the elements from different classes [26, 55, 56, 64, 67].

2.1.2 Regression: General Setup

The general setup of regression problems is rather similar to the classification setup described above. In this case, the sample elements of a given

dataset are represented by pairs (x_i, y_i), $i = 1, ..., N$, where $x_i \in R^n$ is an n-dimensional vector, and $y_i \in R$. The goal of regression problems is to estimate the unknown *regression function* $y(x)$ based on the available sample elements. As one can see, the difference between classification and regression is the fact that unlike classification, the attribute y which needs to be predicted is *continuous* in the case of regression. In the simplest case, when the relationship $y(x)$ is linear (linear regression), one needs to find a vector β such that $y = \beta^T x$.

The common problem setup in this case is finding the vector β that minimizes the norm $||A\beta - y||$, where A is the matrix containing all vectors x_i, $i = 1, ...N$ in its rows, and $y = (y_1, y_2, ..., y_N)^T$ is the vector representing the values y_i for each data element. Among the norms that can be considered are the 2-norm $\sum_{i=1}^{N}(y_i - \beta^T x_i)^2$ and the 1-norm $\sum_{i=1}^{N}|y_i - \beta^T x_i|$. If minimization of the 1-norm is considered, the problem can be easily reduced to a linear program. If the array of new variables z_i representing the differences $(y_i - \beta^T x_i)$ is introduced, then the problem

$$\min_{\beta} \sum_{i=1}^{N} |y_i - \beta^T x_i| \tag{7}$$

can be reformulated as

$$\min_{\beta, z} \sum_{i=1}^{N} z_i \tag{8}$$

subject to

$$-z_i \leq y_i - \beta^T x_i \leq z_i, \quad \forall i = 1, ..., N. \tag{9}$$

It should be noted that this approach to solving regression problems can be generalized to the case when sample data involves noise, moreover, the Support Vector techniques can be modified for dealing with regression problems [66].

Besides continuous optimization formulations, it is also possible to apply integer programming techniques to regression and classification. In [15], the authors propose an alternative approach to predictive modeling which consecutively solves mixed-integer and continuous optimization problems. This technique proved to be rather effective for several real-life case studies.

Later in this chapter, we will also consider another approach to predictive modeling that utilizes the concept of Artificial Neural Networks.

2.2 Clustering: General Setup and Standard Algorithms

Another class of problems widely considered in data mining is referred to as *clustering* (or *segmentation*) [8, 14, 41, 51]. The essence of clustering is partitioning the elements in a certain dataset into several distinct subsets (clusters) grouped according to an appropriate *similarity criterion.* Identifying the groups of objects that are "similar" to each other but "different" from other objects in a given dataset is important in many practical applications.

At the first glance, one can say that the setups of clustering and classification problems are rather similar, since in both cases one tries to "classify" data elements. However, one should clearly understand the essential difference between these two types of problems. As it was pointed out above, the main idea of classification is utilizing the training dataset, whose elements are already classified, and their class attributes are known, i.e., in this case one uses certain a-priori information about the data in order to classify objects. On the contrary, in the clustering problem, the training dataset is not available, and one does not use any initial information about the class attributes of the existing data elements, but tries to *determine a classification* using appropriate criteria.

Besides identifying the similarity criterion, a major difficulty arising in clustering is the fact that the number of clusters in the partitioning of a dataset is usually not known a-priori, which makes this problem non-trivial and challenging.

One of the most common approaches to addressing this type of problems is introducing *metric distances* between data elements. More specifically, if we consider N data points $\{x^1, x^2, ..., x^N\}$, where each point is an n-dimensional vector of attributes ($x^i \in R^n$), we can define a certain distance measure between each pair of elements i and j as

$$d(x^i, x^j) = ||x^i - x^j||,$$

where $||x^i - x^j||$ is some norm in R^n. This value can be used as a measure of "similarity" or "dissimilarity" between a given pair of data elements. It should be noted that this procedure is closely related to the technique of representing a dataset as a graph $G = (V, E)$, where V is the set of N vertices, which corresponding to N data points, and E is the set of edges connecting these points according to a certain rule. An obvious way of constructing the set of edges is to connect every pair of vertices by an edge and assign every edge a *weight* equal to the distance between the corresponding vertices. However, it is a common practice to introduce a certain threshold

on the distance measure and consider only the edges for which the value of the corresponding distance is greater than this threshold. If one deals with a massive dataset, this would help to to make the connectivity matrix of G sparser and simplify the analysis. In these settings, the clustering problem is essentially the problem of optimal partitioning of the graph G into several subsets of vertices, each of which will correspond to a certain cluster. This approach, along with related techniques and real-life examples, will be discussed in more detail in Section 4.

Now, suppose that a dataset needs to be partitioned into a *fixed* number of clusters k. In this case, the clustering problem can be formally defined as follows: find k "centers" $\{c^1, c^2, ..., c^k\}$ $(c^i \in R^n)$ such that this selection would minimize the sum of the distances between every data point and a cluster center closest to it. It can be mathematically formulated as

$$\min_{c^1,...,c^k} \sum_{i=1}^{N} \min_j ||x^i - c^j||,$$

where $||x^i - c^j||$ is some metric in R^n. Popular standard methods applied to solving this optimization problem are referred to as $k-median$ and $k-mean$ algorithms. Both of these techniques utilize the same idea of finding the optimal selection of cluster centers by performing consecutive *iterations.* At each iteration, every data element is assigned to a nearest cluster center c^j, where the distance between a data element and a cluster center is measured in terms of the 1-norm in the case of k-median algorithm, and the 2-norm in the case of k-mean algorithm. The set of cluster centers is then updated as follows: the location of each cluster center c^j is moved to the point that minimizes the sum of the distances from the previous cluster center to all data points assigned to cluster j. If these distances are measured in terms of the 1-norm, the new cluster center c^j represents the *median* of all data points assigned to cluster j (k-median algorithm). If the 2-norm is used, then the new cluster center would be the *mean* of the data points assigned to cluster j (k-mean algorithm). The iterative process is stopped if after some iteration no cluster centers are updated.

The difference between these two methods is the fact that in the case of k-median algorithm the cluster center would always be located in some data point (the most appropriate data point representing the cluster is chosen), whereas in the case of k-mean algorithm the location of a cluster center may not coincide with any data point. Clearly, the performance of the k-mean algorithm can be negatively affected by the presence of "outliers", i.e., data points that are significantly "different" from the others, since these

points would considerably affect the location of the mean of a group of points. On the contrary, the k-median algorithm is less sensitive to outliers, because in this case the cluster center is always located in a data point, and a small number of outliers is unlikely to influence this location. Several computational experiments confirm that the k-median algorithm produces better results than the k-mean algorithm in the case of breast cancer datasets [19].

For an extensive review and formal description of various clustering techniques, the reader is referred to [14, 51].

3 Data Mining Using Artificial Neural Networks

In the next section, we give a brief overview of one of the popular network-based approaches in data mining which is associated with the concept of so-called *Artificial Neural Networks (ANNs)* [34, 37, 28]. This type of models was designed for extracting information from datasets using natural principles similar to the functioning of the human brain. As we will see, this approach is applicable to both predictive modeling and clustering.

3.1 Basic Principles of Artificial Neural Networks

The main components of an ANN are *artificial neurons,* which are organized in a specific order: they form several *layers* that affect each other according to certain rules. In a typical setup, a neural network has an *input layer* corresponding to the initial (input) information, an *output layer* that produces the output of the network, and several *intermediate (hidden) layers* that are responsible for processing the data received from the input layer in order to generate the final output. In the common case when there are no feedback connections in the network (Feedforward Neural Networks, or FNNs), the interaction between neurons works as follows: the output of a neuron in layer j is the input of all the neurons from the next layer $j + 1$, as shown in Figure 2.

Similarly to the notations introduced in the previous section, we denote x and y to be the vectors of inputs and outputs respectively.

The main concepts utilized in the operation of a neural network are the *weight vector* w and the *activation function* f.

Each element of the weight vector w_i corresponds to an input x_i, $i = 1, ..., n$, and the operation performed by each neuron is calculating a *weighted sum* of the inputs with the weights corresponding to the components of w.

This weighted sum is then used as the argument of the activation function f, which produces the output of the neuron.

Therefore, the operations performed by a neuron that receives the vector x as input, can be mathematically represented as:

$$\xi = \sum_{i=0}^{n} w_i x_i, \tag{10}$$

$$y = f(\xi). \tag{11}$$

The examples of the commonly used types of activation functions are presented below:

- *hard limit* function:

$$f(\xi) = \begin{cases} 1, & \text{if } \xi \geq h, \\ 0, & \text{if } \xi < h, \end{cases}$$

- *piecewise linear* function:

$$f(\xi) = \begin{cases} 1, & \text{if } \xi \geq 1, \\ 0, & \text{if } \xi \leq 0, \\ \xi, & \text{if } 0 < \xi < 1, \end{cases}$$

- *logsig* function:

$$f(\xi) = \frac{1}{1+e^{-\xi}},$$

- *hyperbolic tangent* function:

$$f(\xi) = \tanh\frac{\xi}{2} = \frac{1-e^{-\xi}}{1+e^{-\xi}}.$$

The functioning of a neural network can be represented as the process of consecutive activation of the neurons in different layers, starting from the input layer. Initially, the states of the neurons in the input layer are assigned to the values of the components of the corresponding input vector, and all the other neurons (in the hidden and output layers) are not activated. Next, the information from the input layer is passed to the neurons in the first hidden layer, and their output is determined according to (10)–(11). This process is then repeated for all the other layers of the neural network until the output layer is reached and the final output is obtained.

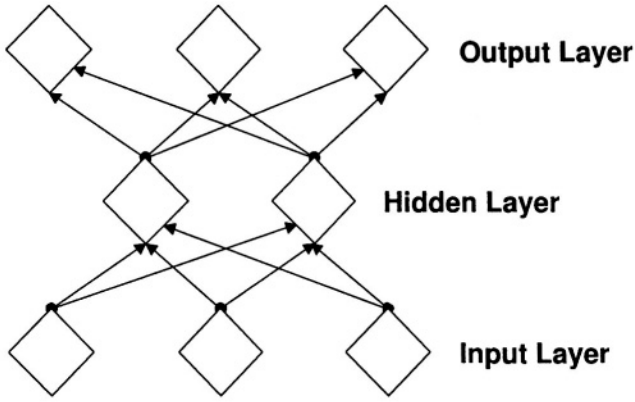


Figure 2: An example of a Feedforward Artificial Neural Network

3.2 Supervised Training of Neural Networks

Similarly to the predictive modeling approaches considered above, the parameters of a neural network can be adjusted to a specific training dataset.

To illustrate the main idea of supervised training of a neural network, we consider the training dataset consisting of N elements, each of which is represented by a pair (x_i, y_i), $i = 1, ..., N$, where x_i is a vector of attributes, and y_i is a *known output* (for instance, the class attribute in the case of classification).

The goal of the training procedure is to determine the optimal values of the parameters (weights) w, that will minimize the difference between the network output and the actual output for the elements in the training dataset. The main idea of training is to pass the elements from the training dataset (as inputs) to the network and compute the *error* corresponding to each training element. If $z_i(w, x_i)$ denotes the network output for the i-th training data element, the network error E_i corresponding to this element can be defined as

$$E_i(w) = \frac{1}{2}\Big(z_i(w, x_i) - y_i\Big)^2.$$

One can then calculate the *total error* as the sum of error terms corresponding to each element from the training dataset:

$$E(w) = \sum_{i=1}^{N} E_i(w).$$

The problem of supervised training of a neural network in the general case can be represented as

$$\min_{w} E(w).$$

Note that the setup of the problem of training a neural network is essentially the same as for the mathematical programming formulations of predictive modeling problems considered in Section 2. In all these cases, a training dataset is used for tuning the parameters of the model, and this model is then applied to new data elements. It is also important to mention that in many cases one can relate the process of training neural networks to the mathematical programming techniques for classification and regression discussed in Section 2. For instance, the LP formulation of the binary classification problem with linear separating surfaces can be put in the framework of neural networks [12, 48].

After a neural network is trained, and the "optimal" weights w are determined, one can use this network to obtain the output corresponding to new data elements passed to the network as the input.

As in the case of any predictive modeling technique, the crucial issue in training a neural network is producing an adequate output for the new elements, in other words, the *generalization error* of a neural network should be sufficiently small. If a neural network exhibits a poor generalization performance, one can say that overtraining the model takes place. In different practical applications, various techniques are applied to avoid overfitting, for instance, in [59] the authors utilize the concept of so-called Distributed Feedforward Neural Networks in order to efficiently forecast currency exchange rates. However, it is clear that the prediction quality of a certain neural network model depends on the structure of the considered dataset.

3.3 Neural Networks in Clustering

Besides predictive modeling, the Artificial Neural Networks approach can also be utilized in clustering. Although, as it was pointed out above, in this case the training dataset is not available, it is possible to "train" a neural network so that it will assign the input elements to certain clusters using a specific activation rule (unsupervised neural network algorithms). One of

the well-known methods in this area is referred to as *competitive learning* [62]. If the number of clusters k is *fixed,* then one can consider a single-layer neural network with k output units (neurons), each of which corresponds to a certain cluster. At each iteration, this network takes the attributes of one data element as the input and adjusts the weight vector w corresponding to each output neuron.

Without specifying all the details of this approach, we need to point out that it is closely related to the k-mean algorithm mentioned in the previous section in the sense that after "training" the network, the vector of weights w for each output neuron (cluster) represents the *mean* of all the data points assigned to the cluster corresponding to this neuron. However, the essential difference between the k-mean and competitive learning algorithms is the fact that in the process of determining the optimal set of cluster centers, k-mean algorithm uses *all* data elements simultaneously at each iteration, whereas the competitive learning method takes data elements as inputs *one-by-one,* and uses only one data element to adjust the vector w at each step. The latter approach may be more efficient for dealing with massive datasets, since in the case of large amounts of data each iteration of the k-mean algorithm is computationally hard to perform.

4 Network Representation of Massive Datasets

In this section, we describe another network-based approach to extracting information from massive datasets, which has been rapidly emerging and developing during the last several years. According to this approach, a certain dataset is represented as a graph with certain attributes associated with its vertices and edges. Unlike artificial neural networks that use the data elements only as inputs, this methodology allows one to visualize a dataset by representing its elements as vertices and observe certain relationships between them.

Studying the structure of a graph representing a dataset is important for understanding the internal properties of the application it represents, as well as for improving storage organization and information retrieval. One can visualize a graph as a set of dots and links connecting them, which in many cases makes this representation convenient and easily understandable.

The main concepts of graph theory were founded several centuries ago, and many network optimization algorithms have been developed since then. However, only recently have the applications of graphs to representing various real-life massive datasets started to be widely discussed in the literature.

Nowadays, graph theory is becoming a more and more practical field of science. The expansion of graph-theoretical approaches in various data mining applications gave a birth to the terms "graph practice" and "graph engineering" [36].

As it was pointed out above, graph representation of a dataset is often useful in the context of clustering. Moreover, a graph representing a real-life dataset can be analyzed from different aspects, and one can apply standard models and algorithms from graph theory to study the properties of this graph in order to obtain the information about the initial dataset. Later in this section, we will provide several real-life examples related to this methodology.

4.1 Basic Concepts from Graph Theory and their Data Mining Interpretation

To give a brief introduction to graph theory, we introduce several basic definitions and notations. Let $G = (V, E)$ be an *undirected* graph with the set of n vertices V and the set of edges E. *Directed* graphs, where the head and tail of each edge are specified, are considered in some applications. The concept of a *multigraph* is also sometimes introduced. A multigraph is a graph where multiple edges connecting a given pair of vertices may exist. An important characteristic of a graph is its *edge density,* which represents the ratio of the number of edges in the graph to the maximum possible number of edges.[1]

4.1.1 Connectivity and Degree Distribution

The graph $G = (V, E)$ is *connected* if there is a path from any vertex to any vertex in the set V. If the graph is disconnected, it can be decomposed into several connected subgraphs, which are referred to as the *connected components* of G.

The *degree* of the vertex is the number of edges emanating from it. For every integer number k one can calculate the number of vertices $n(k)$ with the degree equal to k, and then get the probability that a vertex has the degree k as $P(k) = n(k)/n$, where n is the total number of vertices. The function $P(k)$ is referred to as the *degree distribution* of the graph. In the case of a directed graph, the concept of degree distribution is generalized:

[1]The maximum possible number of edges in a graph is equal to $n(n-1)/2$, where n is the number of vertices.

one can distinguish the distribution of *in-degrees* and *out-degrees,* which deal with the number of edges ending at and starting from a vertex, respectively.

The degree distribution is an important characteristic of a dataset represented by a graph. It reflects the large-scale pattern of connections in the graph, which in many cases reflects the global properties of the dataset this graph represents. One of the important facts discovered during the last several years is the observation that many real-life massive graphs representing the datasets coming from diverse areas (Internet, telecommunications, finance, biology, sociology) have degree distributions that follow the *power-law* model. According to this model, the probability that a vertex has the degree k is

$$P(k) \propto k^{-\gamma}, \tag{12}$$

or, equivalently,

$$\log P(k) \propto -\gamma \log k,$$

which shows that this distribution can be plotted as a straight line in the logarithmic scale.

The amazing fact that graphs representing completely different datasets have a similar well-defined power-law structure has been widely reflected in the literature [7, 11, 10, 16, 36, 68, 69]. It indicates that the global organization and evolution of massive datasets arising in various spheres of life nowadays follow similar laws and patterns. This fact served as a motivation to introduce a concept of "self-organized networks". These networks are also associated with a famous "small-world" hypothesis, which claims that despite the large number of vertices, the distance between any two vertices (or, the *diameter* of the graph) is small [68].

From another perspective, some properties of the graphs that follow the power-law model can be predicted theoretically. Aiello et al. [6] studied the properties of the power-law graphs with respect to the parameter γ in (12). Among their results, one can mention the existence of a giant connected component in a power-law graph with $\gamma < \gamma_0 \approx 3.47875$, and the fact that a giant connected component does not exist otherwise.[2] The emergence of a giant connected component at the point $\gamma_0 \approx 3.47875$ is referred to as *phase transition.*

[2]These results are valid *asymptotically almost surely (a.a.s.),* which means that the probability that a given property takes place tends to 1 as the number of vertices n goes to infinity.

The size of connected components of the graph may provide useful information about the structure of the corresponding dataset, as the connected components would normally represent the groups of "similar" objects. In some applications, decomposing the graph into a set of connected components can provide a reasonable solution to the clustering problem, i.e., the partitioning of the graph into subgraphs, each of which corresponds to a certain cluster.

In the next subsection, we will discuss more specific types of graph structures that correspond to very dense clusters of "similar" or "different" objects. These formations are referred to as *cliques* and *independent sets.* As we will see below, these concepts are important in the context of clustering.

4.1.2 Cliques and Independent Sets

Given a subset $S \subseteq V$, by *G(S)* we denote the subgraph induced by *S.* A subset $C \subseteq V$ is a *clique* if *G*(*C*) is a complete graph, i.e., it has all possible edges. The maximum clique problem is to find the largest clique in a graph.

The following definitions generalize the concept of clique. Namely, instead of cliques one can consider dense subgraphs, or *quasi-cliques.* A *γ-clique* C_γ, also called a *quasi-clique,* is a subset of *V* such that $G(C_\gamma)$ has at least $\lfloor \gamma q(q-1)/2 \rfloor$ edges, where q is the cardinality (i.e., number of vertices) of C_γ.

An *independent set* is a subset $I \subseteq V$ such that the subgraph *G*(*I*) has no edges. The maximum independent set problem can be easily reformulated as the maximum clique problem in the *complementary* graph $\bar{G}(V, \bar{E})$, which is defined as follows. If an edge $(i,j) \in E$, then $(i,j) \notin \bar{E}$, and if $(i,j) \notin E$, then $(i,j) \in \bar{E}$. Clearly, a maximum clique in $\bar{G}$ is a maximum independent set in *G,* so the maximum clique and maximum independent set problems can be easily reduced to each other.

Locating cliques (quasi-cliques) and independent sets in a graph representing a dataset provides important information about this dataset. Intuitively, edges in such graph would connect vertices corresponding to "similar" elements of the dataset, therefore, cliques (or quasi-cliques) would naturally represent dense clusters of similar objects. On the contrary, independent sets can be treated as groups of objects that differ from every other object in the group, and this information is also important in some applications. Clearly, it is useful to find a maximum clique or independent set in the graph, since it would give the maximum possible size of the groups of "similar" or "different" objects.

There are many mathematical programming algorithms developed for lo-

cating cliques and independent sets in a graph. One of the most well-known approaches to finding large cliques and independent sets in a graph is known as Greedy Randomized Adaptive Search Procedure (GRASP) [30, 31]. In short, GRASP is an iterative method that at each iteration constructs, using a greedy function, a randomized solution and then finds a locally optimal solution by searching the neighborhood of the constructed solution. This is a *heuristic* algorithm which gives no guarantee about quality of the solutions found, but proved to be practically efficient for many combinatorial optimization problems. However, it is not easy to find the maximum clique (or independent set) *exactly,* since the maximum clique problem (as well as the maximum independent set problem) is known to be NP-hard [33]. Moreover, it turns out that even the approximation of the maximum clique is NP-hard [9, 35]. This makes these problems especially challenging in large graphs. However, as we will see later, sometimes it is possible to use a special structure of the graph to get the exact solution of the maximum clique problem.

4.1.3 Clique Partitioning and Coloring

The problem of locating cliques and independent sets in a graph can be naturally extended to finding an optimal *partition* of a graph into a minimum number of distinct cliques or independent sets. These problems are referred to as *minimum clique partitioning* and *coloring,* respectively. Various mathematical programming formulations of these problems can be found in [57]. Clearly, as in the case of maximum clique and maximum independent set problems, minimum clique partitioning and coloring are reduced to each other by considering the complimentary graph. Solving these problems for the graphs representing real-life datasets is important from the data mining perspective. As it was pointed out above, if a dataset is represented as a graph, where each data element corresponds to a vertex, the clustering problem essentially deals with decomposing this graph into a set of subgraphs (subsets of vertices), so that each of these subgraphs would correspond to a specific cluster.

Obviously, since the data elements assigned to the same cluster should be "similar" to each other, the goal of clustering can be achieved by solving the minimum clique partitioning problem. Although solving this problem for large graphs is not an easy task (minimum clique partitioning and coloring problems are NP-hard), this approach has an important positive aspect. Recall that standard clustering algorithms considered above are designed for the case of a *fixed* number of clusters. However, the minimum clique

partition of the graph would automatically determine the number of clusters in the considered dataset, which is very useful when the number of clusters is not known a-priori.

Similar arguments hold for the case of the graph coloring problem which should be solved when a dataset needs to be decomposed into the clusters of "different" objects (i.e., each object in a cluster is different from all other objects in the same cluster), that can be represented as independent sets in the corresponding graph. The number of independent sets in the optimal partition is referred to as the *chromatic number* of the graph.

It should be noted that instead of cliques and independent sets one can consider *quasi-cliques* and *quasi-independent sets* and partition the graph on this basis. As it was described above, quasi-cliques are subgraphs that are *dense enough* (i.e., they have a high edge density). Therefore, it is often reasonable to relate clusters to quasi-cliques, since they would represent sufficiently dense clusters of similar objects. Obviously, in the case of partitioning a dataset into clusters of "different" objects, one can use quasi-independent sets (i.e., subgraphs that are sparse enough) to define these clusters.

Next, we present several examples of representing real-life datasets as graphs and extracting non-trivial information from them using graph-based techniques.

4.2 Application: Call Graph

One of the examples of applying graph-theoretical techniques to analyzing massive datasets is the *Call graph* representing telecommunications traffic data, which was studied by Abello, et al. [1] and Aiello et al. [6]. In this graph, the vertices are telephone numbers, and two vertices are connected by an edge if a call was made from one number to another.

The considered one-day call graph representing the data from AT&T telephone billing records had 53,767,087 vertices and over 170 millions of edges. This graph appeared to have 3,667,448 connected components, most of them tiny; only 302,468 (or 8%) components had more than 3 vertices. A giant connected component with 44,989,297 vertices was computed [1].

The maximum clique problem and problem of finding large quasi-cliques with pre-specified density were considered in this giant component. These problems were addressed using a greedy randomized adaptive search procedure (GRASP) [30, 31]. To make application of optimization algorithms in the considered large component possible, the authors use some suitable graph decomposition techniques employing external memory algorithms [3].

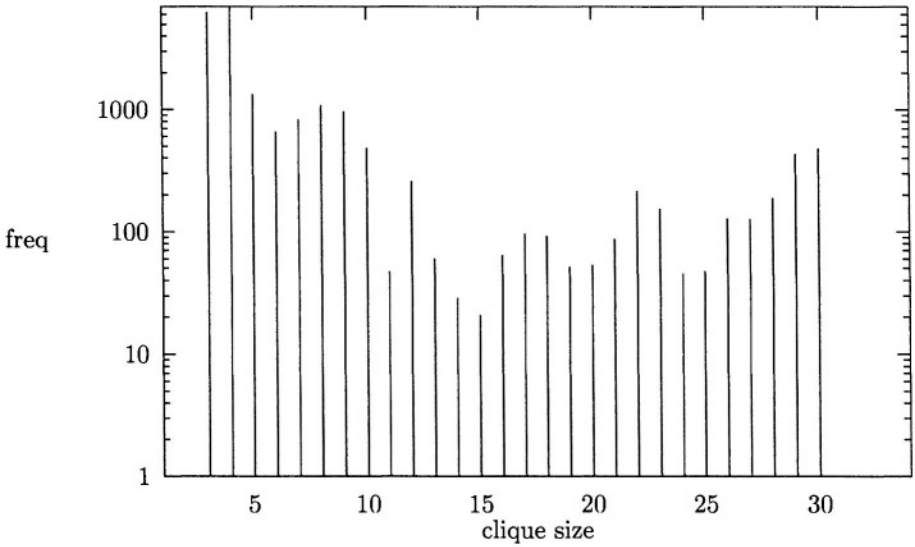


Figure 3: Frequencies of clique sizes found by Abello et al. [1] in the Call graph.

Abello et al. ran 100,000 GRASP iterations taking 10 parallel processors about one and a half days to finish. Of the 100,000 cliques generated, 14,141 appeared to be distinct, although many of them had vertices in common. The authors suggested that the graph contains no clique of a size greater than 32. Figure 3 shows the number of detected cliques of various sizes. Finally, large quasi-cliques with density parameters $\gamma = 0.9, 0.8, 0.7$ and 0.5 for the giant connected component were computed. The sizes of the largest quasi-cliques found were 44, 57, 65 and 98, respectively.

It is also important to investigate the degree distribution of the Call graph. According to [6], the distribution of in-degrees and out-degrees of this graph, as well as the distribution of the sizes of the connected components, can be very well represented by a power law. The corresponding plots are presented in Figure 4.

Summarizing the results presented in this subsection, one can say that graph-based techniques proved to be rather useful in the analysis and revealing the global patterns of the telecommunications traffic dataset. In the next subsection, we will consider another example of a similar type of dataset associated with the World-Wide Web.

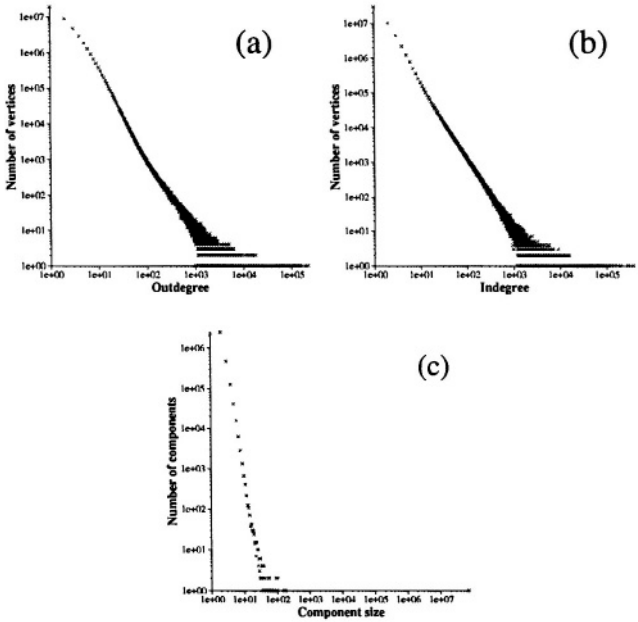


Figure 4: Number of vertices with various out-degrees (a) and in-degrees (b); the number of connected components of various sizes (c) in the Call graph, due to [6].

4.3 Application: Internet and Web Graphs

Probably the largest massive graph considered in the literature is the *Web graph* representing the World Wide Web. The number of web pages indexed by large search engines exceeds 2 billion, and the number of web sites is increasing every day.

In this graph, the vertices are documents and the edges are hyperlinks pointing from one document to another. Similarly to the Call graph, the Web is a directed multigraph, although often it is treated as an undirected graph to simplify the analysis. Another graph is associated with the physical network of the Internet, where the vertices are *routers* navigating packets of data or groups of routers (domains). The edges in this graph represent wires or cables in the physical network.

Graph theory has been applied for web search [44, 21, 27], web mining [52, 53] and other problems arising in the Internet and World Wide Web. In several recent studies there were attempts to understand some

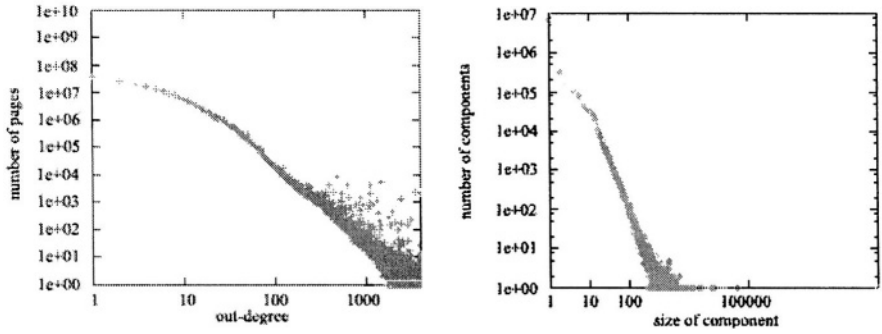


Figure 5: Number of vertices with various out-degrees (left) and distribution of sizes of strongly connected components (right) in a Web graph [22].

structural properties of the Web graph by investigating large Web crawls. Adamic and Huberman [39, 4] used crawls which covered almost 260,000 pages in their studies. Barabási and Albert [10] analyzed a subgraph of the Web graph with approximately 325,000 nodes representing nd.edu pages. In another experiment, Kumar et al. [47] examined a data set containing about 40 million pages. In a recent study, Broder et al. [22] used two Altavista crawls, each with about 200 million pages and 1.5 billion links, thus significantly exceeding the scale of the preceding experiments. This work yielded several remarkable observations about local and global properties of the Web graph. All of the properties observed in one of the two crawls were validated for the other as well. Below in this section by the Web graph we will mean one of the crawls, which has 203,549,046 nodes and 2130 million arcs.

The first observation made by Broder et al. confirms a property of the Web graph suggested in earlier works [10, 47], claiming that the distribution of degrees follows a *power law*.

Interestingly, the degree distribution of the Web graph resembles the power-law relationship of the Internet graph topology, which was first discovered by Faloutsos et al. [29]. The authors of [22] computed the in- and out-degree distributions for both considered crawls and showed that these distributions follow the power law. Moreover, they observed that in the case of in-degrees the constant $\gamma \approx 2.1$ is the same as the exponent of power laws discovered in earlier studies [47, 10]. In another set of experiments conducted by Broder et al., directed and undirected connected components were investigated. It was noticed that the distribution of sizes of these connected

components also obeys a power law. Figure 5 illustrates the experiments with distributions of out-degrees and connected component sizes.

The last series of experiments discussed in [22] aimed to explore the global connectivity structure of the Web. This led to the discovery of the so-called *Bow-Tie* model of the Web [23]. Similarly to the Call graph, the considered Web graph appeared to have a giant connected component, containing 186,771,290 nodes, or over 90% of the total number of nodes. Taking into account the directed nature of the edges, this connected component can be further partitioned into four large classes: *strongly connected component (SCC), In* and *Out* components and *"Tendrils"*. The characteristics of each of these groups are summarized below:

- The **strongly connected component** is the part of the giant connected component in which all nodes are reachable from one another by a directed path.
- The **In component** consists of nodes which can reach any node in the SCC but cannot be reached from the SCC.
- The **Out component** contains the nodes that are reachable from the SCC, but cannot access the SCC through directed links.
- The **Tendrils component** accumulates the remaining nodes of the giant connected component, i.e., the nodes which are not connected with the SCC.
- The **Disconnected component** is the part of the Web which is not connected with the giant connected component.

Figure 6 shows the connectivity structure of the Web, as well as sizes of the considered components. As one can see from the figure, the sizes of SCC, In, Out and Tendrils components are roughly equal, and the Disconnected component is significantly smaller.

4.4 Application: Market Graph

In this subsection, we will show how the network optimization approaches can be applied to extracting information from the stock prices data.

Finding efficient ways of summarizing and visualizing the stock market data, which would allow one to obtain useful information about the behavior of the market, is one of the most important problems in the modern finance. Nowadays, a great number of stocks are traded in the US stock market;

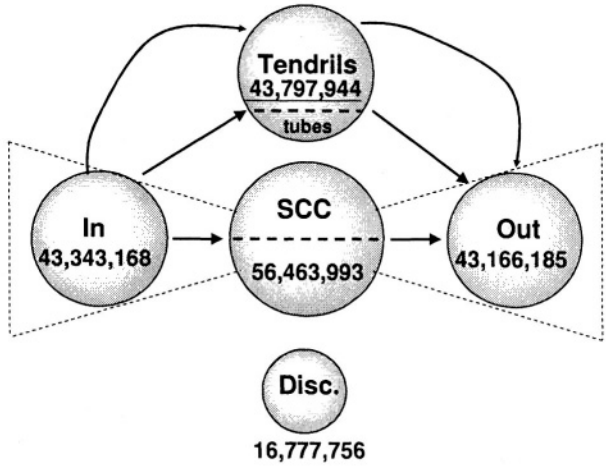


Figure 6: Connectivity of the Web due to [22] (Bow-Tie model).

moreover, this number significantly increases each month. The amount of data generated by the stock market every day is huge. This data is usually visualized by thousands of plots reflecting the price of each stock over a certain period of time. The analysis of these plots becomes more and more complicated as the number of stocks grows.

An alternative way of summarizing the stock prices data, which was recently developed, is based on representing the stock market as a graph.

This graph is referred to as the *market graph.* Clearly, the set of vertices in this graph should represent the set of considered financial instruments. Also, as it was pointed out in Section 2, it is possible to define a distance measure for every pair of data elements, so that the value of this distance would quantify the "similarity" or "dissimilarity" of every pair of elements. A natural way for introducing this type of "similarity measure" for financial instruments is calculating the cross-correlations of price fluctuations for every pair of stocks over a certain period of time. Let $P_i(t)$ denote the price of the instrument i on day t. Then $R_i(t) = \ln \frac{P_i(t)}{P_i(t-1)}$ defines the logarithm of return of the instrument i over the one-day period from $(t-1)$ to t. The correlation coefficient between instruments i and j is calculated as

$$C_{ij} = \frac{E(R_i R_j) - E(R_i)E(R_j)}{\sqrt{Var(R_i)Var(R_j)}},$$

where $E(R_i)$ is defined simply as the average return of the instrument i over N considered days (i.e., $E(R_i) = \frac{1}{N}\sum_{t=1}^{N} R_i(t)$) [50].

One of the possible ways of reflecting the connections between different stocks is to construct a *weighted graph,* which would have all possible edges,[3] and each edge would be assigned a certain value (or, *weight*) representing the cross-correlation between the corresponding pair of stocks. However, the analysis of a weighted graph may be too hard if the number of vertices is large enough. Alternatively, one can construct an unweighted graph (without assigning a weight to every edge) similarly to the examples described above. In order to do this, a *threshold* θ ($\theta \in [-1, 1]$) is introduced, and an edge connecting stocks i and j is added to the graph if $C_{ij} \geq \theta$. In these settings, if two vertices are connected by an edge, it means that the prices of these two stocks behave similarly over time. Moreover, changing the value of the threshold θ allows one to control the measure of this similarity and construct different instances of the market graph for different correlation thresholds.

Clearly, studying the pattern of connections in the market graph would provide helpful information about the internal structure of the stock market.

Boginski et al. [17] analyzed daily fluctuations of the prices of over 6000 stocks traded in the US stock market during 500 consecutive trading days in 2000-2002. Several important results were obtained from this study.

It turns out that the degree distribution of the market graph for $\theta \geq 0.2$ is stable and follows the power-law model (which was also the case for the Internet graph, Web graph and Call graph). Figure 7 shows the degree distribution of some instances of the market graph.

Moreover, the authors investigated cliques and independent sets in the market graph. Cliques and independent sets in the market graph have a simple practical interpretation. A clique in the market graph with a positive value of the correlation threshold θ is a set of instruments whose price fluctuations exhibit a similar behavior (a change of the price of any instrument in a clique is likely to affect all other instruments in this clique), and an independent set in the market graph with a negative value of θ consists of instruments that are negatively correlated with respect to each other, therefore, it represents a "completely diversified" portfolio. The special subject of interest was finding the maximum clique and maximum independent set in the market graph, which would provide interesting information about the internal structure of the market by answering the question, what is the maximum possible group of stocks that behave similarly to each other, and what

[3]A graph with all possible edges is referred to as *complete graph.*

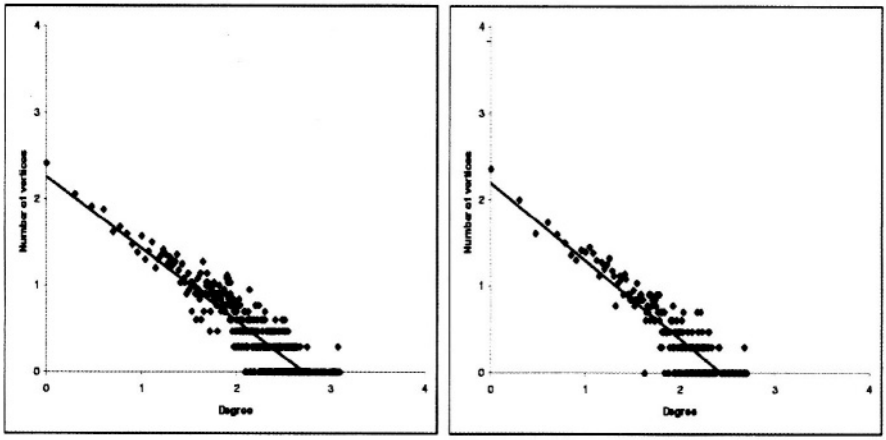


Figure 7: Degree distribution of the market graph for $\theta = 0.4$ (left); $\theta = 0.5$ (right) *(logarithmic scale)* due to [17].

is the maximum possible diversified portfolio?

It turned out that even though the size of this graph was rather large, it is relatively easy to find the exact solution of the maximum clique problem using mathematical programming techniques.

The maximum clique problem admits an integer programming formulation, however, in the case of the graph with over 6000 vertices this integer programming problem cannot be solved in a reasonable time. Therefore, a simple graph algorithm (greedy heuristic) was applied for finding a large clique (and a lower bound of the maximum clique size).

Let $N(i) = \{j | (i, j) \in E\}$ denote the set of neighbors of vertex i in the graph $G = (V, E)$. The algorithm for finding a large clique looks as follows:

$C = \emptyset, G_0 = G;$

do

$G_0 = \bigcap_{i \in C} N(i) \setminus C;$

$C = C \bigcup j$, where j is a vertex of largest degree in G_0;

until $G_0 = \emptyset$.

After running this algorithm, the following preprocessing procedure was applied. Let C be the clique found by the above algorithm, then all the

Table 1: Sizes of the maximum cliques in the market graph with different values of the correlation threshold [17].

θ	edge density	clique size
0.35	0.0090	193
0.4	0.0047	144
0.45	0.0024	109
0.5	0.0013	85
0.55	0.0007	63
0.6	0.0004	45
0.65	0.0002	27
0.7	0.0001	22

vertices which are not in C and whose degree is less than $|C|$ are recursively removed from the graph. It turns out that in the case of the market graph this simple procedure makes it possible to significantly reduce the size of the maximum clique search space, which can be explained by the fact that the market graph is *clustered* (i.e., two vertices are more likely to be connected if they have a common neighbor).

Let $G'(V', E')$ denote the graph induced by the vertices remaining after applying this procedure. In order to find the maximum clique of G' (which is also the maximum clique in the original graph G), the following integer programming formulation of the maximum clique problem was used:

$$\text{maximize} \quad \sum x_i$$

$$s.t.$$

$$x_i + x_j \leq 1, (i,j) \notin E'$$

$$x_i \in \{0,1\}$$

Since the preprocessing procedure decreased the number of vertices in the initial graph (and the number of variables in the problem), this integer programming problem could be solved using a standard software package.

Table 1 summarizes the sizes of the maximum cliques found in the graph for different values of θ, and it turns out that these cliques are rather large.

The authors also considered independent sets in the market graphs with negative (and small positive) values of the correlation threshold θ. As it was indicated above, such independent sets represent "completely diversified"

Table 2: Sizes of independent sets in the market graph found using the greedy algorithm [17]

θ	edge density	indep. set size
0.05	0.4794	36
0.0	0.2001	12
-0.05	0.0431	5
-0.1	0.005	3
-0.15	0.0005	2

portfolios, where all pairs of stocks are negatively correlated. As described in the beginning of this section, the maximum independent set problem can be easily represented as a maximum clique problem in a complementary graph. In the case of the market graph, the complementary graph is constructed as follows: an edge connects two stocks if the correlation between them $C_{ij} \leq \theta$.

However, the approach that was successfully applied for finding a maximum clique was not so efficient for complementary graphs, which means that large independent sets, i.e., diversified portfolios, cannot be easily found in the graph representing the modern stock market. Table 2 summarizes the sizes of the largest independent sets found by using the heuristic algorithm described above.

Large cliques and small independent sets in the market graph confirm the widely discussed idea about the globalization of the economy.

Another issue addressed in the study of the market graph was finding a completely diversified portfolio containing every given stock. Interestingly, for every vertex in the market graph, an independent set that contains this vertex was detected, and the sizes of these independent sets were almost the same. Moreover, all these independent sets were distinct. These facts demonstrate that it is always possible for an investor to find a group of stocks that would form a completely diversified portfolio with any given stock, and this can be efficiently done using the technique of finding independent sets in the market graph.

The methodology of finding cliques and independent sets in the market graph provides an efficient tool of performing clustering based on the stock market data. The choice of the grouping criterion is natural: "similar" or "different" financial instruments are determined according to the correlation between their price fluctuations. Clearly, the minimum number of clusters in the partition of the set of financial instruments is equal to the minimum

number of distinct cliques that the market graph can be divided into (the minimum clique partition problem). If independent sets are used instead of cliques, it would represent the partition of the market into a set of *distinct diversified portfolios.* In this case the minimum possible number of clusters is equal to a minimum number of distinct independent sets, or the *chromatic number* corresponding to the optimal solution of the graph coloring problem.

It should be noted that the approaches to classifying financial instruments commonly considered in the literature usually deal with "training datasets" and use mathematical programming techniques designed for predictive modeling (see, for example, [25]). On the contrary, the market graph model does not require any a-priori information about the classes that certain stocks belong to, but classifies them only based on the behavior of their prices over time, which makes it more efficient in practice, since the construction of a training dataset for classification of stocks may be a non-trivial task.

4.5 Application: Modeling Epileptic Human Brain

Another application of network approaches to data analysis is associated with biomedicine. One of the most important biomedical applications is studying human brain. The enormous number of neurons and dynamic nature of connections between them makes the analysis of brain function especially challenging. Analyzing the connectivity of the brain using graph-theoretical approaches is an important practical task, and the results of this analysis can be applied in treatment of various types of brain disorders, for instance, epileptic seizures. Clearly, the analysis of the graph representing all the neurons as vertices cannot be empirically performed, since the number of neurons in the brain and the connections between them is too large: according to [54], the number of neurons is estimated to be 8.3×10^9, and the number of connections is approximately 6.6×10^{13}. However, one can still judge about some properties of this graph, for instance, the fact that despite its low edge density, this graph is clustered (i.e., it contains dense groups of interconnected vertices), which is also typical for other real-life graphs. Various aspects of the analysis of brain connectivity are discussed in [38].

In order to perform a more detailed quantitative analysis of the graph representing the brain, one can consider much smaller graphs by treating certain groups of neurons (functional units of the brain) as vertices and investigate the connections between these functional units. For instance, this approach was applied to the connectivity analysis of the cortical visual

system of the macaque monkey [32]. The connectivity matrix corresponding to this study is presented in [38]. It should be noted that the connections between different functional units of the brain are *directed,* which reflects the fact that there are only feedforward or feedback connections between certain brain units.

During the last several years, the advances in studying the brain are associated with the extensive use of *electroencephalograms (EEG)* which can be treated as the quantitative representation of the brain function. As we will see below, it is possible to apply graph-theoretical approaches to the analysis of the brain using EEG data.

EEG data essentially represents time series recorded from the electrodes located in different functional units of the brain. The analysis of these time series is not a trivial task, and in order to reveal some patterns underlying the EEG data, a certain statistical preprocessing procedure needs to be applied. The main idea of this technique is calculating the values of T-index corresponding to all possible pairs of functional units of the brain. T-index is a standard statistical concept which can be used as a measure of entrainment of two brain sites at a certain time moment. More specifically, two functional units of the brain are considered to be entrained (i.e., they exhibit a similar behavior) if the corresponding value of the T-index calculated for this pair of electrodes does not exceed a certain threshold referred to as $T_{critical}$. This procedure is described in more detail in [40].

The reason for analyzing the entrainment of different brain sites is the possibility to apply these results to studying brain disorders, for instance, epilepsy. At the moment of an epileptic seizure, some brain sites exhibit the convergence of their EEG signals, which is characterized by the drop of the corresponding T-index below $T_{critical}$ (these brain sites are referred to as "critical sites"). Due to this fact, a natural graph structure representing the epileptic brain can be constructed as follows. The vertices of the graph represent the considered functional units of the brain, and two vertices i and j are connected by an edge if the value of T-index T_{ij} corresponding to these two functional units is less than $T_{critical}$. Clearly, this graph reflects the connections only between those brain sites that are entrained at a certain time moment, therefore, one can investigate the evolution of the properties of this graph over time, and especially at the moments of epileptic seizures.

Next, we discuss the properties of this graph that can be considered, as well as their practical interpretation. In [61], the authors constructed and analyzed the graph containing 28 vertices corresponding to the electrodes located in different parts of the brain. Various characteristics of this graph, such as its connectivity, edge density, and degrees of the vertices, were an-

alyzed, and several interesting results were obtained. It turns out that the number of edges in this graph dramatically increases at seizure points, and it decreases immediately after seizures. It means that the global structure of the graph significantly changes during the seizure and after the seizure, i.e., the density of connections increases during ictal state (i.e., at the point of the seizure) and decreases in postictal state (after the seizure). Moreover, the connectivity of this graph is affected by epileptic seizures: the graph is connected during the period between the seizures (i.e., the brain is a connected system), however, it becomes disconnected after the seizure (during the postictal state): the size of the largest connected component significantly decreases. This fact can be intuitively explained, since after the seizure the brain needs some time to "reset" and restore the connections between the functional units. Another non-trivial result obtained in this analysis is the fact that the vertex with the maximum degree in this graph is usually located near the epileptogenic focus.

One more aspect in the analysis of the graph model of the epileptic brain is finding a maximum clique in this graph. To explain the practical interpretation of a maximum clique in this case, we need to briefly describe a related mathematical programming technique applied in the field of epileptic seizures prediction. The main idea of this approach is to construct a quadratic programming model that would select a certain number of "critical" electrode sites, i.e., those that are the most entrained during the seizure. Clearly, such group of electrode sites should produce a minimal sum of T-indices calculated for all pairs of electrodes within this group. If the number of critical sites is set equal to k, and the total number of electrode sites is n, then the problem of selecting the optimal group of critical sites can be formulated as the following quadratic 0-1 problem:

$$\min \quad x^T A x \tag{13}$$

$$\text{s.t.} \quad \sum_{i=1}^{n} x_i = k. \tag{14}$$

$$x_i \in \{0,1\} \quad \forall i \in \{1,\ldots,n\}, \tag{15}$$

where $x = (x_1, x_2, \ldots, x_n)$ is a 0-1 vector, where $x_i = 1$ if the i-th site is included into the set of critical sites, and $x_i = 1$, otherwise. The elements of the matrix $A = [a_{ij}]_{i,j=1,\ldots,n}$ are the values of T_{ij}'s at the seizure point.

We should also mention that, as it was shown in the previous work, one can observe the "resetting" of the brain, that is, the statistical "divergence" of EEG profiles after a seizure [63, 65]. Therefore, to ensure that the optimal group of critical sites shows this divergence, one can reformulate this

optimization problem by adding one more quadratic constraint:

$$x^T Bx \geq T_{critical}\, k\,(k-1), \tag{16}$$

where the matrix $B = [b_{ij}]_{i,j=1,\ldots,n}$ is the T-index matrix of brain sites i and j within 10 minute windows after the onset of a seizure.

It should be noted that this quadratic programming technique was successfully applied in the analysis of the predictability of epileptic seizures [40, 58], and, as we will show below, there is a strong connection between the graph-theoretical and quadratic programming approaches to the analysis of the brain.

The graph model of the epileptic brain described above can be easily adjusted using the same principles as were utilized in the formulation of this quadratic problem, that is, one can consider only the connections between the pairs of sites i, j satisfying both of the two conditions: $T_{ij} < T_{critical}$ at the seizure point, and $T_{ij} > T_{critical}$ 10 minutes after the seizure point, which are exactly the conditions that the critical sites must satisfy. A natural way of detecting such a group of sites is to find *cliques* in this modified graph. Since a clique is a subgraph where all vertices are interconnected, all pairs of electrode sites in a clique would satisfy the above conditions. Therefore, it is clear that the size of the *maximum clique* in this graph would represent the *upper bound* on the number of selected critical sites, i.e., the maximum value of the parameter k in the aforementioned quadratic programming problem. Computational results show that this upper bound is rather tight, i.e., the maximum clique sizes calculated for the graphs representing different patients are rather close to the actual values of k empirically selected in the quadratic programming model.

5 Concluding Remarks

In this chapter, we have addressed several issues regarding the use of network-based mathematical programming techniques for solving various problems arising in the broad area of data mining. We have pointed out that applying these approaches often proved to be effective in many applications, including biomedicine, finance, telecommunications, etc. In particular, if a real-world massive dataset can be appropriately represented as a network structure, its analysis using standard graph-theoretical techniques often yields important practical results.

However, one should clearly understand that the success or failure of applying a certain methodology essentially depends on the structure of the considered dataset, and there is no "universal recipe" that would allow one to obtain useful information from any type of data. This indicates that despite the availability of a great variety of data mining techniques and software packages, choosing an appropriate method of the analysis of a certain dataset is a non-trivial task.

Moreover, as technological progress continues, new types of datasets may emerge in different practical fields, which would lead to further research in the field of data mining algorithms. Therefore, developing and modifying mathematical programming approaches in data mining is an exciting and challenging research area for years to come.

References

[1] J. Abello, P.M. Pardalos, and M.G.C. Resende, 1999. On maximum clique problems in very large graphs, *DIMACS Series,* 50, American Mathematical Society, 119-130.

[2] J. Abello, P.M. Pardalos, and M.G.C. Resende (eds.), 2002. Handbook of Massive Data Sets, Kluwer Academic Publishers.

[3] J. Abello and J. S. Vitter (eds.). *External Memory Algorithms.* Vol. 50 of DIMACS Series in Discrete Mathematics and Theoretical Computer Science. American Mathematical Society, 1999.

[4] L. Adamic and B. Huberman. Power-law distribution of the World Wide Web. *Science,* 287: 2115a, 2000.

[5] R. Agrawal, J. Gehrke, D. Gunopulos, and P. Raghvan, 1998. Automatic Subspace Clustering of High Dimensional Data for Data Mining Applications, in Proceedings of ACM SIGMOD International Conference on Management of Data, ACM, New York, 94105.

[6] W. Aiello, F. Chung, and L. Lu, 2001. A random graph model for power law graphs, *Experimental Math.* **10**, 53-66.

[7] R. Albert and A.-L. Barabasi, 2002. Statistical mechanics of complex networks, Reviews of Modern Physics 74, 47-97.

[8] M.R. Anderberg, 1973. *Cluster Analysis for Applications,* Academic Press, New York.

[9] S. Arora and S. Safra, 1992. Approximating clique is NP-complete, *Proceedings of the 33rd IEEE Symposium on Foundations on Computer Science,* 2-13.

[10] A.-L. Barabasi and R. Albert, 1999. Emergence of scaling in random networks. *Science* **286**: 509–511.

[11] A.-L. Barabasi, , 2002. *Linked,* Perseus Publishing.

[12] K.P. Bennett and O.L. Mangasarian, 1992. Neural Network Training via Linear Programming, in Advances in Optimization and Parallel Computing, P.M. Pardalos, (ed.), North Holland, Amsterdam, 5667.

[13] C. Berge, 1976. *Graphs and Hypergraphs.* North-Holland Mathematical Library, 6.

[14] P. Berkhin, 2002. Survey of Clustering Data Mining Techniques. Technical Report, Accrue Software, San Jose, CA.

[15] D. Bertsimas and R. Shioda, 2002. Classification and Regression via Integer Optimization. http://pages.stern.nyu.edu/ rcaldent/seminar02/Romy.pdf

[16] V. Boginski, S. Butenko, and P.M. Pardalos, 2003. Modeling and Optimization in Massive Graphs. In: P. M. Pardalos and H. Wolkowicz, editors. Novel Approaches to Hard Discrete Optimization, American Mathematical Society, 17–39.

[17] V. Boginski, S. Butenko, and P.M. Pardalos, 2003. On Structural Properties of the Market Graph. In: A. Nagurney (editor), Innovations in Financial and Economic Networks, Edward Elgar Publishers, 28–45.

[18] I.M. Bomze, M. Budinich, P.M. Pardalos, and M. Pelillo, 1999. The maximum clique problem. In: D.-Z. Du and P.M. Pardalos, editors, Handbook of Combinatorial Optimization, Kluwer Academic Publishers, 1-74.

[19] P.S. Bradley, U.M. Fayyad, and O.L. Mangasarian, 1999. Mathematical Programming for Data Mining: Formulations and Challenges. INFORMS Journal on Computing, 11(3), 217–238.

[20] P.S. Bradley, O.L. Mangasarian, and W.N. Street, 1998. Feature Selection via Mathematical Programming, INFORMS Journal on Computing 10, 209217.

[21] S. Brin and L. Page, 1998. The anatomy of a large scale hypertextual web search engine. *Proc. 7th WWW.*

[22] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tomkins, and J. Wiener, 2000. Graph structure in the Web. *Computer Networks,* 33: 309–320.

[23] A. Broder, R. Kumar, F. Maghoul, P. Raghavan, S. Rajagopalan, R. Stata, A. Tompkins, and J. Wiener, 2000. The Bow-Tie Web. *Proceedings of the 9th International World Wide Web Conference.*

[24] V. Bugera, H. Konno, and S. Uryasev, 2002. Credit Cards Scoring with Quadratic Utility Functions, Journal of Multi-Criteria Decision Analysis, 11(4-5), 197-211.

[25] V. Bugera, S. Uryasev, and G. Zrazhevsky, 2003. Classification Using Optimization: Application to Credit Ratings of Bonds. Univ. of Florida, ISE Dept., Research Report #2003-14.

[26] C.J.C. Burges, 1998. A Tutorial on Support Vector Machines for Pattern Recognition, Data Mining and Knowledge Discovery 2, 121167.

[27] S. Chakrabarti, B. Dom, D. Gibson, S. R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Experiments in topic distillation. *Proc. ACM SIGIR workshop on Hypertext Information Retrieval on the Web,* 1998.

[28] M. Craven and J. Shavlik, 1997. Using Neural Networks for Data Mining. Future Generation Computer Systems (Special Issue on Data Mining) 13, 211-229.

[29] M. Faloutsos, P. Faloutsos and C. Faloutsos. On power-law relationships of the Internet topology. *ACM SICOMM,* 1999.

[30] T. A. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization,* 6:109-133, 1995.

[31] T. A. Feo and M. G. C. Resende. A greedy randomized adaptive search procedure for maximum independent set. *Operations Research,* 42:860-878, 1994.

[32] D.J. Felleman and D.C. Van Essen, 1991. Distributed Hierarchical Processing in the Primate Cerebral Cortex. Cereb. Cortex, 1, 1–47.

[33] M.R. Garey and D.S. Johnson, 1979. Computers and Intractability: A Guide to the Theory of NP-completeness, Freeman.

[34] M.H. Hassoun, 1995. Fundamentals of Artificial Neural Networks, MIT Press, Cambridge, MA.

[35] J. Håstad, 1999. Clique is hard to approximate within $n^{1-\epsilon}$, *Acta Mathematica* **182** 105-142.

[36] B. Hayes, 2000. Graph Theory in Practice. *American Scientist,* **88**: 9-13 (Part I), 104-109 (Part II).

[37] S. Haykin, 1999. *Neural Networks: A Comprehensive Foundation.* Macmillan College Publishing Company, New York.

[38] C.C. Hilgetag, R. Kötter, K.E. Stephen, O. Sporns, 2002. Computational Methods for the Analysis of Brain Connectivity, In: G. A. Ascoli, ed., Computational Neuroanatomy, Humana Press.

[39] B. Huberman and L. Adamic. Growth dynamics of the World-Wide Web. *Nature,* 401: 131, 1999.

[40] L.D. Iasemidis, P.M. Pardalos, J.C. Sackellares, D-S. Shiau, 2001. Quadratic binary programming and dynamical system approach to determine the predictability of epileptic seizures. J. Combinatorial Optimization 5, 9-26

[41] A.K. Jain and R.C. Dubes, 1988. Algorithms for Clustering Data, Prentice-Hall, Englewood Cliffs, NJ.

[42] H. Jeong, B. Tomber, R. Albert, Z.N. Oltvai, and A.-L. Barabasi, 2000. The large-scale organization of metabolic networks, *Nature* **407**: 651-654.

[43] D. S. Johnson and M. A. Trick (eds.), 1996. Cliques, Coloring, and Satisfiability: Second DIMACS Implementation Challenge, Vol. 26 of DIMACS Series, American Mathematical Society.

[44] J. Kleinberg. Authoritative sources in a hyperlinked environment. *Proc. 9th ACM-SIAM SODA,* 1998.

[45] J. Kleinberg and S. Lawrence. The Structure of the Web. *Science,* 294: 1849-50, 2001.

[46] R. Kumar, P. Raghavan, S. Rajagopalan, D. Sivakumar, A. Tomkins, and E. Upfal. The Web as a graph. *Symposium on Principles of Database Systems,* pages 1-10, 2000.

[47] R. Kumar, P. Raghavan, S. Rajagopalan, and A. Tomkins. Trawling the Web for cyber communities. *Proc. 8th WWW,* 1999.

[48] O.L. Mangasarian, 1993. Mathematical Programming in Neural Networks, ORSA Journal on Computing 5, 349-360.

[49] O.L. Mangasarian, W.N. Street, and W.H. Wolberg, 1995. Breast Cancer Diagnosis and Prognosis via Linear Programming, Operations Research 43(4), 570-577.

[50] R. N. Mantegna, and H. E. Stanley, 2000. An Introduction to Econophysics: Correlations and Complexity in Finance, Cambridge University Press.

[51] B. Mirkin and I. Muchnik, 1998. Combinatoral Optimization in Clustering. In: Handbook of Combinatorial Optimization (D.-Z. Du and P.M. Pardalos, eds.), Volume 2. Kluwer Academic Publishers, 261-329.

[52] A. Mendelzon, G. Mihaila, and T. Milo. Querying the World Wide Web. *Journal of Digital Libraries,* 1: 68-88, 1997.

[53] A. Mendelzon and P. Wood. Finding regular simple paths in graph databases. *SIAM J. Comp.,* 24: 1235-1258, 1995.

[54] J.M. Murre and D.P. Sturdy, 1995. The Connectivity of the Brain: Multi-Level Quantitative Analysis. Biol. Cybern., 73, 529–545.

[55] E. Osuna, R. Freund, and F. Girosi, 1997. Improved Training Algorithm for Support Vector Machines, in Proceedings of IEEE NNSP97, Amelia Island, FL, September 1997, IEEE Press, New York, 276285.

[56] E. Osuna, R. Freund, and F. Girosi, 1997. Training Support Vector Machines: An Application to Face Detection, in IEEE Conference on Computer Vision and Pattern Recognition, Puerto Rico, June 1997, IEEE Press, New York, 130136.

[57] P.M. Pardalos, T. Mavridou, and J. Xue, 1998. The Graph Coloring Problem: A Bibliographic Survey. In: Handbook of Combinatorial Optimization (D.-Z. Du and P.M. Pardalos, eds.), Volume 2. Kluwer Academic Publishers, 331–395.

[58] P.M. Pardalos, W. Chaovalitwongse, L.D. Iasemidis, J.C. Sackellares, D.-S. Shiau, P.R. Carney, O.A. Prokopyev, V.A. Yatsenko, 2003. Seizure Warning Algorithm Based on Spatiotemporal Dynamics of Intracranial EEG, Submitted to Mathematical Programming.

[59] N.G. Pavlidis, D.K. Tasoulis, G.S. Androulakis, and M.N. Vrahatis, 2004. Exchange Rate Forecasting through Distributed Time-Lagged Feedforward Neural Networks. In: Supply Chain and Finance (P.M. Pardalos, A. Migdalas, G. Baourakis, eds.), World Scientific, 283–298.

[60] G. Piatetsky-Shapiro and W. Frawley (eds.), 1991. Knowledge Discovery in Databases, MIT Press, Cambridge, MA.

[61] O.A. Prokopyev, V. Boginski, W. Chaovalitwongse, P. M. Pardalos, J. C. Sackellares, and P. R. Carney, 2003. Network-Based Techniques in EEG Data Analysis and Epileptic Brain Modeling. Submitted to Computational Statistics and Data Analysis.

[62] D.E. Rumelhart and D. Zipser, 1985. Feature Discovery by Competitive Learning. Cognitive Science, 9, 75–112.

[63] J.C. Sackellares, L.D. Iasemidis, R.L. Gilmore, S.N. Roper, 1997. Epileptic seizures as neural resetting mechanisms. Epilepsia 38, S3, 189.

[64] B. Scholkopf, C. Burges, and V. Vapnik, 1995. Extracting Support Data for a Given Task, in Proceedings of the First International Conference in Knowledge Discovery and Data Mining, AAAI Press, Menlo Park, CA. 69-88.

[65] D.S. Shiau, Q. Luo, S.L. Gilmore, S.N. Roper, P.M. Pardalos, J.C. Sackellares, L.D. Iasemidis, 2000. Epileptic seizures resetting revisited. Epilepsia. 41, S7, 208-209

[66] V. Vapnik, S.E. Golowich, and A. Smola, 1997. Support Vector Method for Function Approximation, Regression Estimation, and Signal Processing, in Advances in Neural Information Processing Systems 9, M.C. Mozer, M.I. Jordan, and T. Petsche (eds.), MIT Press, Cambridge, MA.

[67] V.N. Vapnik, 1995. The Nature of Statistical Learning Theory, Springer, New York.

[68] D. Watts, 1999. Small Worlds: The Dynamics of Networks Between Order and Randomness, Princeton University Press.

[69] D. Watts and S. Strogatz, 1998. Collective dynamics of 'small-world' networks, *Nature* **393** 440-442.

HANDBOOK OF COMBINATORIAL OPTIMIZATION (VOL 5)
D.-Z. Du and P.M. Pardalos (Eds.) pp. 259-311

The Generalized Assignment Problem and Extensions

Dolores Romero Morales
Saïd Business School
University of Oxford, Oxford 0X1 1HP, United Kingdom
E-mail: Dolores.Romero-Morales@sbs.ox.ac.uk

H. Edwin Romeijn
Department of Industrial and Systems Engineering
University of Florida, Gainesville, FL 32611
E-mail: romeijn@ise.ufl.edu

Contents

1 Introduction

Consider that we have a set of tasks that need to be processed, and a set of agents that are able to process these tasks. A limited amount of a single resource is available to each of the agents, and each task requires a particular amount of this resource when it is processed by a particular agent. The resource consumption may depend on which agent processes a given task. The *Generalized Assignment Problem* is then the problem of assigning each

task to exactly one agent, so that the total cost of processing all tasks is minimized and no agent exceeds its resource capacity.

The Generalized Assignment Problem was inspired by real-life problems. The first problem of this form was studied by De Maio and Roveda [20]. They consider a transportation problem where each demand point must be supplied by exactly one supply point, sometimes called *source.* Here the agents are the supply points and the tasks are the demand points. Clearly, the requirements of the demand points do not depend on the particular supply point that supplies it, i.e., the requirements are *agent-independent.* Srinivasan and Thompson [72] extend this model by making the requirements *agent-dependent,* and are the first ones to propose the model that is known today as the Generalized Assignment Problem. The term *Generalized Assignment Problem (GAP)* for this type of problem was first introduced by Ross and Soland [67]. When the tasks are jobs to be performed, and the agents are computer networks, we obtain the problem described by Balachandran [3]. Another example is the fixed-charge plant location problem, where the agents are capacitated plants and the tasks are customers, where each of the customer demands must be supplied from a single plant (see Geoffrion and Graves [31]). Other applications that have been studied are the p-median location problem (see Ross and Soland [68]), the maximal covering location problem (see Klastorin [44]), various routing problems (see Fisher and Jaikumar [25]), R & D planning problems (see Zimokha and Rubinshtein [79]), and the loading problem in flexible manufacturing systems (see Kuhn [46]).

The GAP with m *agents* and n *tasks* can be formulated as an integer programming problem by defining the zero-one decision variables x_{ij}, where $x_{ij} = 1$ if task j is assigned to agent i, and $x_{ij} = 0$ otherwise:

$$\text{minimize} \sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij}$$

subject to

$$\sum_{j=1}^{n} a_{ij} x_{ij} \leq b_i \qquad i = 1, \ldots, m \tag{1}$$

$$\sum_{i=1}^{m} x_{ij} = 1 \qquad j = 1, \ldots, n \tag{2}$$

$$x_{ij} \in \{0,1\} \qquad i = 1, \ldots, m;\ j = 1, \ldots, n.$$

The cost coefficients c_{ij}, the requirement coefficients a_{ij}, and the capacity parameters b_i are all assumed to be nonnegative scalars. Constraints (2) are known in the literature as *semi-assignment constraints.* Some applications call for a maximization, rather than a minimization, formulation, which is clearly equivalent from an optimization point of view. See, for example, Martello and Toth [49], Fisher et al. [26], and Savelsbergh [69].

Several extensions of the GAP have been proposed in the literature in order to more closely describe various real-world applications. Srinivasan and Thompson [72] mention an extension of the model where the capacity of the agents b_i can be increased at a certain cost, which is linear in the amount of capacity added. Neebe and Rao [55] consider a fixed-charge version of the GAP with agent-independent requirements where each agent processing at least one task incurs a fixed cost. This can be used for example to model setup costs for the agents. Even though these two extensions have been proposed for the GAP with agent-independent requirements, they can clearly be relevant in the case of general requirements as well.

As shown by Ross and Soland [68], some location problems can be modeled as a GAP. When considering the location of emergency service facilities, a min-max objective function is more adequate than the classical min-sum. Mazzola and Neebe [53] consider two min-max formulations of the GAP. The first one is called the Task Bottleneck General Assignment Problem (Task BGAP) where the objective function to be minimized is equal to

$$\max_{i=1,\ldots,m;\, j=1,\ldots,n} c_{ij}x_{ij}.$$

The Task BGAP can model the location of emergency service facilities where the response time must be minimized. Martello and Toth [51] propose various relaxations for the Task BGAP. The second version is the Agent Bottleneck General Assignment Problem (Agent BGAP) where the objective function to be minimized is equal to

$$\max_{i=1,\ldots,m} \sum_{j=1}^{n} c_{ij}x_{ij}.$$

In this min-max formulation the goal is to minimize the costs incurred by the agent facing the largest costs. This model is applicable for instance for the problem of minimizing the makespan in a parallel machine scheduling problem.

Mazzola [52] considers nonlinear capacity interaction in the GAP. He mentions that this problem can be found in hierarchical production planning

problems where product families must be assigned to production facilities and a changeover is translated into nonlinear capacity interaction between the product families assigned to the same facility.

As mentioned above, the GAP assumes that there is just one resource available to the agents. Gavish and Pirkul [29] propose the Multi-Resource Generalized Assignment Problem (hereafter MRGAP) where tasks consume several resources when being processed by the agents. Campbell and Langevin [13] show an application of the MRGAP to the assignment of snow removal sectors to snow disposal sites in the city of Montreal. Blocq [10] studies another application in the distribution of gasoline products. An oil company is interested in minimizing the costs when delivering their oil products (super, unleaded petrol, etc.) from the depots to the different petrol stations under some restrictions. For each depot and type of product the total amount delivered has to be in a certain range. The same has to hold for the aggregation of all products, and also for some subsets of products. This problem can be modeled as a MRGAP with lower bounds on the consumption of the resources. He points out that companies sometimes make agreements with other oil companies to share their depots to a certain extent. To deal with these restrictions, Blocq et al. [11] have extended the MRGAP considering constraints on the consumption of each resource for each subset of agents (rather than each individual agent only).

The multi-level generalized assignment problem was first described by Glover et al. [33], in the context of the lot-sizing and machine loading problem for multiple products. In this extension of the GAP, agents can process tasks with different levels of efficiency, and tasks must be assigned to agents at a specific level of efficiency. This problem was studied further by Laguna et al. [47].

Park et al. [57] argue that tasks may need to be performed by more than one agent to increase reliability, thus introducing the Generalized Multi-Assignment Problem. They suggest an application where files are allocated to several computing sites so that files will be still available even if one computing site fails.

Kogan et al. [45] consider a continuous-time version of the GAP incorporating the order in which the tasks are processed by the agents. In this model tasks can be split over different agents. Shtub and Kogan [71] extend the model to the MRGAP. Since these models differ qualitatively from the GAP in that tasks can be processed by more than one agent, we will not analyze these models any further.

Solution approaches for the GAP have received significant attention in

the literature. Both exact and heuristic solution procedures have been proposed to solve this problem, most of which have been summarized by Cattrysse and Van Wassenhove [17] and Osman [56]. Some research has been devoted to probabilistic analyses of the GAP, see Dyer and Frieze [21], Romeijn and Piersma [60], and Romeijn and Romero Morales [61].

Many of the algorithms and results given in the literature for the GAP can be extended to a more general class of convex assignment problems which includes some of the extensions mentioned above. In the remainder, we will present many results for this class of convex assignment problems, and pay special attention to the particular case of the GAP.

The outline of this chapter is as follows. In Section 2 we will introduce a class of convex assignment problems (CAP), and we will show that the GAP and some of its extensions proposed in the literature to date can be formulated as a CAP. In Section 3 we will study the computational complexity of the CAP, and in Section 4 we propose different approaches to obtain lower bounds for the CAP. In Section 5 we discuss general solution procedures approaches for solving the CAP. In Section 6 we focus on the generation of experimental data for the CAP, for purposes of testing and comparing solution procedures. In Section 7 we present some additional results for the GAP, which cannot be readily extended to the CAP. In Section 8 we particularize the results obtained throughout this chapter for the MRGAP. Finally, in Section 9 we discuss the application of these results to a class of multi-period single-sourcing problems.

2 A Class of Convex Assignment Problems

2.1 Introduction

In a general assignment problem there are tasks which need to be processed and agents which can process these tasks. Each agent faces a set of capacity constraints and a cost when processing the tasks. The problem is then how to assign each task to exactly one agent, so that the total cost of processing the tasks is minimized and each agent does not violate his capacity constraints. Freling et al. [27] introduce the class of *Convex Assignment Problems (CAP)* where each agent's capacity constraint and cost function are convex. The problem can be formulated as follows:

$$\text{minimize} \sum_{i=1}^{m} g_i(x_{i\cdot})$$

subject to

$$\begin{array}{rcll} \omega_i(x_{i\cdot}) & \leq & b_i & i = 1, \ldots, m \\ \sum_{i=1}^{m} x_{ij} & = & 1 & j = 1, \ldots, n \\ x_{ij} & \in & \{0,1\} & i = 1, \ldots, m;\ j = 1, \ldots, n \end{array}$$

where $g_i : \mathbb{R}^n \to \mathbb{R}$ and $\omega_i : \mathbb{R}^n \to \mathbb{R}^{k_i}$ are convex functions and $b_i \in \mathbb{R}^{k_i}_+$, for all $i = 1, \ldots, m$. The GAP is clearly an example of a convex assignment problem, where the functions g_i and ω_i associated with agent i are linear in $x_{i\cdot}$, and $k_i = 1$ for all $i = 1, \ldots, m$.

To deal with many variants of multi-period single-sourcing problems (hereafter MPSSP) using a single solution approach, Romero Morales [66] introduces a subclass of convex capacitated assignment problems where the capacity constraints are linear. These models, which combine inventory and transportation decisions, are suitable when evaluating the total costs of a logistics distribution network in a dynamic environment. The MPSSP is a CAP and we will discuss this problem in more detail in Section 9.

The purpose of introducing the class of convex assignment problems is to deal with the GAP and many of its proposed extensions in a unified manner. We devote this section to finding the CAP formulations for these extensions (or to establish where they differ from a CAP).

Classes other than CAP have been proposed in the literature. Most notably, Ferland et al. [23] introduce a more general class of assignment problems, and discuss the applicability of object-oriented programming by developing software containing several heuristics.

2.2 Extensions of the GAP

2.2.1 Convex extensions of the GAP

The GAP with flexible capacities. Srinivasan and Thompson [72] extend the GAP by allowing for the use of additional capacity at a cost which is linear in the amount of capacity added. Let B_i be the maximal capacity at agent i where $B_i \geq b_i$ for all $i = 1, \ldots, m$, and ξ_i be the unit cost of increasing the capacity available to agent i beyond the base capacity b_i. Then, the GAP with additional capacity can be formulated as

$$\text{minimize} \sum_{i=1}^{m} \left(\sum_{j=1}^{n} c_{ij} x_{ij} + \xi_i \max \left\{ 0, \sum_{j=1}^{n} a_{ij} x_{ij} - b_i \right\} \right)$$

subject to

$$\begin{aligned} \sum_{j=1}^{n} a_{ij}x_{ij} &\leq B_i & i &= 1,\ldots,m \\ \sum_{i=1}^{m} x_{ij} &= 1 & j &= 1,\ldots,n \\ x_{ij} &\in \{0,1\} & i &= 1,\ldots,m;\ j = 1,\ldots,n. \end{aligned}$$

As for the GAP, the capacity constraints are linear. Furthermore, the objective function is convex piecewise linear, and thus the GAP with additional capacity is a CAP.

The GAP with fixed-charge costs. Neebe and Rao [55] study a fixed-charge version of the GAP. Let ν_i be the fixed cost incurred by agent i when this agent processes at least one task. The fixed-charge GAP can then be formulated as follows:

$$\text{minimize} \left(\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij} + \sum_{i=1}^{m} \nu_i y_i \right)$$

subject to

$$\begin{aligned} \sum_{j=1}^{n} a_{ij}x_{ij} &\leq b_i y_i & i &= 1,\ldots,m \\ \sum_{i=1}^{m} x_{ij} &= 1 & j &= 1,\ldots,n \\ x_{ij} &\in \{0,1\} & i &= 1,\ldots,m;\ j = 1,\ldots,n \\ y_i &\in \{0,1\} & i &= 1,\ldots,m. \end{aligned}$$

Observe that, without loss of optimality, $y_i = \max_{j=1,\ldots,n} x_{ij}$. Therefore, the fixed-charge GAP can be reformulated as a nonlinear integer problem in the assignment variables only as follows:

$$\text{minimize} \sum_{i=1}^{m} \left(\sum_{j=1}^{n} c_{ij}x_{ij} + \nu_i \max_{j=1,\ldots,n} x_{ij} \right)$$

subject to

$$\begin{aligned} \sum_{j=1}^{n} a_{ij}x_{ij} &\leq b_i & i &= 1,\ldots,m \\ \sum_{i=1}^{m} x_{ij} &= 1 & j &= 1,\ldots,n \\ x_{ij} &\in \{0,1\} & i &= 1,\ldots,m;\ j = 1,\ldots,n. \end{aligned}$$

Again, the capacity constraints are linear and the objective function piecewise linear convex, so that the fixed-charge GAP is a CAP.

The Multi-Resource GAP. Gavish and Pirkul [29] propose the Multi-Resource Generalized Assignment Problem, in which tasks consume more than one resource when being processed by the agents. This problem can be formulated as

$$\text{minimize} \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$$

subject to

$$\begin{aligned} \sum_{j=1}^{n} t_{ijk}x_{ij} &\leq b_{ik} & i &= 1,\ldots,m;\ k = 1,\ldots,K \\ \sum_{i=1}^{m} x_{ij} &= 1 & j &= 1,\ldots,n \\ x_{ij} &\in \{0,1\} & i &= 1,\ldots,m;\ j = 1,\ldots,n. \end{aligned}$$

This clearly is an example of a CAP formulation where both the objective function and the capacity constraints are linear and $k_i = K$ for all $i = 1,\ldots,m$.

Nonlinear capacity interactions. Mazzola [52] allows for nonlinear capacity interaction, yielding the following model formulation:

$$\text{minimize} \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$$

subject to

$$\rho_i(x_{i\cdot}) \leq b_i \qquad i = 1,\ldots,m$$

$$\begin{aligned} \sum_{i=1}^{m} x_{ij} &= 1 && j = 1, \ldots, n \\ x_{ij} &\in \{0,1\} && i = 1, \ldots, m;\ j = 1, \ldots, n \end{aligned}$$

where $\rho_i : \mathbb{R}^n \to \mathbb{R}$ is a general nonlinear function and $b_i \in \mathbb{R}_+$ for all $i = 1, \ldots, m$. It is straightforward to see that the GAP with nonlinear capacity interaction is a CAP if and only if the functions ρ_i are all convex.

2.2.2 Extensions of the GAP outside the class CAP

Several generalizations of the GAP that have received attention in the literature do not fall within the class of Convex Assignment Problems. Nevertheless, many algorithmic approaches developed for the GAP can easily be extended to these problems as well. We will describe some of these problems below.

Bottleneck GAPs. Mazzola and Neebe [53] propose two min-max formulations for the GAP, the Task BGAP and the Agent BGAP. The feasible regions for these two problems are the same as for the GAP, and their objectives are, respectively,

$$\text{minimize} \max_{i=1,\ldots,m;\, j=1,\ldots,n} c_{ij} x_{ij}$$

and

$$\text{minimize} \max_{i=1,\ldots,m} \sum_{j=1}^{n} c_{ij} x_{ij}.$$

Unfortunately, the objective function for these two problems is not separable in the agents, implying that both the Task BGAP and the Agent BGAP cannot be formulated as a CAP.

The Multi-Level GAP. Glover et al. [33] propose the multi-level generalized assignment problem where agents can process tasks at different levels of efficiency, and tasks must be assigned to agents at a specific level of efficiency. This problem can be formulated as

$$\text{minimize} \sum_{i=1}^{m} \sum_{j=1}^{n} \sum_{\ell=1}^{L} c_{ij\ell} x_{ij\ell}$$

subject to

$$\begin{aligned}
\sum_{j=1}^{n}\sum_{\ell=1}^{L} t_{ij\ell}x_{ij\ell} &\leq b_i & i &= 1,\ldots,m \\
\sum_{\ell=1}^{L}\sum_{i=1}^{m} x_{ij\ell} &= 1 & j &= 1,\ldots,n \\
x_{ij\ell} &\in \{0,1\} & i &= 1,\ldots,m;\ j = 1,\ldots,n;\ \ell = 1,\ldots,L.
\end{aligned}$$

This model differs from the rest of the extensions of the GAP analyzed in this section since the tasks are assigned to (agent,efficiency level)-pairs. Therefore, the agents, as understood in the CAP, correspond to the (agent,efficiency level)-pairs in the multi-level GAP. However, since there are no capacity constraints for individual (agent,efficiency level)-pairs, but only aggregate capacity constraints for each agent, the multi-level GAP belongs to a more general class of convex assignment problems which allows for aggregate capacity constraints over subsets of agents.

The Generalized Multi-Assignment Problem. Finally, Park et al. [57] propose to have tasks duplicated at several agents. Let $r_j \leq m$, $j = 1,\ldots,m$, be the number of times that task j must be duplicated. The Generalized Multi-Assignment Problem can be formulated as

$$\text{minimize} \sum_{i=1}^{m}\sum_{j=1}^{n} c_{ij}x_{ij}$$

subject to

$$\begin{aligned}
\sum_{j=1}^{n} a_{ij}x_{ij} &\leq b_i & i &= 1,\ldots,m \\
\sum_{i=1}^{m} x_{ij} &= r_j & j &= 1,\ldots,n \\
x_{ij} &\in \{0,1\} & i &= 1,\ldots,m;\ j = 1,\ldots,n.
\end{aligned}$$

Obviously, the Generalized Multi-Assignment Problem is a CAP only if $r_j = 1$ for all $j = 1,\ldots,n$, in which case it actually reduces to a GAP.

3 Complexity

Due to the abundance of practical applications, capacitated assignment problems have been studied extensively from an algorithmic point of view. As will shown in Section 5, various exact algorithms and heuristic procedures have been proposed in the literature. Nevertheless, all approaches suffer from the $\mathcal{NP}$-*Hardness* of the GAP (see Fisher et al. [26]). Moreover, the decision problem associated with the feasibility of the GAP is an $\mathcal{NP}$-*Complete* problem (see Martello and Toth [49]). Therefore, even to test whether a problem instance has at least one feasible solution is computationally hard. In these proofs, problem instances of the GAP with agent-independent requirements are used. We will summarize these proofs in the following theorem.

Theorem 3.1 (cf. Fisher et al. [26], Martello and Toth [49]) *The GAP with agent-independent requirements is an* $\mathcal{NP}$-Hard *problem. Moreover, the decision problem associated with its feasibility is an* $\mathcal{NP}$-Complete *problem.*

Proof. To show the $\mathcal{NP}$-*Hardness* of the GAP with agent-independent requirements, we will prove that the 2-partition problem is reducible to the GAP. Given a set of n real numbers $A_1, \ldots, A_n$, the 2-partition problem asks if there exists a set $S \subseteq \{1, \ldots, n\}$ such that $\sum_{j \in S} A_j = \sum_{j \in \{1,\ldots,n\} \setminus S} A_j$. This problem is equivalent to a GAP with agent-independent requirements where $m = 2$, $a_{ij} = A_j$ for all $i = 1, \ldots, m$, $b_1 = b_2 = \frac{1}{2} \sum_{j=1}^{n} A_j$, and are c_{ij} arbitrary.

Following a similar argument, we can derive that the decision problem associated with the feasibility of the GAP with agent-independent requirements is an $\mathcal{NP}$-*Complete* problem. □

The feasible region of the GAP with agent-independent requirements is defined by m knapsack constraints, each one associated with an agent. In a general convex assignment problem, each agent faces some capacity constraints. Therefore, the decision problem associated with the feasibility of the CAP is as difficult as for the GAP with agent-independent requirements. Similar conclusions can be drawn for the optimization problem since the objective function of the GAP with agent-independent requirements is a particular case of the one for the CAP. Therefore, we can derive the following corollary.

Corollary 3.2 *The CAP is an* $\mathcal{NP}$-Hard *problem. Moreover, the decision problem associated with its feasibility is an* $\mathcal{NP}$-Complete *problem.*

4 Lower Bounds for the CAP

4.1 Introduction

Most of the solution procedures for (nonlinear) integer minimization problems involve the computation of lower bounds. Moreover, due to the complexity of these minimization problems, the quality of the performance of heuristic solution methods is illustrated by error bounds which also necessitate the computation of lower bounds. In this section, we will present several approaches to obtain lower bounds for the CAP.

4.2 Relaxing the Integrality Constraints

The most straightforward lower bound for a (nonlinear) integer programming problem in the minimization form is the optimal value of the relaxation of the integrality constraints. For the CAP, this yields, in general, a nonlinear problem since the objective function and the capacity constraints are convex. This relaxation can be formulated as follows

$$\text{minimize} \sum_{i=1}^{m} g_i(x_{i\cdot})$$

subject to (RCAP)

$$\begin{aligned} \omega_i(x_{i\cdot}) &\leq b_i && i = 1, \ldots, m \\ \sum_{i=1}^{m} x_{ij} &= 1 && j = 1, \ldots, n \\ x_{ij} &\geq 0 && i = 1, \ldots, m;\ j = 1, \ldots, n. \end{aligned}$$

If the optimal solution for (RCAP) does not contain any fractional variable, then this clearly is the optimal solution for the CAP as well. In general, however, this is not the case. Given a feasible solution x for R(CAP), we will say that task j is a *non-split task* of x if there exists an index i such that $x_{ij} = 1$. The remaining tasks, called *split tasks,* are assigned to more than one agent. Let $B(x)$ be the set of split tasks in x, i.e.

$$B(x) = \{j = 1, \ldots, n : \exists i = 1, \ldots, m \text{ such that } 0 < x_{ij} < 1\}.$$

In the following we show that for some subclasses of CAPs there exist optimal solutions so that the number of split tasks is bounded from above by a constant independent of the number of tasks.

We will first analyze the linear case.

Lemma 4.1 *If the functions g_i and w_i are linear for all $i = 1, \ldots, m$, then for all basic optimal solutions x^{RCAP} of (RCAP) we have*

$$|B(x^{\mathrm{RCAP}})| \leq \sum_{i=1}^{m} k_i.$$

Proof. The proof resembles that of Lemma 6.4.1 in Romero Morales [66]. Rewrite the problem (RCAP) with equality constraints and nonnegative variables only. We then obtain a problem with, in addition to the assignment constraints, $\sum_{i=1}^{m} k_i$ equality constraints. Now consider a basic optimal solution to (RCAP). The number of variables having a nonzero value in this solution is no larger than the number of equality constraints in the reformulated problem. Since there is at least one nonzero assignment variable corresponding to each assignment constraint, and exactly one nonzero assignment variable corresponding to each assignment that is feasible with respect to the integrality constraints of (RCAP), there can be no more than $\sum_{i=1}^{m} k_i$ assignments that are split. Thus, the desired inequality follows. □

A similar result can be found when the functions g_i are allowed to be piecewise linear convex functions, i.e., there exist $\kappa_i \in \mathbb{N}$ and g_i^κ for each $\kappa = 1, \ldots, \kappa_i$ so that $g_i^\kappa : \mathbb{R}^n \to \mathbb{R}$ is linear and $g_i = \max_{\kappa=1,\ldots,\kappa_i} g_i^\kappa$. Following standard techniques, we can obtain an equivalent linear programming formulation for the corresponding relaxation:

$$\text{minimize} \sum_{i=1}^{m} z_i$$

subject to

$$\begin{aligned}
z_i &\geq g_i^\kappa(x_{i\cdot}) && i = 1, \ldots, m;\ \kappa = 1, \ldots, \kappa_i \\
\omega_i(x_{i\cdot}) &\leq b_i && i = 1, \ldots, m \\
\sum_{i=1}^{m} x_{ij} &= 1 && j = 1, \ldots, n \\
x_{ij} &\geq 0 && i = 1, \ldots, m;\ j = 1, \ldots, n.
\end{aligned}$$

Corollary 4.2 *If the functions g_i are piecewise linear convex and the functions w_i are linear for all $i = 1, \ldots, m$, then for all basic optimal solutions x^{RCAP} of (RCAP) we have*

$$|B(x^{\mathrm{RCAP}})| \leq \sum_{i=1}^{m} \kappa_i + \sum_{i=1}^{m} k_i.$$

Proof. Similar to the proof of Lemma 4.1. □

Benders and Van Nunen [9] derive a slightly tighter bound on the number of split tasks for the GAP, by showing that there exists an optimal solution to the LP-relaxation of the GAP in which the number of split tasks is at most equal to the number of agents used to full capacity. Dyer and Frieze [21] also show that the number of fractional variables in any basic solution for the LP-relaxation of the GAP is at most equal to two times the number of agents.

Jörnsten and Värbrand [42] and Cattrysse et al. [14] improve the linear programming bound for the GAP by adding valid inequalities for each knapsack constraint. In the latter a more detailed explanation of the procedure is provided. For each agent i, they use a separation procedure to search for strong covers $C_i \subseteq \{1, \ldots, n\}$ (see Balas and Zemel [4]). The corresponding valid inequality $\sum_{j=1}^{n} x_{ij} \leq |C_i| - 1$ is strengthened by a lifting procedure.

4.3 An Improved Lower Bound Based on a Set Partitioning Formulation

The CAP can alternatively be formulated as a set partitioning problem (SP), which leads to very successful solution approaches. This reformulation was first introduced for the GAP by Cattrysse et al. [16], and Savelsbergh [69], and generalized to the CAP by Freling et al. [27]. Barnhart et al. [7] discuss set partitioning models in a more general context.

In particular, a feasible solution for the CAP can be seen as a partition of the set of tasks $\{1, \ldots, n\}$ into m subsets. Each element of the partition is associated with one of the m agents. Now let L_i be the number of subsets of tasks that can feasibly be assigned to agent i $(i = 1, \ldots, m)$. Let $\alpha_{i\cdot}^{\ell}$ denote the ℓ-th subset (for fixed i), i.e., $\alpha_{ij}^{\ell} = 1$ if task j is an element of subset ℓ for agent i, and $\alpha_{ij}^{\ell} = 0$ otherwise. We will call $\alpha_{i\cdot}^{\ell}$ the ℓ-th *column* for agent i. The vector $\alpha_{i\cdot}^{\ell}$ is a zero-one feasible solution to the multi-knapsack constraints associated with agent i. The set partitioning problem can be

formulated as follows:

$$\text{minimize} \sum_{i=1}^{m} \sum_{\ell=1}^{L_i} g_i(\alpha_{i\cdot}^{\ell})\, y_i^{\ell}$$

subject to (SP)

$$\sum_{i=1}^{m} \sum_{\ell=1}^{L_i} \alpha_{ij}^{\ell} y_i^{\ell} = 1 \qquad j = 1, \ldots, n \tag{3}$$

$$\sum_{\ell=1}^{L_i} y_i^{\ell} = 1 \qquad i = 1, \ldots, m \tag{4}$$

$$y_i^{\ell} \in \{0, 1\} \qquad \ell = 1, \ldots, L_i;\ i = 1, \ldots, m$$

where y_i^{ℓ} is equal to 1 if column ℓ is chosen for agent i, and 0 otherwise. As mentioned by Barnhart et al. [7], the convexity constraint (4) for agent i $(i = 1, \ldots, m)$ can be written as

$$\sum_{\ell=1}^{L_i} y_i^{\ell} \leq 1$$

if $\alpha_{ij} = 0$ for each $j = 1, \ldots, n$ is a feasible column for agent i with associated costs $g_i(\alpha_{i\cdot}) = 0$. Note that (SP) is a zero-one integer program, while the CAP is, in general, a zero-one convex program. Moreover, this formulation allows for adding additional constraints that may be difficult to express analytically. On the downside, the number of variables in (SP) is impractically large for any realistic problem size. However, as we will discuss below, this problem can be solved by a column generation framework, and as such be embedded in a branch-and-price procedure.

Since this problem is a zero-one programming problem, a bound on its optimal value can be obtained in a similar way as for the standard formulation by relaxing the integrality constraints (see Section 4.2). An important property of the set partitioning formulation however, is that the part of any feasible solution of the LP-relaxation LP(SP) corresponding to a particular agent is a convex combination of zero-one candidate solutions for that agent, which can be used to show that LP(SP) gives a bound on the optimal solution value of (SP) (and thus the CAP) that is at least as tight (and usually tighter) as the one obtained by straightforwardly relaxing the integrality constraints in the CAP, yielding the relaxation (RCAP). Hence, if we let $v(\text{(RCAP)})$ and $v(\text{LP(SP)})$ denote the optimal objective values of (RCAP) and LP(SP), respectively, then the following result holds.

Theorem 4.3 (cf. Freling et al. [27]) *The following inequality holds:*

$$v((RCAP)) \leq v(LP(SP)).$$

As mentioned above, the number of variables (columns) in (SP), and thus also in LP(SP), is prohibitively large. Therefore, this problem is usually solved using a Column Generation procedure. There are two critical factors in this procedure. The structure of the pricing problem is a major issue in the success of the column generation procedure. Moreover, standard branching rules may destroy the structure of the pricing problem. These two issues have been studied by Freling et al. [27].

The column generation procedure is based on the observation that an optimal basic solution of LP(SP) will contain relatively few decision variables that are nonzero. Therefore, only a small number of columns, say N, will initially be included in the formulation. Let $(u^*(N), \delta^*(N))$ be the optimal dual solution to this approximation to LP(SP). The following, so-called *pricing problem,* can be used to verify whether the solution thus obtained is optimal for LP(SP), and if not supplies a column to be added to the problem. Observe that each column is associated with a particular agent, and therefore a pricing problem is defined for each agent. For agent i the pricing problem reads

$$\text{minimize} \quad \left(g_i(z) - \sum_{j=1}^{n} u_j^*(N) z_j + \delta_i^*(N) \right)$$

subject to

$$\begin{aligned} \omega_i(z) &\leq b_i \\ z_j &\in \{0,1\} \qquad\qquad j = 1, \ldots, n. \end{aligned}$$

If a solution with negative objective value exists, it corresponds to a column that should be added to the current approximation of LP(SP). If not, the current solution is optimal for LP(SP). Freling et al. [27] have identified a subclass of CAPs for which the pricing problem can be solved efficiently. In this subclass, the pricing problem is a so-called Penalized Knapsack Problem (PKP), which is a knapsack problem in which the objective function includes an additional convex term penalizing the total use of capacity in the knapsack. It can be shown that the optimal solution of the problem when the integrality constraints are relaxed has a similar structure to the optimal solution of the LP-relaxation of the Knapsack Problem.

4.4 Lagrangean Relaxations

4.4.1 Introduction

The Lagrangean relaxation is a more elaborate technique than the relaxation of the integrality constraints to obtain lower bounds for a (nonlinear) integer programming problem, see Geoffrion [30]. There are optimization problems which would be much easier to solve if some of their constraints were not present. The Lagrangean relaxation is based on eliminating these constraints by dualizing them into the objective function with a vector of multipliers. For each of these vectors, we obtain a lower bound to the original problem. The best possible lower bound, known as the Lagrangean bound, is given by maximizing over the set of multipliers. One may solve this maximization problem to optimality or heuristically to find a *good* vector of multipliers. Fisher [24] enumerates several methods which have been used to solve this maximization problem. The Lagrangean function, i.e., the objective function of this maximization problem is, in general, nondifferentiable, therefore we can use the subgradient method introduced by Held et al. [40]. The multiplier adjustment method moves the multipliers along a set of predefined directions. Finally, a column generation scheme may also be used to search for good multipliers.

In the particular case of the CAP, there are two possible Lagrangean bounds depending on which type of constraints are dualized. We will describe them separately.

4.4.2 Dualizing the capacity constraints

Let $\lambda = (\lambda_1^\top, \ldots, \lambda_m^\top)^\top$ where $\lambda_i \in I\!R^{k_i}$. If the capacity constraints are dualized we obtain the following lower bound:

$$L^c(\lambda) = \text{minimize} \left(\sum_{i=1}^{m} g_i(x_{i\cdot}) + \sum_{i=1}^{m} \lambda_i^\top (\omega_i(x_{i\cdot}) - b_i) \right)$$

subject to

$$\begin{aligned} \sum_{i=1}^{m} x_{ij} &= 1 && j = 1, \ldots, n \\ x_{ij} &\in \{0,1\} && i = 1, \ldots, m;\ j = 1, \ldots, n \end{aligned}$$

where $\lambda_i \in I\!R_+^{k_i}$ for each $i = 1, \ldots, m$. For $\lambda = 0$, we obtain a straightforward lower bound based on leaving out the capacity constraints. The Lagrangean

bound is equal to

$$\max_{\lambda \in \mathbb{R}_+^{k1} \times \ldots \times \mathbb{R}_+^{km}} L^c(\lambda).$$

If the functions g_i and ω_i are linear for all $i = 1, \ldots, m$, the value of $L^c(\lambda)$ can be given explicitly: an optimal solution vector for the corresponding optimization problem is obtained by assigning each task to the agent that can process it at least cost. Unfortunately, this result does not hold in general, making the calculation of the Lagrangean bound computationally expensive for general CAPs.

The constraint matrix of the relaxed problem is totally unimodular. Thus, if the functions g_i and ω_i are linear for all $i = 1, \ldots, m$, the Lagrangean bound when relaxing the capacity constraints coincides with the optimal value of the relaxation of the integrality constraints of the CAP, see Geoffrion [30], which suggests that this Lagrangean bound is of limited use for the GAP. However, Chalmet and Gelders [18] argue that the Lagrangean bound can be calculated more efficiently than the LP-relaxation bound for large problem instances.

Ross and Soland [67] propose to relax the capacity constraints of the GAP to $a_{ij}x_{ij} \leq b_i$. If $b_i/a_{ij} < 1$ for some (i, j), it can be concluded that $x_{ij} = 0$. In all other cases, the constraints are redundant. As for the solution to the Lagrange relaxation mentioned above, this relaxation yields a simple cost minimization problem where each task is assigned to the agent that can process it in the cheapest way. In general, this solution will violate the capacity constraints of some of the agents. A stronger lower bound can be obtained as follows. Let i be an agent for which the capacity constraint is violated, and associate with each task assigned to this agent the minimal cost increase incurred when allocating it to another agent. Then, a knapsack problem is solved minimizing the total penalty while requiring that the used capacity is at least equal to the violation of the capacity constraint. The optimal solution value of this knapsack problem can be interpreted as the minimal cost to make this agent feasible with respect to its capacity constraint. Solving this knapsack problem for each agent whose capacity constraint is violated and adding the optimal values to the original bound yields a stronger lower bound on the optimal value of the GAP.

4.4.3 Dualizing the semi-assignment constraints

When we dualize the semi-assignment constraints we obtain the following lower bound:

$$L^{\mathrm{a}}(\mu) = \text{minimize } \left(\sum_{i=1}^{m} g_i(x_{i\cdot}) + \sum_{j=1}^{n} \mu_j \left(\sum_{i=1}^{m} x_{ij} - 1 \right) \right)$$

subject to

$$\begin{aligned} \omega_i(x_{i\cdot}) &\leq b_i && i = 1, \ldots, m \\ x_{ij} &\in \{0,1\} && i = 1, \ldots, m;\ j = 1, \ldots, n \end{aligned}$$

where $\mu = (\mu_j) \in \mathbb{R}^n$. As before, $L^{\mathrm{a}}(0)$ is equal to the lower bound obtained by leaving out the semi-assignment constraints. Again the best lower bound is given by

$$\max_{\mu \in \mathbb{R}^n} L^{\mathrm{a}}(\mu).$$

Since the objective function and the feasible region are separable in the agents, the problem defining the value $L^{\mathrm{a}}(\mu)$ clearly decomposes into m convex optimization problems. Each of these optimization problems is non-linear, which suggests that for general convex functions this lower bound may be computationally expensive. On the other hand, the constraint matrix of the relaxed problem is not totally unimodular, so we can expect this procedure to yield better lower bounds than when relaxing the capacity constraints.

Chalmet and Gelders [18] and Fisher et al. [26] propose the Lagrangean relaxation of the semi-assignment constraints described above for the GAP. In the latter, a multiplier adjustment method (see Fisher [24]) is used to find good multipliers. Guignard and Rosenwein [37] propose some enhancements and additions to the approach of Fisher et al. [26]. In particular, they enlarge the set of possible directions used by the multiplier adjustment method. In addition, if the obtained primal solution violates the constraint

$$\sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} = n$$

then the corresponding surrogate constraint

$$\sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} \leq n \text{ or } \sum_{i=1}^{m} \sum_{j=1}^{n} x_{ij} \geq n$$

is added to the Lagrangean relaxation. This constraint is dualized into the objective function and the new scalar problem is solved. Karabakal et al. [43] argue that the multiplier adjustment methods proposed by Fisher et al. [26] and Guignard and Rosenwein [37] move along the first descent direction (for a maximization formulation of the GAP) found with nonzero step size. They propose to use the steepest descent direction. Martello and Toth [49] propose, for a maximization version of the GAP, to calculate bounds for the GAP by simply removing the semi-assignment constraints (corresponding to choosing the Lagrange multipliers $\mu_j = 0$). As seen above, the relaxed problem decomposes into m knapsack problems. As Ross and Soland [67] did for the deletion of the capacity constraints, the corresponding bound is improved by adding a lower bound on the penalty to be paid to satisfy the violated semi-assignment constraints.

4.5 Lagrangean Decomposition

The Lagrangean relaxation tries to find lower bounds by dualizing one type of constraints in the model. A Lagrangean decomposition approach consists of combining in one lower bound the structures obtained by the Lagrangean technique when relaxing different types of constraints, see Guignard and Kim [36]. Moreover, it is straightforward to prove that this lower bound is at least as good as the corresponding Lagrangean relaxations.

The CAP contains two clear types of constraints, namely the capacity and the semi-assignment constraints. In the previous section we have derived the two lower bounds corresponding to the relaxation of each of these constraints. The Lagrangean decomposition will try to combine these two structures. Let α and β be two nonnegative scalars so that $\alpha + \beta = 1$. We will duplicate the decisions variables and add the corresponding constraint to obtain the following equivalent formulation of the CAP:

$$\text{minimize} \left(\alpha \sum_{i=1}^{m} g_i(x_{i\cdot}) + \beta \sum_{i=1}^{m} g_i(y_{i\cdot}) \right)$$

subject to

$$\begin{aligned} \omega_i(x_{i\cdot}) &\leq b_i && i = 1, \ldots, m \\ \sum_{i=1}^{m} y_{ij} &= 1 && j = 1, \ldots, n \\ x_{ij} &= y_{ij} && i = 1, \ldots, m;\ j = 1, \ldots, n \end{aligned} \tag{5}$$

$$\begin{aligned} x_{ij} &\in \{0,1\} && i=1,\ldots,m;\ j=1,\ldots,n \\ y_{ij} &\in \{0,1\} && i=1,\ldots,m;\ j=1,\ldots,n. \end{aligned}$$

If we dualize constraints (5) into the objective function, the problem separates in the following two models

$$\text{minimize} \left(\alpha \sum_{i=1}^{m} g_i(x_{i\cdot}) + \sum_{i=1}^{m} \sum_{j=1}^{n} \eta_{ij} x_{ij} \right)$$

subject to

$$\begin{aligned} \omega_i(x_{i\cdot}) &\leq b_i && i=1,\ldots,m \\ x_{ij} &\in \{0,1\} && i=1,\ldots,m;\ j=1,\ldots,n \end{aligned}$$

and

$$\text{minimize} \left(\beta \sum_{i=1}^{m} g_i(y_{i\cdot}) - \sum_{i=1}^{m} \sum_{j=1}^{n} \eta_{ij} y_{ij} \right)$$

subject to

$$\begin{aligned} \sum_{i=1}^{m} y_{ij} &= 1 && j=1,\ldots,n \\ y_{ij} &\in \{0,1\} && i=1,\ldots,m;\ j=1,\ldots,n. \end{aligned}$$

Any value $\eta \in \mathbb{R}^{mn}$ yields a lower bound $L^{\mathrm{d}}(\eta)$, given by the sum of the optimal values of these two optimization problems. It is easy to see the purpose of the Lagrangean decomposition since the two problems obtained resemble the two Lagrangean relaxations discussed in the previous section. Again the best lower bound is given by the maximization of this expression over the set of multipliers $\eta \in \mathbb{R}^{mn}$. This lower bound is at least as good as the two Lagrangean relaxations of the CAP.

Theorem 4.4 *The bounds derived in this section satisfy*

$$\max_{\eta \in \mathbb{R}_+^{mn}} L^{\mathrm{d}}(\eta) \geq \max_{\mu \in \mathbb{R}^n} L^{\mathrm{a}}(\mu) \geq \max_{\lambda \in \mathbb{R}_+^{k_1} \times \ldots \times \mathbb{R}_+^{k_m}} L^{\mathrm{c}}(\lambda) \geq v(RCAP).$$

Proof. Similar to the proof for the GAP in Jörnsten and Näsberg [41]. □

Jörnsten and Näsberg [41] apply a Lagrangean decomposition approach to the GAP. In their limited numerical results, they consider $\alpha = \beta = \frac{1}{2}$. Barcia and Jörnsten [6] use the bound improving technique (see Barcia [5]) to tighten the bound obtained by Lagrangean decomposition. Observe that for all $\alpha, \beta \geq 0$ and $\alpha + \beta = 1$ a valid lower bound is obtained, so that one may attempt to find the best bound by maximizing the lower bound over α and β.

4.6 Surrogate Constraints

Finally, another method to generate lower bounds for a (nonlinear) integer programming problem is adding surrogate constraints, see Glover [32]. Given a subset of constraints and a vector of multipliers λ of the same size, we derive a new constraint for the problem by multiplying each constraint by λ and then aggregating. This new constraint is implied by the original ones. The lower bound is obtained by adding the surrogate constraint to the feasible region and deleting the original ones. Observe that we obtain different lower bounds depending on the type of constraints we surrogate.

This approach has only been applied to a very limited extent to the GAP, and has not been very successful. Therefore, we will not describe this approach in detail. Lorena and Narciso [48] propose the surrogate relaxation of the capacity constraints. In a later work, Narciso and Lorena [54] develop a Lagrangean/surrogate relaxation.

5 Solution Methods

5.1 Introduction

As we have seen in the previous sections, the GAP and its extensions are useful for modeling a wide variety of real-life problems. Therefore, we need for efficient solutions methods to find an optimal, or at least a good, solution for these problems. In an attempt to summarize all the solution methods presented in the literature for these problems, we separately consider heuristics (or approximate) methods, branch-and-bound procedures, and branch-and-price schemes. We will often describe the solution approaches for the CAP, and then pay particular attention to how they can be applied to the GAP.

5.2 Heuristics

5.2.1 Introduction

In Section 3 we have shown that the CAP is an *NP-Hard* problem. Therefore, solving large instances of the problem to optimality can require a substantial computational effort. This calls for heuristic approaches where good solutions can be found with a reasonable computational effort. There clearly are situations where a good solution is sufficient in its own right. But in addition, a good suboptimal solution can accelerate an exact procedure, for instance by allowing for the elimination of parts of the tree in a branch-and-bound procedure (see Section 5.3).

In the following we will describe several ways of developing heuristic solution approaches for the CAP. First, we describe heuristics based on the relaxation of the integrality constraints of the CAP. Then, we present heuristic approaches which search for primal solutions while solving the Lagrangean relaxation. The set partitioning formulation of the CAP suggests new heuristic procedures for the CAP. Finally, we will describe some greedy and meta-heuristics. We may point out that all heuristics suffer from the *NP-Completeness* of the CAP, and thus may fail to even find a feasible solution for the CAP. The literature on heuristic procedures for the GAP reports very little about their success in finding feasible solutions for the GAP.

5.2.2 Relaxing the integrality constraints

An optimal solution vector for (RCAP), say x^{RCAP}, is feasible with respect to the capacity constraints at the agents but, in general, it violates the integrality constraints, i.e., there exist tasks which are assigned to more than one agent. We may expect that a feasible solution vector for the CAP which almost coincides with x^{RCAP} is close to optimality. A straightforward way of generating this type of vector solutions is to fix all the non-split tasks of x^{RCAP}. We then obtain a reduced CAP with the same agents as in the original CAP and where only the split tasks of x^{RCAP} must be assigned. Feasible solutions to the reduced CAP completed with the assignments of the non-split tasks in x^{RCAP} are feasible solution vectors to the original CAP. The reduced CAP can be solved by an exact method or by a heuristic procedure.

Benders and Van Nunen [9] propose a heuristic to solve the reduced GAP while Cattrysse [15] solves it by a branch-and-bound technique. Cattrysse

[15] observes that adding cuts to the LP-relaxation increases the number of fractional variables in its optimal solution vector, and thus the number of split tasks. Numerical experiments show that this increases the success of his primal heuristic. Trick [74] uses another property of the LP-relaxation of the GAP to propose his LR-heuristic. He defines the variable associated with the assignment of task j to agent i useless if the requirement of task j on agent i is larger than the capacity of agent i, which means that without loss of optimality useless variables can be fixed to zero. The LR-heuristic solves the LP-relaxation of the GAP, define the reduced GAP, and fix to zero the useless variables of this new problem. This procedure is successively applied.

5.2.3 Lagrangean relaxation and decomposition

The Lagrangean relaxations presented in Section 4.4 yield solutions violating some of the dualized constraints. For example, when dualizing the capacity constraints we obtain primal solutions where there may be agents using more resources than the available capacity. If the semi-assignment constraints are dualized the corresponding primal solutions may contain tasks which are not assigned, or are assigned to more than one agent. Lagrangean decomposition described in Section 4.5 obtains solutions of both types. These solutions can be used as starting point for local exchange procedures which search for feasible solutions to the CAP.

Lorena and Narciso [48] propose two local exchange procedures to obtain a primal solution from the one obtained by the subgradient method applied to the Lagrangean relaxation of the capacity constraints of the GAP. If both fail, another heuristic is used to find a primal solution which has as an input the dual vector obtained by the subgradient method. Guignard and Rosenwein [37] propose an interchange procedure to find a primal solution from the one given by a multiplier adjustment method applied to the Lagrangean relaxation of the semi-assignment constraints of the GAP and improved by the addition of a surrogate constraint. Karabakal et al. [43] solve again by a multiplier adjustment method the Lagrangean relaxation of the semi-assignment constraints. They improve the feasibility of the obtained solution by a heuristic where some assignments are deleted and several knapsack problems are solved with the tasks not assigned. Järnsten and Näsberg [41] propose local exchange heuristics in an attempt to improve the feasibility of the solutions obtained by the Lagrangean decomposition of the GAP.

5.2.4 The set partitioning formulation

The CAP has been formulated as a set partitioning problem in Section 4.3. Therefore, a solution vector of the problem is seen as a collection of columns at most one per agent. Each column represents the set of tasks assigned to the corresponding agent. A very straightforward framework for a heuristic for the CAP based on this set partitioning formulation could be the following. First, we generate a subset of feasible columns for the CAP. Then, we search for a feasible solution for the CAP using these columns. As an example of such a heuristic, we could solve the LP-relaxation of (SP) to optimality by a column generation scheme and then use a branch-and-bound procedure to find the optimal integer solution associated with these columns.

Cattrysse [15] and Cattrysse et al. [16] develop a heuristic based on the set partitioning formulation of the GAP. They propose a column generation procedure for the LP-relaxation of (SP). A dual ascent heuristic is applied to the master problem. Since optimality is not guaranteed, several iterations of the subgradient method are applied to the Lagrangean relaxation of the constraints (3) in the current master problem. They look for primal solutions by reduction techniques combined with the enumeration procedure of Garfinkel and Nemhauser [28].

5.2.5 Greedy heuristics

In the following we will describe a class of greedy heuristics for the CAP. Suppose that each possible assignment of a task to an agent is evaluated by a pseudo-cost function $f(i,j)$, and that the *desirability* of assigning a task is measured by the difference between the second smallest and the smallest values of $f(i,j)$ over the set of agents. The greedy heuristic assigns tasks to their best agents in decreasing order of this difference. Along the way, some agents will not be able to handle some of the tasks due to the constraints faced by each agent, and consequently the values of the desirabilities will be updated taking into account that the two most desirable agents for each task should be feasible. The output of this heuristic is a vector of feasible assignments, which is (at least) a partial solution to the CAP. The challenge is to choose a pseudo-cost function that will yield high quality feasible solutions to the CAP.

Martello and Toth [49] propose the class of greedy heuristics described above for the GAP which has universally been used. They claim that their computational results suggest the following pseudo-cost functions as good

choices:

(i) $f(i,j) = c_{ij}$,

(ii) $f(i,j) = a_{ij}$,

(iii) $f(i,j) = a_{ij}/b_i$, and

(iv) $f(i,j) = (c_{ij} - t_j)/a_{ij}$

where $t_j > \max_{i=1,\ldots,m} c_{ij}$ for each $j = 1, \ldots, n$. The motivation for choosing the pseudo-cost function (i) is that it is desirable to assign a task to an agent that can process it as cheaply as possible, and for the pseudo-cost functions (ii) and (iii) is that it is desirable to assign a task to an agent that can process it using the least (absolute or relative) capacity. The pseudo-cost function (iv) tries to consider the effects of the previous pseudo-cost functions jointly. (Observe that the definition of this pseudo-cost function depends on the parameters t_j which do not have a clear meaning.) With the same purpose as the pseudo-cost function (iv), Romeijn and Romero Morales [61] propose the family of pseudo-cost functions

$$f_\lambda(i,j) = c_{ij} + \lambda_i a_{ij}$$

where $\lambda \in \mathbb{R}^m_+$. Romeijn and Romero Morales [61] and Romero Morales [66] show that, for large problem instances (as measured by the number of tasks) generated by the stochastic model for the GAP proposed by Romeijn and Piersma [60], the greedy heuristic finds a feasible and optimal solution with probability one when λ_i is equal to the optimal dual multiplier of the i-th capacity constraint in the LP-relaxation of the GAP. Moreover, conditions are given under which there exists a unique vector of multipliers, only depending on the number of agents and the probabilistic model for the parameters of the problem, so that the corresponding heuristic is asymptotically feasible and optimal.

Martello and Toth [49] add a local search phase to the greedy heuristic in order to try to improve the objective value of the current solution. Romero Morales [66] proposes two local exchange procedures to improve the feasibility and the objective value of the solution found by the greedy heuristic. Wilson [76] proposes another greedy heuristic procedure for the GAP which use the solution where each task is assigned to its cheapest agent as the starting point for an exchange procedure where the violation of the capacity constraints is decreased in each step.

5.2.6 Greedy Randomized Adaptive Search Procedures

Greedy Randomized Adaptive Search Procedures (GRASP) are relatively new and powerful tools to solve combinatorial problems, see Feo and Resende [22] for an overview of GRASP. It consists of an iterative procedure in which in each iteration a random solution vector for the problem is constructed, which is subsequently improved by a local search procedure. The construction phase follows the same idea as the greedy heuristics described in the previous section, i.e., elements are ordered in a candidate list with respect to a desirability representing the benefit of selecting each element. The random component in the construction phase of the GRASP comes from the fact that we randomly choose one of the best candidates rather than the best one. For the particular case of the CAP, by not assigning the task with the largest difference between the two smallest values for the corresponding pseudo-cost function in a greedy fashion, but rather choosing the task to be assigned randomly among a list of candidates having the largest differences, a so-called GRASP can easily be constructed. Ramalhinho Lourenço and Serra [59] propose a GRASP for the GAP.

5.2.7 Meta-heuristics

In the last years, meta-heuristics have shown to be an adequate tool to find good solutions to combinatorial problems. Most of them are based on local search, i.e., given an initial solution to the combinatorial problem they move to a solution in its neighborhood. This procedure is repeated until a stopping criterion is satisfied. Meta-heuristics try to avoid stopping in a local optimum to the combinatorial problem by allowing movements where the objective function get worse. The main ideas behind these solutions procedures come from Artificial Intelligence, biological evolution, and statistical mechanisms. The most well-known meta-heuristics are Simulating Annealing, Tabu Search, Genetic Algorithm, and Variable Depth Search. This field is in continuous development yielding new solution procedures.

A vast literature is devoted to meta-heuristics for the GAP. Cattrysse [15] implements a simulating annealing algorithm, and concludes that it is only competitive for small problem sizes. Racer and Amini [58] describe a variable depth search heuristic. They compare their results with the heuristic from Martello and Toth [49] on five classes of problems. At the expense of high computation times, the variable depth search heuristic finds better feasible

solutions than the greedy heuristic for one of the problem classes. In order to decrease computation times, Amini and Racer [2] describe a hybrid heuristic where initial solutions are generated with the heuristic from Martello and Toth [49] and refined with a variable depth search heuristic. Osman [56] proposes a simulating annealing and a tabu search algorithm. Chu and Beasley [19] and Wilson [75] propose genetic algorithms for the GAP. In the first one, a family of potential solutions is generated, and steps are made to improve feasibility and optimality. On the contrary, good starting solutions in terms of objective value are assumed in the second one. Ramalhinho Lourenço and Serra [59] propose a MAX-MIN ant system (see also Stützle and Hoos [73]) combined with a local search and a tabu search schemes. Yagiura et al. [78] notice that searching only in the feasible region may be too restrictive. Therefore, they propose a variable depth search heuristic where it is allowed to move to infeasible solutions of the problem. Yagiura et al. [77] propose an ejection chain approach combined with a tabu search for the GAP.

5.3 Branch-and-Bound

A branch-and-bound scheme is an implicit enumerative procedure for (non-linear) (mixed) integer programming problems based on the concept of *divide-and-conquer*. The basic idea is to attempt to solve an optimization problem by partitioning its feasible region, thus creating a set of, often easier, subproblems. If necessary, this idea is repeated for each of the subproblems. A so-called *branch-and-bound tree* is created by associating a node with each problem to be solved, and creating arcs from a particular problem to all of its subproblems. It is clear that the best solution among all optimal solutions to the subproblems is the optimal solution to the original problem.

If we can solve a particular subproblem to optimality, or if we can guarantee that its solution is worse than the optimal solution to the original problem, we *prune* the corresponding node, i.e., we will not consider any further subproblems of this problem. Whether a node can be pruned can be established by computing a *lower bound* (for minimization problems) on its optimal solution value. If this lower bound is at least equal to the value of a feasible solution to the original problem, then the node can be pruned. If the lower bound corresponds to a feasible solution to the original problem, we may in addition be able to improve the current upper bound.

A general branch-and-bound procedure consists of the following elements:

(i) An *upper bounding* procedure, usually in the form of a heuristic;

(ii) a *lower bounding* procedure;

(iii) a strategy for creating subproblems, called a *branching strategy;*

(iv) an order in which the subproblems are considered, called a *searching strategy.*

If the upper bound is simply set to ∞ and the lower bounds are all set to $-\infty$, branch-and-bound simply reduces to complete enumeration of the feasible region. It is therefore clear that good bounding procedures are essential ingredients of a successful branch-and-bound procedure.

For the case of the CAP, we can in principle use any heuristic described in Section 5.2 and any lower bounding procedure described in Section 4 to construct a branch-and-bound procedure. The branching strategy may for instance be to choose a particular assignment variable, say x_{ij}, and creating two subproblems, in which $x_{ij} = 0$ and $x_{ij} = 1$ respectively. Finally, the most often used search strategies are depth-first search, breath-first search, and best-first search. In depth-first-search, if the node currently inspected is not pruned we choose one of its descendants as the next node. If it is pruned, we backtrack up the branch-and-bound tree until we find the first node for which not all subproblems have been considered. In breadth-first-search, we inspect all nodes at a given level in the branch-and-bound tree before proceeding to nodes at a lower level. Finally, in best-first-search, we define a measure with all nodes, and the nodes are visited in decreasing order of this measure. For example, we visit the node with the largest lower bound first.

Based on their empirical performance on the GAP, we have decided to present a few branch-and-bound algorithms from the literature in some detail. The good performance shown for the GAP suggests that these approaches may yield similar performance for the CAP.

The earliest branch-and-bound algorithm for the GAP is due to Ross and Soland [67]. They use the lower bounding procedure described at the end of Section 4.4.2, which is based on ignoring the the capacity constraints. Martello and Toth [49] illustrate empirically that the algorithm from Ross and Soland is not fast enough when the capacity constraints are tight. They propose, for a maximization formulation of the GAP, a branch-and-bound procedure where bounds are calculated by leaving out the semi-assignment constraints, see Section 4.4.3. To avoid a trivial bound for the case of a minimization formulation, we would consider the Lagrange bound for a set of

equal and negative Lagrange multipliers (ensuring that the resulting objective coefficients are negative for all assignment variables). Recall that this relaxed problem decomposes into m Knapsack Problems. In general, the optimal solution vector of this relaxation will violate the semi-assignment constraints for some of the tasks, i.e., there will be tasks assigned to more than one agent or tasks that are not assigned at all. With the purpose of improving the lower bound and developing a branching rule, a penalty for violating the semi-assignment constraints is introduced. If the node is not pruned, we branch on the task with the highest penalty. If in the relaxed solution this task was not assigned at all, we create m new subproblems, one for the assignment of this task to each of the m agents. If the task is assigned to more than one agent, say $i_1, i_2, \ldots, i_{m'}$, m' subproblems are created. In the first $m' - 1$ subproblems, the constraint that the task is assigned to subset of $m' - 1$ of the previously mentioned agents is imposed. In the last subproblem, the constraint that the agent is not assigned to any of the m' agents is imposed. This algorithm is frequently used in the literature for comparative purposes.

Fisher et al. [26] propose a branch-and-bound procedure where the lower bounds are given by the Lagrangean relaxation of the semi-assignment constraints (see Section 4.4.3). If a particular node is not pruned, the solution vector obtained from the optimization of the Lagrangean function in that node is used to create two subproblems, associated with each of the two possible values of the assignment variable x_{ij} that has not been fixed to zero or one in this branch and has the largest value of a_{ij}. For a general CAP, this would translate to choosing the assignment variable x_{ij} that has not been fixed to zero or one in this branch and has the largest value of $\frac{\partial \omega_L}{\partial x_{ij}}\Big|_{x=\bar{x}}$ where $\bar{x}$ is the solution of the Lagrange relaxation. This branch-and-bound procedure is compared numerically with the ones proposed by Ross and Soland [67] and Martello and Toth [49], showing that for more difficult problems they outperform these two. Guignard and Rosenwein [37] observe that the largest test problems reported in the literature until that moment contained at most 100 variables. They solve the GAP using a branch-and-bound procedure where the bounding technique is similar to the one used by Fisher et al. [26] (see Section 4.4.3), and are able to solve problems with 500 variables.

5.4 Branch-and-Price

The set partitioning formulation of the CAP that was introduced in Section 4.3 lends itself to a branch-and-price approach to the CAP. In short, a

branch-and-price scheme solves the set partitioning formulation (SP) for the CAP to optimality by a branch-and-bound scheme, where the bounds are obtained by solving the LP-relaxations of the subproblems by a Column Generation procedure.

If the optimal solution of the LP-relaxation of (SP) is not integer we need to branch to obtain an optimal integer solution. Since the LP-relaxation of (SP) has been solved by column generation, it is unlikely that all columns are present in the final linear programming problem. If we want a certificate of optimality for the integer programming problems, new columns (when needed) should be generated when branching. The choice of the branching rule is crucial since it can destroy the structure of the pricing problem. The straightforward choice would be to branch on the decision variables y_i^ℓ corresponding to (SP). Fixing one of those variables to zero is equivalent to prohibiting the use of that column. In terms of the pricing problem, this means that we may need to find the second best solution rather than the optimal solution. Usually this renders the pricing problem much harder to solve. However, each feasible solution $\boldsymbol{y}$ of (SP) has a corresponding feasible solution $\boldsymbol{x}$ in the original formulation of the CAP. Moreover, if $\boldsymbol{y}$ is fractional then $\boldsymbol{x}$ is fractional as well. Thus, we can branch on the fractional variables x_{ij}. We may observe that the subproblems obtained by branching on the x_{ij} variables are again convex assignment problems. Therefore, the column generation procedure in each node of the branch-and-bound tree is the same as in the root node.

Savelsbergh [69] proposes this branch-and-price scheme for the GAP, and proves that the pricing problem turns out to be a Knapsack Problem for the GAP.

6 Testing Solution Procedures

6.1 Introduction

The behaviour of a solution procedure for an optimization problem is frequently illustrated by testing it on a collection of problem instances. Conclusions about relevant characteristics of the solution procedure are drawn from such a study. A more thorough analysis may be performed by comparing the behaviour of the solution procedure, with respect to these characteristics, to other solution procedures on the same collection of problem instances. Typical examples of such characteristics are the computation time required by the solution procedure and the quality of the solution obtained (in case

finding an optimal solution is not guaranteed). The validity of the derived conclusions strongly depends on the set of problem instances chosen for this purpose. Therefore, the collection of test problems should possess certain properties to ensure the credibility of the conclusions.

Hall and Posner [38] study test problem generations for several classes of scheduling problems. They suggest a list of desirable properties of a collection of test problems which applies to most types of deterministic mathematical programming problems. This list includes properties like robustness, consistency, and convenience. The first one requires a vast and diverse generation of test problems modeling real-life situations. This property is of special relevance when comparing the behaviour of several solution procedures. The consistency refers to invariance of the characteristics of the data generation scheme with respect to the size and the scale of the input data. The third property ensures that problem instances are easy to generate. An additional issue included by the authors under this property is the feasibility of the problem instances. Due to the *NP-Completeness* of the CAP, this property is of special interest for this problem. Therefore, we devote this section to elaborate on this issue for the CAP.

Generally, the availability of real data for problem instances is very limited. Since a reduced number of problem instances can bias the conclusions drawn about the behaviour of the solution procedure, it is often desirable to generate artificial problem instances. In the literature, randomly generated problem instances are frequently used. However, the properties of the problem instances obtained by these random generators are rarely studied. In this section we present a stochastic model for the CAP and analyze the tightness of the problem instances generated by this model. For the particular case of the GAP, this stochastic model is general enough to basically cover all the random generators proposed in the literature for the GAP.

In the following we will restrict ourselves to the subclass of CAPs called *Convex Capacitated Assignment Problems (CCAP)* where the capacity constraints are linear, i.e., $\omega_i(x_{i\cdot}) = A^i x_{i\cdot}$, for all $i = 1, \ldots, m$, where $A^i \in \mathcal{M}_{k_i \times n}$ is a nonnegative matrix (see Romero Morales [66] for an extensive study of this class of problems). Notice that the GAP and almost all of the extensions discussed in Section 2.2.1 are of this form. In the remainder, we will write the matrix A^i in terms of its columns, i.e., $A^i = (A^i_1 | \ldots | A^i_n)$ where $A^i_j \in \mathbb{R}^{k_i}$ for each $j = 1, \ldots, n$.

6.2 Generating Experimental Data

6.2.1 The CAP

Probabilistic analysis is a powerful tool when designing an appropriate model for generating random problem instances. Performing a feasibility analysis yields a suitable probabilistic model that can be used for randomly generating experimental data for the problem, with the property that the problem instances are asymptotically feasible with probability one. In the literature we mainly find probabilistic analyses of the optimal value of optimization problems. Such analyses have been performed for a large variety of problems, starting with the pioneering paper by Beardwood et al. [8] on a probabilistic analysis of Euclidean TSPs, spawning a vast number of papers on the probabilistic analysis of various variants of the TSP and the VRP (see Bramel and Simchi-Levi [12] for an overview).

Since we will focus in this section on the issue of feasibility of random instances generated by a probabilistic model, we will omit here the parameters defining the objective function. Note that this does not preclude correlations between parameters appearing in the objective function and the feasible region. Romero Morales [66] proposes the following probabilistic model for the parameters defining the feasible region of the CCAP. Let the random vectors $\boldsymbol{A}_j = ((\boldsymbol{A}_j^1)^\top, \ldots, (\boldsymbol{A}_j^m)^\top)^\top$ $(j = 1, \ldots, n)$ be i.i.d. in the bounded set $[\underline{A}, \overline{A}]^{k_1} \times \ldots \times [\underline{A}, \overline{A}]^{k_m}$ where $\underline{A}$ and $\overline{A} \in \mathbb{R}_+$. Generalizing an often used class of probabilistic models for instances for the GAP, the capacity of agent i, say $\boldsymbol{b}_i$, is then defined equal to

$$\boldsymbol{b}_i = \delta \left(\alpha_1 \, \mu_i + \alpha_2 \sum_{j=1}^{n} \boldsymbol{A}_j^i / n \right) \, n/m$$

where δ is a strictly positive number, and can be viewed as a measure of the tightness of the capacity constraints. Furthermore, α_1 and α_2 are nonnegative and $\alpha_1 + \alpha_2 = 1$, and $\mu_i \in \mathbb{R}_+^{k_i}$ for all $i = 1, \ldots, m$. In this model, the values of m and k_i, $i = 1, \ldots, m$, will be considered fixed, leaving the number of tasks as a measure of the size of the problem instances. Even though the requirements must be identically distributed, by considering the appropriate mixture of distribution functions this stochastic model is suitable to model a situation in which there are several types of tasks as well.

The following result gives an important characterization of the models yielding (mostly) feasible instances for the CCAP. In the following, $\mathcal{E}(X)$ represents the expected value of the random variable X.

Theorem 6.1 (cf. Romero Morales [66]) *The CCAP is feasible with probability one as* $n \to \infty$, *if the* excess capacity

$$\Delta \equiv \min_{\lambda \in S} \left(\delta/m \sum_{i=1}^{m} \lambda_i^\top (\alpha_1 \mu_i + \alpha_2 \mathcal{E}(A_1^i)) - \mathcal{E}\left(\min_{i=1,\ldots,m} \lambda_i^\top A_1^i \right) \right) > 0,$$

where S is the unit simplex in $\mathbb{R}^{k_1} \times \ldots \times \mathbb{R}^{k_m}$, *and infeasible with probability one if* $\Delta < 0$. □

This result yields an implicit condition that ensures asymptotic feasibility in the probabilistic sense involving, in general, the minimization of a nonlinear function. In the following sections, we will present some examples of CCAPs where the condition $\Delta > 0$ reduces to an explicit lower bound on the measure of tightness δ that is a function of the other parameters of the probabilistic model.

6.2.2 The GAP

Stochastic models for the GAP have been proposed by Dyer and Frieze [21], Romeijn and Piersma [60], and Romeijn and Romero Morales [62]. Dyer and Frieze [21] perform a limited probabilistic analysis, with the purpose of constructing an algorithm that solves the GAP with high probability in polynomial time (in the number of tasks). Romeijn and Piersma [60] propose for the GAP the stochastic model we have presented above for the CCAP with deterministic right-hand sides. Using empirical processes they derive the condition on the excess capacity given in Theorem 6.1 for the GAP. Furthermore, a probabilistic analysis of the optimal solution for the GAP under this model is performed, studying the asymptotic behaviour of the optimal solution value.

In the literature on algorithmic approaches to the GAP, a consensus seems to have developed on the use of a set of 4 probabilistic models, dubbed Type A-D by Martello and Toth [49]:

A. The costs and the requirements are uniformly distributed in [10,50] and [5,25] respectively, and the capacities are set to

$$b_i^{\mathrm{A}} = \delta \left(0.6 \cdot 15\, n/m + 0.4 \max_{i=1,\ldots,m} \sum_{j \in J_i^*} a_{ij} \right) \tag{6}$$

where J_i^* is defined as the set of tasks for which agent i is the cheapest one, i.e.,

$$J_i^* = \{j = 1, \ldots, n : i = \arg \min_{s=1,\ldots,m} c_{sj}\}$$

where ties are broken arbitrarily. The tightness parameter δ is chosen equal to 1.

B. The costs and the requirements are generated in the same way as in Type A, and the capacities are set to 70 percent of the ones generated by Type A, i.e., $\delta = 0.7$ in (6).

C. Again the costs and the requirements are generated in the same way as in Type A, and the capacities are set to

$$b_i^C = \delta \sum_{j=1}^{n} a_{ij}/m$$

where the tightness parameter is equal to 0.8.

D. This model introduces a correlation between the requirements and costs. The requirements are uniformly generated in [1, 100] and the costs are defined as $c_{ij} = 111 - a_{ij} + u_{ij}$, where u_{ij} is uniformly generated in (−10,10). The capacities are set as in Type C, i.e.,

$$b_i^D = \delta \sum_{j=1}^{n} a_{ij}/m$$

where $\delta = 0.8$.

See Amini and Racer [2], Cattrysse et al. [16], Chu and Beasley [19], Fisher et al. [26], Guignard and Rosenwein [37], Lorena and Narciso [48], Osman [56], Racer and Amini [58], and Savelsbergh [69] for studies involving these probabilistic models. Romeijn and Romero Morales [62] derive conditions under which each of these probabilistic models are asymptotically feasible with probability one. More importantly however, they show that all of these models have the property that the capacities become less tight as the number of agents, m, increases. This is clearly an unexpected and undesirable property of these models. Figure 1 gives us an impression on the tightness of the problem instances generated through models A-C by showing the measure of tightness for these models, as well as a lower bound on

this measure that needs to be satisfied in order to obtain problem instances that are feasible with probability one, which is given by

$$\underline{\delta}^{\mathrm{A-C}}(m) = \frac{5m+25}{15(m+1)}.$$

Similarly, Figure 2 illustrates the tightness of the problem instances generated through model D comparing with the corresponding lower bound

$$\underline{\delta}^{\mathrm{D}}(m) = \frac{2(m+100)}{101(m+1)}.$$

These figures show that, and even how, the tightness parameter δ should depend on the number of agents m. We therefore propose that the tightness parameters of the models A-D in the literature are replaced by

$$\gamma\,\underline{\delta}(m)$$

with $\gamma > 1$. In particular, that means that we suggest the following capacities:

$$b_i^1 = \gamma\,\frac{5m+25}{15(m+1)}\left(0.6\cdot 15\,n/m + 0.4\max_{i=1,\ldots,m}\sum_{j\in J_i^*} a_{ij}\right)$$

for Types A and B,

$$b_i^2 = \gamma\,\frac{5m+25}{15(m+1)}\sum_{j=1}^{n} a_{ij}/m$$

for type C, and

$$b_i^3 = \gamma\,\frac{2(m+100)}{101(m+1)}\sum_{j=1}^{n} a_{ij}/m$$

for Type D, all with $\gamma > 1$.

6.3 Design of Experiments

Amini and Racer [1] claim that the most common way of illustrating the performance characteristics of a solution procedure is the use of tables and graphs where average results are presented. They argue that conclusions drawn about the behaviour of the solution procedure may not be correct if the testing procedure is not adequate. They present a design of experiments for computational comparison of several solution procedures for the GAP.

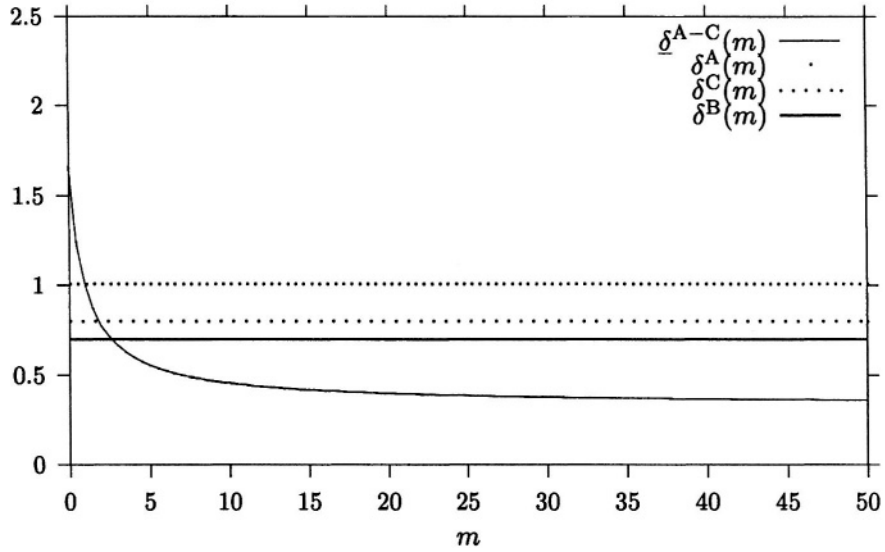


Figure 1: Tightness of the random generators A-C

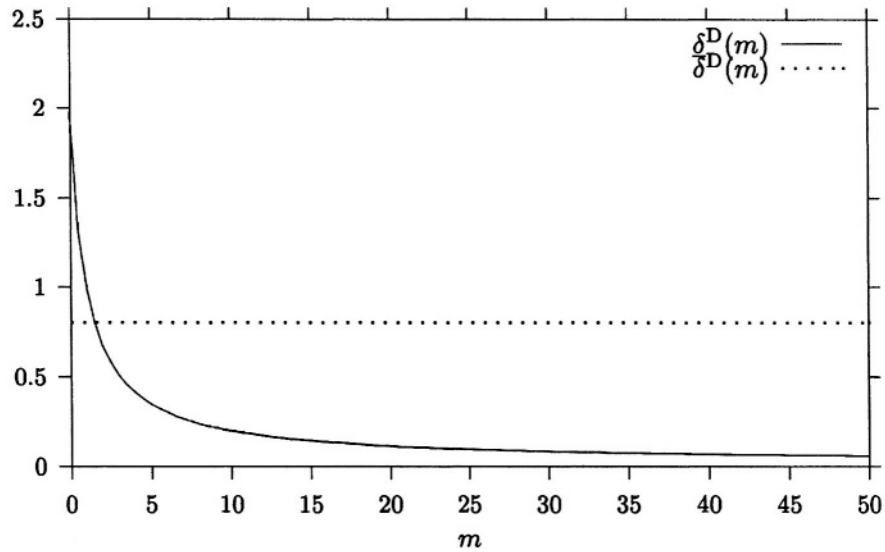


Figure 2: Tightness of the random generator D

They compare the branch-and-bound schemes from Ross and Soland [67], Martello and Toth [49], and Fisher et al. [26], the greedy heuristic of Martello and Toth [49], and a variable depth search heuristic which is apparently the same one as the authors have proposed in Racer and Amini [58]. The computation time is chosen as a dependent variable, and the problem class, problem size, and solution method as explanatory variables. They conclude that none of the solution methods outperforms the others.

7 Additional Approaches to the GAP

In this section we will summarize some results that have been obtained for the GAP that do not seem extendable to the more general CAP.

In Section 3 we discussed the $\mathcal{NP}$-*Hardness* of the GAP. To overcome this difficulty, Shmoys and Tardos [70] propose a polynomial-time algorithm for the GAP that, given some $C \in \mathbb{R}$, either proves that there is no feasible solution for the GAP with cost C or finds an assignment of tasks to agents that costs at most C, and where the consumption of the resource at agent i is at most $2b_i$ for all $i = 1, \ldots, m$.

Hallefjord et al. [39] use an aggregation/disaggregation technique for large scale GAPs. They use a column aggregation where the set of tasks is partitioned and a unique decision variable is associated with each subset and each agent. After disaggregating the obtained solution vector, they cannot ensure that all the tasks will be assigned. Therefore, they suggest some heuristics that can be used to reduce the feasibility violations in the disaggregated solution.

Gottlieb and Rao [34, 35] perform a polyhedral study of the GAP. It is straightforward to see that any valid inequality for the Knapsack Problem is also valid for the GAP; they also prove that each facet of the Knapsack Problem is also a facet for the GAP. In addition, they have derived other valid and facet-defining inequalities for the GAP based on more than one knapsack constraint.

8 The Multi-Resource GAP

The MRGAP is one of the most straightforward generalizations of the GAP, and (as was noted above) is clearly a member of the class of CAP. For this problem class, both the costs and the capacity constraints associated with

each agent are linear. The number of capacity constraints associated with each agent is equal to K. If $K = 1$, the problem reduces to the GAP.

Gavish and Pirkul [29] take advantage of the linearity of the objective function and the capacity constraints when looking for lower bounds for the MRGAP. They use out three different Lagrangean relaxations for the MRGAP. The first two are the relaxations discussed in Sections 4.4.2 and 4.4.3, in which the capacity constraints and the semi-assignment constraints are relaxed, respectively. As for the GAP, the former relaxed problem can be solved explicitly by simple inspection. In the latter relaxation, the relaxed problem is a Multi-Knapsack Problem. (See Martello and Toth [50] for an overview on solution procedures for the Multi-Knapsack Problem.) In addition, they propose a third Lagrangean relaxation where all the semi-assignment constraints and all the capacity constraints except for the ones corresponding to a single resource (say k') are dualized, i.e.,

$$\text{minimize} \left(\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} + \sum_{j=1}^{n} \mu_j \left(\sum_{i=1}^{m} x_{ij} - 1 \right) + \sum_{k=1;k\neq k'}^{K} \sum_{i=1}^{m} \lambda_{ik} \left(\sum_{j=1}^{n} a_{ijk} x_{ij} - b_{ik} \right) \right)$$

subject to

$$\begin{aligned} \sum_{j=1}^{n} a_{ijk'} x_{ij} &\leq b_{ik'} && i = 1, \ldots, m \\ x_{ij} &\in \{0,1\} && i = 1, \ldots, m;\ j = 1, \ldots, n \end{aligned}$$

where $\mu \in \mathbb{R}^n$ and $\lambda \in \mathbb{R}_+^{m(K-1)}$. Observe that this problem decomposes into m Knapsack Problems. Again the best lower bound is obtained by maximizing over the set of vectors $(\mu, \lambda) \in \mathbb{R}^n \times \mathbb{R}_+^{m(K-1)}$, and over the unrelaxed resource k'. Gavish and Pirkul prove that the Lagrangean relaxation of the capacity constraints performs at least as good as their proposed relaxation, which in turn is at least as good as the Lagrangean relaxation of the semi-assignment constraints only. They propose a subgradient method to search for a good set of Lagrange multipliers. To obtain primal solutions for the MRGAP they suggest two heuristics which modify the solution obtained in the first and second Lagrangean relaxations respectively, as well as a greedy heuristic which is a variant of the one described in Section 5.2.5.

In this greedy heuristic the desirability of assigning task j is defined as

$$\frac{c_{ij}}{\sum_{k=1}^{K} a_{ijk}} - \frac{c_{i_j^2 j}}{\sum_{k=1}^{K} a_{i_j^2 jk}}$$

where $c_{ij} \leq c_{ij}$ for all $i = 1, \ldots, m$ and $c_{i_j^2 j} \leq c_{ij}$ for all $i = 1, \ldots, m$ and $i \neq i_j$. (Note that there exists, in general, no pseudo-cost function yielding these desirabilities.) Gavish and Pirkul use the lower bounds and the primal heuristics in a branch-and-bound scheme for the MRGAP.

Blocq et al. [11] have extended the MRGAP by including constraints on the consumption of each resource by each subset of agents. As done for the GAP by Romeijn and Romero Morales [61], they propose a family of pseudo-cost functions for the greedy heuristic described in Section 5.2.5. In the particular case of the MRGAP, this family is equal to

$$f_\lambda(i, j) = c_{ij} + \sum_{k=1}^{K} \lambda_{ik} a_{ijk}$$

where $\lambda \in \mathbb{R}_+^{Km}$. Expecting asymptotic optimality of the greedy heuristic as for the GAP, they pay special attention to $\lambda = \lambda^*$ where λ^* is equal to the optimal dual subvector of the capacity constraints in the LP-relaxation of the MRGAP. Their computational results seem to support the conjecture of asymptotic feasibility and optimality of this greedy heuristic.

Branch-and-Price is one of the exact procedures described for the CAP, and is therefore applicable to the MRGAP as well. One of the main features to deal with is the structure of the pricing problem. A subset of tasks can be assigned to a particular agent if and only if its K capacity constraints are satisfied. Therefore, the pricing problem for the MRGAP is a Multi-Knapsack Problem.

9 The MPSSP

9.1 The Model

In this section we discuss a class of assignment problems that arise in the optimization of supply chains. In particular, these models are suitable for evaluating the performance of a logistics distribution network in a dynamic environment, taking both transportation and inventories into consideration. We restrict ourselves here to two-level multi-period single-sourcing problems

where production and storage take place at the same location and only the production capacity is constrained. See Romero Morales [66] for more details on the results presented in this section, and for extensions to this model. In particular, we will consider a problem with m facilities (each of which may, for instance, be interpreted as a production plant and its corresponding warehouse), n retailers, and a planning horizon of T periods. The demand of retailer j in period t is given by d_{jt}, while the production capacity at facility i in period t is equal to b_{it}. The problem is to assign each customer to a facility in each period, at minimum costs. The costs of assigning customer j to facility i in period t are c_{ijt}, and the costs of holding a unit of inventory at facility i in period t is equal to h_{it}. Although production costs are not explicitly mentioned here, it can be shown that production costs that are linear in the quantity produced can be incorporated in the model, by a suitable redefinition of the assignment and inventory costs. However, in this case the inventory costs may become negative, so that we will only assume that all parameters except the inventory costs are nonnegative.

The problem can now be formulated as follows, using the zero-one decision variables x_{ijt}, having the value 1 if retailer j is assigned to facility i in period t, and the nonnegative decision variables I_{it} denoting the inventory level at facility i at the end of period t:

$$\text{minimize} \sum_{t=1}^{T}\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ijt}x_{ijt} + \sum_{t=1}^{T}\sum_{i=1}^{m} h_{it}I_{it}$$

subject to (P)

$$\sum_{j=1}^{n} d_{jt}x_{ijt} + I_{it} \leq b_{it} + I_{i,t-1} \qquad i = 1,\ldots,m;\ t = 1,\ldots,T$$

$$\sum_{i=1}^{m} x_{ijt} = 1 \qquad j = 1,\ldots,n;\ t = 1,\ldots,T$$

$$x_{ijt} = x_{ij1} \qquad i = 1,\ldots,m;\ j \in \mathcal{S};\ t = 2,\ldots,T \tag{7}$$

$$I_{i0} = I_{iT}1_{\{i\in\mathcal{C}\}} \qquad i = 1,\ldots,m \tag{8}$$

$$x_{ijt} \in \{0,1\} \qquad i = 1,\ldots,m;\ j = 1,\ldots,n;\ t = 1,\ldots,T$$

$$I_{it} \geq 0 \qquad i = 1,\ldots,m;\ t = 1,\ldots,T.$$

Constraints (7) model the fact that customer service considerations may dictate that some or all customers are assigned to the same facility throughout

the planning horizon. In particular, the set $\mathcal{S} \subseteq \{1,\ldots,n\}$ of *static* customers contains the customers for which this is the case. When convenient, we let $\mathcal{D} = \{1,\ldots,n\}\backslash\mathcal{S}$ denote the remaining set of *dynamic* customers.

In typical problems of the form (P), the inventory level at the end of period T will be zero (assuming that the inventory costs are nonnegative). However, when the MPSSP is used for strategic purposes, it may be more reasonable to assume that the set of periods $1,\ldots,T$ represent a typical planning period in the future, which will repeat itself. In that case, the ending inventory levels need to be equal to the starting inventory levels. Letting the indicator function $1_{\{Q\}}$ take on the value 1 if statement Q is true, and 0 otherwise, constraints (8) model this situation, where $\mathcal{C}$ is the set of *cyclic* warehouses for which the starting and ending inventories need to be equal. It is clear that the only interesting and realistic cases are the two extremes $\mathcal{C} = \emptyset$ and $\mathcal{C} = \{1,\ldots,m\}$. Therefore, we will pay particular attention to these two cases. In order for the optimization problem (P) to be well-defined (i.e., for its optimal value to be bounded from below), we need to assume that $\sum_{t=1}^{T} h_{it} \geq 0$ for each $i \in \mathcal{C}$.

9.2 A CAP Formulation

In this section we will show that the MPSSP introduced above is in fact a CAP, so that many of the techniques described above can be applied to this problem. In particular, we reformulate (P) in terms of the assignment variables x_{ijt} only, by eliminating the inventory variables:

$$\text{minimize} \sum_{t=1}^{T}\sum_{i=1}^{m}\sum_{j=1}^{n} c_{ijt}x_{ijt} + \sum_{i=1}^{m} H_i(x_{i\cdot\cdot})$$

subject to

$$\begin{aligned} \sum_{i=1}^{m} x_{ijt} &= 1 && j = 1,\ldots,n;\ t = 1,\ldots,T \\ x_{ijt} &= x_{ij1} && i = 1,\ldots,m;\ j \in \mathcal{S};\ t = 2,\ldots,T \\ x_{ijt} &\in \{0,1\} && i = 1,\ldots,m;\ j = 1,\ldots,n;\ t = 1,\ldots,T \end{aligned} \tag{9}$$

where the function $H_i(z)$, $z \in \mathbb{R}_+^{nT}$, denotes the minimal inventory costs for facility i corresponding to the assignment of customers according to the vector z, and can be computed as follows:

$$\text{minimize} \sum_{t=1}^{T} h_{it}I_t$$

subject to (P′)

$$\begin{aligned} I_t - I_{t-1} &\leq b_{it} - \sum_{j=1}^{n} d_{jt} z_{jt} && t = 1, \ldots, T \\ I_0 &= I_T 1_{\{i \in \mathcal{C}\}} \\ I_t &\geq 0 && t = 1, \ldots, T. \end{aligned}$$

Given a feasible set of assignments $x \in \mathbb{R}^{mnT}$, the capacities of facility i are sufficient to supply the corresponding demands if $H_i(x_{i\cdot\cdot}) < \infty$. The function H_i is unbounded for assignment vectors for which the required demand cannot be feasibly supplied due to the capacity constraints. In particular, we have that $H_i(x_{i\cdot\cdot}) < \infty$ for some $x \in \mathbb{R}_+^{mnT}$ if and only if

$$\begin{aligned} \sum_{t=1}^{T} \sum_{j=1}^{n} d_{jt} x_{ijt} &\leq \sum_{t=1}^{T} b_{it} \quad \text{if } i \in \mathcal{C} \\ \sum_{\tau=1}^{t} \sum_{j=1}^{n} d_{j\tau} x_{ij\tau} &\leq \sum_{\tau=1}^{t} b_{i\tau} \quad \text{for all } t = 1, \ldots, T \text{ if } i \notin \mathcal{C}. \end{aligned}$$

Moreover, it can be shown that the functions H_i are convex, as well as Lipschitz continuous. The latter property formalizes the intuitive idea that the optimal inventory holding costs corresponding to two assignment solutions that nearly coincide should not differ by very much.

We can now conclude that the MPSSP is a CAP, provided that we add the necessary and sufficient conditions for the finiteness of H_i as constraints to the reformulation (P′), and eliminate the variables x_{ijt} for $j \in \mathcal{S}$ and $t = 2, \ldots, T$ as well as the corresponding constraints (9).

In terms of the CAP, this reformulation means that we have two types of agents and two types of tasks. The agents are the cyclic facilities (each facing a single resource capacity constraint) and the acyclic facilities (each facing a set of T resource capacity constraints). The tasks are the assignments of the static customers $j \in \mathcal{S}$ to facilities, and the assignments of the (customer,period)-pairs (j,t) for dynamic customers $j \in \mathcal{D}$. Note that if we consider the case of cyclic facilities and static customers, i.e., $\mathcal{C} = \{1, \ldots, m\}$ and $\mathcal{S} = \{1, \ldots, n\}$, we obtain a GAP with convex objective function.

9.3 Results on the MPSSP

In this section we summarize the results derived for the MPSSP in the literature. Romeijn and Romero Morales [63, 64, 65] propose the stochastic

model given in Section 6.2 for the particular case of the MPSSP. They probabilistically analyze the feasibility of the problem instances for the MPSSP generated by this stochastic model, as well as the optimal solution value. The former analysis yields a special case of the implicit condition given in Theorem 6.1. An explicit condition to ensure asymptotic feasibility in the probabilistic sense for the cyclic case, as well as for subclasses of the acyclic case, have been obtained. These conditions can be summarized as follows: a random instance to (P') is asymptotically feasible with probability one as $n \to \infty$ if

(i) $\mathcal{C} = \{1, \ldots, m\}$ and

$$\sum_{t=1}^{T} \mathcal{E}(\boldsymbol{D}_{1t}) < \sum_{t=1}^{T} \sum_{i=1}^{m} \beta_{it}.$$

(ii) $\mathcal{C} = \emptyset$, one of the following conditions holds:

(a) $\mathcal{S} = \emptyset$;

(b) all facilities are identical, i.e., $\beta_{it} = \beta_t$ for all $i = 1, \ldots, m$ and all $t = 1, \ldots, T$,

and

$$\sum_{\tau=1}^{t} \mathcal{E}(\boldsymbol{D}_{1\tau}) < \sum_{\tau=1}^{t} \sum_{i=1}^{m} \beta_{i\tau} \quad \text{for } t = 1, \ldots, T.$$

(iii) $\mathcal{C} = \emptyset$, $\mathcal{S} = \{1, \ldots, n\}$, all customers have the same demand pattern, i.e., $\boldsymbol{D}_{jt} = \sigma_t \boldsymbol{D}_j$ for $t = 1, \ldots, T$, and

$$\sum_{i=1}^{m} \min_{t=1,\ldots,T} \left(\frac{\sum_{\tau=1}^{t} \beta_{i\tau}}{\sum_{\tau=1}^{t} \sigma_\tau} \right) > \mathcal{E}(\boldsymbol{D}_1).$$

Note that the conditions on the parameters in case (ii-b) reduce to

$$m \cdot \sum_{\tau=1}^{t} \beta_\tau > \sum_{\tau=1}^{t} \mathcal{E}(\boldsymbol{D}_{1\tau}) \qquad t = 1, \ldots, T.$$

In all cases, the random problem instances are asymptotically infeasible with probability one if one of the inequalities is reversed.

For solving the MPSSP, Romeijn and Romero Morales [64, 65] propose the following family of pseudo-cost functions for the greedy heuristic given in Section 5.2.5:

$$f(i,\ell) = \begin{cases} \sum_{t=1}^{T}(c_{ijt} + \lambda_{it}d_{jt}) & \text{if } \ell = j \in \mathcal{S} \\ c_{ijt} + \lambda_{it}d_{jt} & \text{if } \ell = (j,t);\ j \in \mathcal{D} \text{ and } t = 1,\ldots,T \end{cases}$$

where $\lambda \in \mathbb{R}_{+}^{mT}$. Special attention is paid to $\lambda = \lambda^*$, where λ^* is the optimal dual subvector corresponding to the capacity constraints in the LP-relaxation of (P). Asymptotic feasibility and optimality in the probabilistic sense of the greedy heuristic has been shown on problem instances of the MPSSP where all facilities exhibit a cyclic inventory pattern, as well as for the acyclic case with only dynamic customers, and the acyclic case with static customers whose demand patterns exhibit the same seasonality pattern.

A branch-and-price algorithm for solving the MPSSP, like the one outlined in Section 5.4, is proposed in Freling et al. [27]. As mentioned before, the structure of the pricing problem is a major issue in the success of the column generation procedure. It turns out that the pricing problem for the acyclic case with static customers and seasonal demand patterns can be formulated as a so-called Penalized Knapsack Problem (PKP), where a convex penalty is imposed on using of the resource. Their computational results reveal that the branch-and-price is very well suited for solving this particular variant of the MPSSP, especially when the ratio between the number of customers and the number of facilities is not too large.

10 Concluding Remarks

In this chapter we have described the state of the art in solving the Generalized Assignment Problem, as well as many extensions thereof. The approach we have taken is to generalize the GAP to a much larger class of Convex Assignment Problems, show that many of the extensions of the GAP proposed in the literature are members of this class, and describe many of the proposed solution approaches to the GAP in terms of the larger class of problems. Throughout the chapter we have paid particular attention to the Generalized Assignment Problem, the Multi-Resource Generalized Assignment Problem, and the Multi-Period Single-Sourcing Problem.

References

[1] M.M. Amini and M. Racer. A rigorous computational comparison of alternative solution methods for the generalized assignment problem. *Management Science,* 40(7):868–890, 1994.

[2] M.M. Amini and M. Racer. A hybrid heuristic for the generalized assignment problem. *European Journal of Operational Research,* 87:343–348, 1995.

[3] V. Balachandran. An integer generalized transportation model for optimal job assignment in computer networks. *Operations Research,* 24(4):742–759, 1976.

[4] E. Balas and E. Zemel. Facets of the knapsack polytope from minimal covers. *SIAM Journal of Applied Mathematics,* 34:119–148, 1978.

[5] P. Barcia. The bound improving sequence algorithm. *Operations Research Letters,* 4(1):27–30, 1985.

[6] P. Barcia and K. Jörnsten. Improved Lagrangean decomposition: An application to the generalized assignment problem. *European Journal of Operational Research,* 46:84–92, 1990.

[7] C. Barnhart, E.L. Johnson, G.L. Nemhauser, M.W.P. Savelsbergh, and P.H. Vance. Branch-and-price: column generation for solving huge integer programs. *Operations Research,* 46(3):316–329, 1998.

[8] J. Beardwood, J.H. Halton, and J.M. Hammersley. The shortest path through many points. *Proceedings of the Cambridge Philosophical Society,* 55:299–327, 1959.

[9] J.F. Benders and J.A.E.E. van Nunen. A property of assignment type mixed integer linear programming problems. *Operations Research Letters,* 2(2):47–52, 1983.

[10] R.J.M. Blocq. Het multi-resource generalised assignment problem toegepast op de toeleveringsstrategie van een oliemaatschappij. Master's thesis, Vrije Universiteit Amsterdam, 1999.

[11] R.J.M. Blocq, D. Romero Morales, H.E. Romeijn, and G.T. Timmer. The multi-resource generalized assignment problem with an application

to the distribution of gasoline products. Working Paper, Department of Decision and Information Sciences, Rotterdam School of Management, The Netherlands, 2000.

[12] J. Bramel and D. Simchi-Levi. *The Logic of Logistics – theory, algorithms, and applications for logistics management.* Springer-Verlag, New York, 1997.

[13] J.F. Campbell and A. Langevin. The snow disposal assignment problem. *Journal of the Operational Research Society,* 46:919–929, 1995.

[14] D. Cattrysse, Z. Degraeve, and J. Tistaert. Solving the generalised assignment problem using polyhedral results. *European Journal of Operational Research,* 108:618–628, 1998.

[15] D.G. Cattrysse. *Set Partitioning Approaches to Combinatorial Optimization Problems.* PhD thesis, Katholieke Universiteit Leuven, 1990.

[16] D.G. Cattrysse, M. Salomon, and L.N. Van Wassenhove. A set partitioning heuristic for the generalized assignment problem. *European Journal of Operational Research,* 72:167–174, 1994.

[17] D.G. Cattrysse and L.N. Van Wassenhove. A survey of algorithms for the generalized assignment problem. *European Journal of Operational Research,* 60:260–272, 1992.

[18] L. Chalmet and L. Gelders. Lagrangean relaxation for a generalized assignment–type problem. In M. Roubens, editor, *Advances in OR,* pages 103–109. EURO, North–Holland, Amsterdam, 1976.

[19] P.C. Chu and J.E. Beasley. A genetic algorithm for the generalised assignment problem. *Computers and Operations Research,* 24(1):17–23, 1997.

[20] A. De Maio and C. Roveda. An all zero-one algorithm for a certain class of transportation problems. *Operations Research,* 19(6):1406–1418, 1971.

[21] M. Dyer and A. Frieze. Probabilistic analysis of the generalised assignment problem. *Mathematical Programming,* 55:169–181, 1992.

[22] T.A. Feo and M.G.C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization,* 6:109–133, 1995.

[23] J.A. Ferland, A. Hertz, and A. Lavoie. An object-oriented methodology for solving assignment-type problems with neighborhood search techniques. *Operations Research,* 44(2):347–359, 1996.

[24] M.L. Fisher. The lagrangian relaxation method for solving integer programming problems. *Management Science,* 27:1–18, 1981.

[25] M.L. Fisher and R. Jaikumar. A generalized assignment heuristic for vehicle routing. *Networks,* 11:109–124, 1981.

[26] M.L. Fisher, R. Jaikumar, and L.N. Van Wassenhove. A multiplier adjustment method for the generalized assignment problem. *Management Science,* 32(9):1095–1103, 1986.

[27] R. Freling, H.E. Romeijn, D. Romero Morales, and A.P.M. Wagelmans. A branch and price algorithm for the multi-period single-sourcing problem. *Operations Research,* 51(6):922–939, 2003.

[28] R.S. Garfinkel and G.L. Nemhauser. Set partitioning problem: set covering with equality constraints. *Operations Research,* 17:848–856, 1969.

[29] B. Gavish and H. Pirkul. Algorithms for the multi-resource generalized assignment problem. *Management Science,* 37(6):695–713, 1991.

[30] A.M. Geoffrion. Lagrangean relaxation for integer programming. *Mathematical Programming Studies,* 2:82–114, 1974.

[31] A.M. Geoffrion and G.W. Graves. Multicommodity distribution system design by Benders decomposition. *Management Science,* 20(5):822–844, 1974.

[32] F. Glover. Surrogate constraints. *Operations Research,* 16(4):741–749, 1968.

[33] F. Glover, J. Hultz, and D. Klingman. Improved computer-based planning techniques, part II. *Interfaces,* 9(4):12–20, 1979.

[34] E.S. Gottlieb and M.R. Rao. $(1,k)$-configuration facets for the generalized assignment problem. *Mathematical Programming,* 46:53–60, 1990.

[35] E.S. Gottlieb and M.R. Rao. The generalized assignment problem: Valid inequalities and facets. *Mathematical Programming,* 46:31–52, 1990.

[36] M. Guignard and S. Kim. Lagrangean decomposition: a model yielding stronger lagrangean bounds. *Mathematical Programming,* 39:215–228, 1987.

[37] M. Guignard and M.B. Rosenwein. An improved dual based algorithm for the generalized assignment problem. *Operations Research,* 37(4):658–663, 1989.

[38] N.G. Hall and M.E. Posner. Generating experimental data for scheduling problems. *Operations Research,* 49(6):854–865, 2001.

[39] Å. Hallefjord, K.O. Jörnsten, and P. Värbrand. Solving large scale generalized assignment problems- An aggregation/disaggregation approach. *European Journal of Operational Research,* 64:103–114, 1993.

[40] M. Held, P. Wolfe, and H.P. Crowder. Validation of subgradient optimization. *Mathematical Programming,* 6:62–88, 1974.

[41] K. Jörnsten and M. Näsberg. A new lagrangian relaxation approach to the generalized assignment problem. *European Journal of Operational Research,* 27:313–323, 1986.

[42] K.O. Jörnsten and P. Värbrand. A hybrid algorithm for the generalized assignment problem. *Optimization,* 22(2):273–282, 1991.

[43] N. Karabakal, J.C. Bean, and J.R. Lohmann. A steepest descent multiplier adjustment method for the generalized assignment problem. Technical Report 92-11, Department of Industrial and Operations Engineering, The University of Michigan, 1992.

[44] T.D. Klastorin. On the maximal covering location problem and the generalized assignment problem. *Management Science,* 25(1):107–113, 1979.

[45] K. Kogan, A. Shtub, and V.E. Levit. DGAP–the dynamic generalized assignment problem. *Annals of Operations Research,* 69:227–239, 1997.

[46] H. Kuhn. A heuristic algorithm for the loading problem in flexible manufacturing systems. *International Journal of Flexible Manufacturing Systems,* 7:229–254, 1995.

[47] M. Laguna, J.P. Kelly, J.L. González-Velarde, and F. Glover. Tabu search for the multilevel generalized assignment problem. *European Journal of Operational Research,* 82:176–189, 1995.

[48] L.A.N. Lorena and M.G. Narciso. Relaxation heuristics for a generalized assignment problem. *European Journal of Operational Research,* 91:600–610, 1996.

[49] S. Martello and P. Toth. An algorithm for the generalized assignment problem. In J.P. Brans, editor, *Operational Research '81,* pages 589–603. IFORS, North–Holland, Amsterdam, 1981.

[50] S. Martello and P. Toth. *Knapsack problems, algorithms and computer implementations.* John Wiley & Sons, New York, 1990.

[51] S. Martello and P. Toth. The bottleneck generalized assignment problem. *European Journal of Operational Research,* 83(3):621–638, 1995.

[52] J.B. Mazzola. Generalized assignment with nonlinear capacity interaction. *Management Science,* 35(8):923–941, 1989.

[53] J.B. Mazzola and A.W. Neebe. Bottleneck generalized assignment problems. *Engineering Costs and Production Economics,* 14:61–65, 1988.

[54] M.G. Narciso and L.A.N. Lorena. Lagrangean/surrogate relaxation for generalized assignment problems. *European Journal of Operational Research,* 114:165–177, 1999.

[55] A.W. Neebe and M.R. Rao. An algorithm for the fixed–charge assigning users to sources problem. *Journal of the Operational Research Society,* 34(11):1107–1113, 1983.

[56] I.H. Osman. Heuristics for the generalised assignment problem: simulated annealing and tabu search approaches. *OR Spektrum,* 17:211–225, 1995.

[57] J.S. Park, B.H. Lim, and Y. Lee. A lagrangian dual-based branch-and-bound algorithm for the generalized multi-assignment problem. *Management Science,* 44(12, part 2 of 2):S271–S282, 1998.

[58] M. Racer and M.M. Amini. A robust heuristic for the generalized assignment problem. *Annals of Operations Research,* 50:487–503, 1994.

[59] H. Ramalhinho Lourenço and D. Serra. Adaptive search heuristics for the generalized assignment problem. *Mathware and Soft Computing,* 9(2–3):209–234, 2002.

[60] H.E. Romeijn and N. Piersma. A probabilistic feasibility and value analysis of the generalized assignment problem. *Journal of Combinatorial Optimization,* 4(3):325–355, 2000.

[61] H.E. Romeijn and D. Romero Morales. A class of greedy algorithms for the generalized assignment problem. *Discrete Applied Mathematics,* 103:209–235, 2000.

[62] H.E. Romeijn and D. Romero Morales. Generating experimental data for the generalized assignment problem. *Operations Research,* 49(6):866–878, 2001.

[63] H.E. Romeijn and D. Romero Morales. A probabilistic analysis of the multi-period single-sourcing problem. *Discrete Applied Mathematics,* 112:301–328, 2001.

[64] H.E. Romeijn and D. Romero Morales. An asymptotically optimal greedy heuristic for the multi-period single-sourcing problem: the cyclic case. *Naval Research Logistics,* 50(5):412–437, 2003.

[65] H.E. Romeijn and D. Romero Morales. Asymptotic analysis of a greedy heuristic for the multi-period single-sourcing problem: the acyclic case. *Journal of Heuristics,* 10:5–35, 2004.

[66] D. Romero Morales. *Optimization Problems in Supply Chain Management.* PhD thesis, TRAIL Thesis Series nr. 2000/4 & ERIM PhD series Research in Management nr. 3, The Netherlands, 2000.

[67] G.T. Ross and R.M. Soland. A branch and bound algorithm for the generalized assignment problem. *Mathematical Programming,* 8:91–103, 1975.

[68] G.T. Ross and R.M. Soland. Modeling facility location problems as generalized assignment problems. *Management Science,* 24(3):345–357, 1977.

[69] M.W.P. Savelsbergh. A branch–and–price algorithm for the generalized assignment problem. *Operations Research,* 45(6):831–841, 1997.

[70] D.B. Shmoys and É. Tardos. An approximation algorithm for the generalized assignment problem. *Mathematical Programming,* 62:461–474, 1993.

[71] A. Shtub and K. Kogan. Capacity planning by a dynamic multi-resource generalized assignment problem (DMRGAP). *European Journal of Operational Research,* 105:91–99, 1998.

[72] V. Srinivasan and G.L. Thompson. An algorithm for assigning uses to sources in a special class of transportation problems. *Operations Research,* 21:284–295, 1972.

[73] T. Stützle and H. Hoos. The max-min ant system and local search for combinatorial optimization problems: Towards adaptive tools for combinatorial global optimization. In S. Voss, S. Martello, I.H. Osman, and C. Roucairol, editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization,* pages 313–329. Kluwer Academic Publishers, 1998.

[74] M.A. Trick. A linear relaxation heuristic for the generalized assignment problem. *Naval Research Logistics,* 39:137–151, 1992.

[75] J.M. Wilson. A genetic algorithm for the generalised assignment problem. *Journal of the Operational Research Society,* 48:804–809, 1997.

[76] J.M. Wilson. A simple dual algorithm for the generalised assignment problem. *Journal of Heuristics,* 2:303–311, 1997.

[77] M. Yagiura, T. Ibaraki, and F. Glover. An ejection chain approach for the generalized assignment problem. Technical Report 99013, Department of Applied Mathematics and Physics, Graduate School of Informatics, Kyoto Univeristy, 1999.

[78] M. Yagiura, T. Yamaguchi, and T. Ibaraki. A variable depth search algorithm for the generalized assignment problem. In S. Voss, S. Martello, I.H. Osman, and C. Roucairol, editors, *Meta-Heuristics: Advances and Trends in Local Search Paradigms for Optimization,* pages 459–471. Kluwer Academic Publishers, 1998.

[79] V.A. Zimokha and M.I. Rubinshtein. R & D planning and the generalized assignment problem. *Automation and Remote Control,* 49:484–492, 1988.

HANDBOOK OF COMBINATORIAL OPTIMIZATION (VOL 5)
D.-Z. Du and P.M. Pardalos (Eds.) pp. 313-327

Optimal Rectangular Partitions

Xiuzhen Cheng
Ding-Zhu Du
Joon-Mo Kim
Lu Ruan
Computer Science Department
University of Minnesota, Minneapolis, MN 55455
E-mail: {cheng,dzd,jkim,ruan}@cs.umn.edu

Contents

1 Introduction

There are two interesting popular minimization problems on rectangular partition in the literature:

Minimum Length Rectangular Partition (MLRP): Given a rectilinear polygon possibly with some rectangular holes, partition it into rectangles with minimum total edge-length.

Minimum Cardinality Rectangular Partition (MCRP): Given a rectilinear polygon possibly with some rectangular holes, partition it into minimum number of rectangles.

The holes in the input rectangular polygon can be, possibly in part, degenerated into a line segment or a point. For the same input rectilinear polygon, MLRP, and MCRP may have different solutions.

In this survey, we present the state of arts on the two problems.

2 Minimum Length Rectangular Partition

The MLRP was first proposed by Lingas, Pinter, Rivest, and Shamir [18]. There are several applications mentioned for the background of the problem: "Process control (stock cutting), automatic layout systems for integrated circuit (channel definition), and artitecture (internal partitioning into offices). The *minimum edge-length* partition is a natural goal for these problems since there is a certain amount of waste (e.g. sawdust) or expense incurred (e.g. for dividing walls in the office) which is proportional to the sum of edge lengths drawn. For VLSI design, this criterion is used in the MIT 'PI' (Placement and Interconnect) System to divide the routing region up into channels - we find that this produces large 'natural-looking' chanels with a minimum of channel-to-channel interaction to consider."

They showed that the holes in the input make difference on the computational complexity. While the MLRP in general is NP-complete, the MLRP for hole-free inputs can be solved in time $O(n^4)$ where n is the number of vertices in the input rectilinear polygon. The polynomial algorithm is essentially a dynamic programming based on the following fact.

At each vertex on the boundary, there is an angle inside of the input area. The vertex is said to be concave if this angle is either 360° or 270°. For example, a point as a hole is a concave vertex, an endpoint of a segment as a hole is also a concave vertex.

Lemma 2.1 *In an optimal solution, every maximal cut segment must be incident to a concave vertex.*

Proof. If there is a maximal cut segment $[A, B]$ not incident to a concave vertex, then we count the number of cut segments touch its interior (A, B) from each side. Moving $[A, B]$ toward the side having a smaller number of cut segments touching (A, B) so that the moving does not increase the total length of cut segments. The moving can be stopped only by overlapping

another parallel segment, which result a saving of the total length of cut segments, a contradiction. □

A naive idea to design approximation algorithm for MLRP is to use a forest connecting all holes to the boundary and then to solve the resulting hole-free case in $O(n^4)$ time. With this idea, Lingas [20] gave the first constant-bounded approximation; its performance ratio is 41. Later, Du [9, 10] improved the algorithm and obtained a approximation with performance ratio 9. Meanwhile, Levcopoulos [17] provided a greedy-type faster approximation with performance ratio 29 and conjectured that his approximation may have performance ratio 4.5.

Motivated from a work of Du, Hwang, Shing, and Witbold [6] on application of dynamic programming to optimal routing trees, Du, Pan, and Shing [7] initiated an idea which leads to interesting further development. This idea is about guillotine cut.

2.1 Guillotine Cut

A cut is called a *guillotine cut* if it breaks a connected area into at least two parts. A rectangular partition is called a *guillotine rectangular partition* if it can be performed by a sequence of guillotine cuts. Du *et al* [7] noticed that there exists a minimum length guillotine rectangular partition which can be computed by a dynamic programming in $O(n^5)$ time. Therefore, they suggested to use the minimum length guillotine rectangular partition to approximate the MLRP and tried to analyze the performance ratio. Unfortunately, they failed to get a constant ratio in general and only obtained a result in a special case.

In this special case, the input is a rectangle with some points inside. Those points are holes. It had been showed (see [11]) that the MLRP in this case is still NP-hard. Du *et al* [7] showed that the minimum length guillotine rectangular partition as an approximation of the MLRP has performance ratio at most 2 in this special case. The following is a simple version of their proof, published in [8].

Theorem 2.2 *The minimum length guillotine rectangular partition is a approximation with performance ratio 2 for the MELGP.*

Proof. Consider a rectangular partition P. Let $proj_x(P)$ denote the total length of segments on a horizontal line covered by vertical projection of the partition P.

A rectangular partition is said to be covered by a guillotine partition if each segment in the rectangular partition is covered by a guillotine cut of the latter. Let *guil(P)* denote the minimum length of guillotine partition covering P and *length(P)* the total length of rectangular partition P. We will prove

$$guil(P) \leq 2 \cdot length(P) - proj_x(P)$$

by induction on the number k of segments in P.

For $k = 1$, we have *guil(P)= length(P).* If the segment is horizontal, then we have $proj_x(P) = length(P)$ and hence

$$guil(P) = 2 \cdot length(P) - proj_x(P).$$

If the segment is vertical, then $proj_x(P) = 0$ and hence

$$guil(P) < 2 \cdot length(P) - proj_x(P).$$

Now, we consider $k \geq 2$. Suppose that the initial rectangle has each vertical edge of length a and each horizontal edge of length b. Consider two cases:

Case 1. There exists a vertical segment s having length $\geq 0.5a$. Apply a guillotine cut along this segment s. Then the remainder of P is divided into two parts P_1 and P_2 which form rectangular partition of two resulting small rectangles, respectively. By induction hypothesis,

$$guil(P_i) \leq 2 \cdot length(P_i) - proj_x(P_i)$$

for $i = 1, 2$. Note that

$$\begin{aligned} guil(P) &\leq guil(P_1) + guil(P_2) + a, \\ length(P) &= length(P_1) + length(P_2) + length(s), \\ proj_x(P) &= proj_x(P_1) + proj_x(P_2). \end{aligned}$$

Therefore,

$$guil(P) \leq 2 \cdot length(P) - proj_x(P).$$

Case 2. No vertical segment in P has length $\geq 0.5a$. Choose a horizontal guillotine cut which partitions the rectangle into two equal parts. Let P_1 and P_2 denote rectangle partitions of the two parts, obtained from P. By induction hypothesis,

$$guil(P_i) \leq 2 \cdot length(P_i) - proj_x(P_i)$$

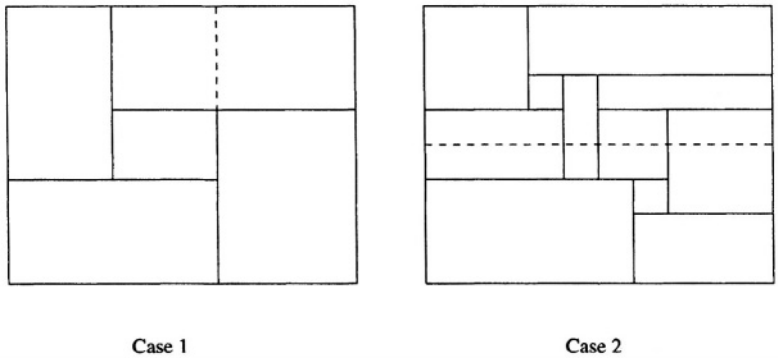


Figure 1: The proof of Theorem 2.2.

for $i = 1, 2$. Note that

$$\begin{aligned} guil(P) &= guil(P_1) + guil(P_2) + b, \\ length(P) &\geq length(P_1) + length(P_2), \\ proj_x(P) &= proj_x(P_1) = proj_x(P_2) = b. \end{aligned}$$

Therefore,

$$guil(P) \leq 2 \cdot length(P) - proj_x(P).$$

□

Gonzalez and Zheng [12] improved the constant 2 in Theorem 2.2 to 1.75 with a very complicated case-by-case analysis. Du, Hsu, and Xu [8] extended the idea of guillotine cuts to the convex partition problem.

The proof of Theorem 2.2 can be expressed in another way.

To do so, let us call a point a *vertical 1-dark point* if the vertical line through the point would meet cut segment in both directions.

In Case 1, there exists a vertical segment s having length $\geq 0.5a$. This is equivalent to say that there is a vertical middle point which is not a 1-dark point. Since the guillotine cut along s needs to add a segment of length less than $0.5a$, we may charge 1 to s. In Case 2, all vertical middle points are 1-dark, which form a guillotine cut. We may charge 0.5 to all horizontal pieces of the partition, facing this guillotine cut. Since each piece can be charged at most twice, therefore, the total length of segments added during the process of modifying a rectangular partition to a guillotine rectangular partition is bounded by the total length of original partition.

This "charge" argument was used by Mitchell [27, 28]. Actually, he gave an approximation with performance ratio 2 for the MLRP in the general case by extending the idea of guillotine cut.

2.2 1-Guillotine Cut

Mitchell [27, 28] noted that to have the dynamic programming runs in polynomial time, the cut does not need to be completely guillotine. He introduced a concept of 1-guillotine cut.

Initially, uses a rectangle to cover the input rectangular polygon with holes so that we can always work with a rectangle instead of a complicated rectangular polygon.

A vertical *1-guillotine cut* is a segment consisting of all vertical 1-dark points on the line, whuch partitions a rectangle into two rectangles each with two possible open boundary pieces resulting from the cut. (Fig. 2). A rectangular partition is 1-guillotine if it can be realized by a sequence

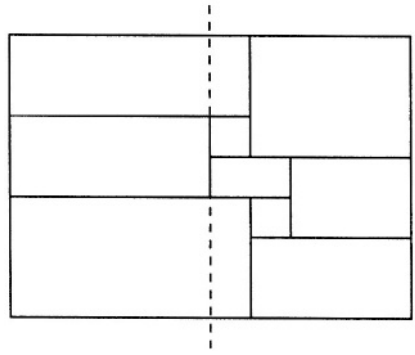


Figure 2: 1-guillotine cut.

of 1-guillotine cuts (Fig. 3). The minimum length 1-guillotine rectangular partition can be computed by dynamic programming in $O(n^{16})$ time. In fact, at each step, the 1-guillotine cut has $O(n^4)$ choices. There are $O(n^4)$ possible rectangles appearing in the algorithm. Each rectangle has $O(n^8)$ possible boundary conditions.

To establish the performance ratio of the minimum length 1-guillotine rectangular partition as an approximation of the MLRP, Mitchell [27] showed the following.

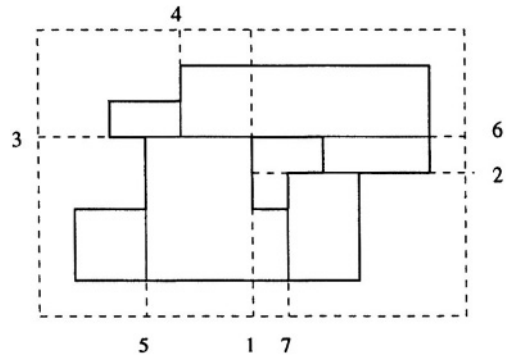


Figure 3: 1-guillotine rectangular partition with seven cuts.

Theorem 2.3 *For any rectangular partition P, there exists a 1-guillotine rectangular partition P' covering P such that*

$$length(P') \leq 2length(P).$$

Proof. It can be proved by an argument similar to the proof of Theorem 2.2. Let $guil_1(P)$ denote the minimum length of a 1-guillotine rectangular partition covering P and *length(P)* the length of the rectangular partition P. Let $proj_x(P)$ $(proj_y(P))$ denote the total length of segments on a horizontal (vertical) line covered by vertical (horizontal) projection of the partition P We will prove

$$guil_1(P) \leq 2 \cdot length(P) - proj_x(P) - proj_y(P)$$

by induction on the number k of segments in P.

For $k = 1$, we have $guil_1(P) = length(P)$. Without loss of generality, assume that the segment is horizontal. Then we have $proj_x(P) = length(P)$ and $proj_y(P) = 0$. Hence

$$guil_1(P) = 2 \cdot length(P) - proj_x(P) - proj_y(P).$$

Now, we consider $k \geq 2$ in the following two cases:

Case 1. There exists a 1-guillotine cut. Without loss of generality, assume this 1-guillotine cut is vertical with length a. Suppose the remainder of P is divided into two parts P_1 and P_2. By induction hypothesis,

$$guil_1(P_i) \leq 2 \cdot length(P_i) - proj_x(P_i) - proj_y(P_i)$$

for $i = 1, 2$. Notethat

$$\begin{aligned} guil_1(P) &\leq guil_1(P_1) + guil_1(P_2) + a, \\ length(P) &= length(P_1) + length(P_2) + a, \\ proj_x(P) &= proj_x(P_1) + proj_x(P_2) \\ proj_y(P) &\leq proj_y(P_1) + proj_y(P_2). \end{aligned}$$

Therefore,

$$guil_1(P) \leq 2 \cdot length(P) - proj_x(P) - proj_y(P).$$

Case 2. There does not exist 1-guillotine cut. In this case, we need to add a segment to partition P such that the resulting partition has a 1-guillotine cut and the length of added segment is at most $proj_x(P_1) + proj_x(P_2) - proj_x(P)$ if the 1-guillotine cut is horizontal and at most $proj_y(P_1) + proj_y(P_2) - proj_y(P)$ if the 1-guillotine cut is vertical, where P_1 and P_2 are partitions obtained from P by the 1-guillotine cut. To do so, it suffices to show that there exists a line such that the length of added segment for the line to become a 1-guillotine cut is not more than the total length of segments on the line, receiving projection from both sides. For simplicity of description, let us call by *horizontal (vertical) 1-dark point* a point receiving horizontal (vertical) projection from both sides. Then, for a horizontal (vertical) line, the set of vertical (horizontal) 1-dark points form the segment adding which would make the line become a 1-guillotine cut.

Lemma 2.4 *Let H (V) be the set of all horizontal (vertical) 1-dark points. Then there exists either a horizontal line L such that*

$$length(L \cap H) \leq length(L \cap V)$$

or a vertical line L such that

$$length(L \cap H) \geq length(L \cap V).$$

Proof. First, assume that the area of H is not smaller than the area of V. Denote $L_a = \{(x, y) \mid x = a\}$. Then areas of H and V can be represented by

$$\int_{-\infty}^{+\infty} length(L_a \cap H) da$$

and

$$\int_{-\infty}^{+\infty} length(L_a \cap V)da,$$

respectively. Since

$$\int_{-\infty}^{+\infty} length(L_a \cap H)da \geq \int_{-\infty}^{+\infty} length(L_a \cap V)da,$$

there must exist a such that

$$length(L_a \cap H) \geq length(L_a \cap V).$$

Similarly, if the area of H is smaller than the area of V, then there exists a horizontal line L such that

$$length(L \cap H) \leq length(L \cap V).$$

□

By Lemma 2.4, without loss of generality, we may assume that there exists a horizontal line L such that

$$length(L \cap H) \leq length(L \cap V),$$

that is,

$$length(L \cap H) \leq proj_x(P_1) + proj_x(P_2) - proj_x(P)$$

where P_1 and P_2 are subpartitions obtained from P by the line which becomes a 1-guillotine cut after adding segment $L \cap H$ to the partition P. By induction hypothesis,

$$guil(P_i) \leq 2 \cdot length(P_i) - proj_x(P_i) - proj_y(P_i)$$

for $i = 1, 2$. Note that

$$proj_y(P) \leq proj_y(P_1) + proj_y(P_2).$$

Therefore,

$$\begin{aligned} guil(P) &= guil(P_1) + guil(P_2) + length(L \cap H), \\ &\leq 2\sum_{i=1}^{2} length(P_i) - \sum_{i=1}^{2} proj_x(P_i) - \sum_{i=1}^{2} proj_y(P_i) + length(L \cap H) \\ &\leq 2 \cdot length(P) - proj_x(P) - proj_y(P). \end{aligned}$$

□

Mitchell [27, 28] used a different way to present the proof of Theorem 2.3. He symmetrically charged a half of the length of added segment to those parts of segments in P which face to 1-dark points. Since charge must be performed symmetrically, each point in P can be charged at most twice during the entire modification from a rectangular partition to a 1-guillotine rectangular partition. Therefore, the total length of added segments is at most *length(P)* and hence Theorem 2.3 holds. Actually, this argument is equivalent to the above proof of Theorem 2.3. In fact, only in case that projections from both sides exist (Case 2), $proj_x(P)$ or $proj_y(P)$ can contribute something against the length of the added segment.

2.3 m-Guillotine Cut

Mitchell [29] extended the 1-guillotine cut to the m-guillotine cut in the following way: A point p is a horizontal (vertical) *m-dark* point if the horizontal (vertical) line passing through p intersects at least $2m$ vertical (horizontal) segments of the considered rectangular partition P, among which at least m are on the left of p (above p) and at least m are on the right of p (below p). Let H_m (V_m) denote the set of all horizontal (vertical) m-dark points. An *m-guillotine cut* is either a horizontal line L satisfying

$$L \cap H_m \subseteq L \cap P$$

or a vertical line L satisfying

$$L \cap V_m \subseteq L \cap P.$$

An *m-guillotine cut* is the set of all m-dark points on a line. A rectangular partition is *m-guillotine* if it can be realized by a sequence of m-guillotine cuts. The minimum m-guillotine rectangular partition can also be computed by dynamic programming in $O(n^{10m+6})$ time. In fact, at each step, an m-guillotine cut has at most $O(n^{2(m+1)})$ choices. There are $O(n^4)$ possible rectangles appearing in the algorithm. Each rectangle has $O(n^{8m})$ possible boundary conditions. By a similar argument, Mitchell [29] established the following result.

Theorem 2.5 *For any rectangular partition P, there exists an m-guillotine rectangular partition P' covering P such that*

$$length(P') \leq (1 + \frac{1}{m})length(P).$$

Corollary 2.6 *There exists a* $(1+\varepsilon)$*-approximation with running time* $n^{O(\log 1/\varepsilon)}$ *for MLRP.*

It is very interesting to note that the technique of m-guillotine cut can be applied to many geometric optimization problems. Indeed, it is one of two major techniques in adaptive partition to design polynomial-time approximation schemes and was considered an important achievement in 1996.

In 1996, Arora [1] published a surprising result that many geometric optimization problems, including the Euclidean TSP (traveling salesman problem), the Euclidean SMT (Steiner minimum tree), the rectilinear SMT, the degree-restricted-SMT, k-TSP, and k-SMT, have polynomial-time approximation schemes. More precisely, for any $\varepsilon > 0$, there exists an approximation algorithm for those problems, running in time $n^{O(1/\varepsilon)}$, which produces approximation solution within $1+\varepsilon$ from optimal. *New York Times* reported this result. Several weeks later, Mitchell [29] claimed that a minor modification on his earlier work [27] (its journal version [28]) can lead to the similar results. Indeed, we already see that there is no technique difficulties from 1-guillotine cut to m-guillotine.

Although both Arora and Mitchell's approach follow the same framework of guillotine cut, i.e., sequential rectangular partition with dynamic programming, they use the different way to reduce the number of relations between two resulting smaller rectangles. In fact, one cannot replace another one and combination of two approaches made some more interesting results [2, 3, 30]. The reader may find more information from [26, 4].

3 Minimum Cardinality Rectangular Partition

The MCRP is also related to VLSI designs, specially VLSI mask generation. Liou, Tan and Lee [22] gave a clear description about this relation: "A VLSI mask is usually a piece of glass with a figure engraved on it. The engraved figure can be viewed as a rectilinear polygon on a digitized plane [31]. In order to engrave the figure on VLSI mask, a pattern generator is often used. A traditional pattern generator has a rectangular opening for exposure, which exposes rectangles onto the mask. Therefore, the engraved figure has to be decomposed into rectangles. The number of rectangles will determine the time required for mask generation."

When the input rectilinear polygon contains no degenerate hole, the MCRP was solved in polynomial-time at earlier stage. We can find $O(n^{2.5})$

time algorithms in [14, 16, 23], $O(n^{1.5}\log n)$ time algorithms in [15, 24, 25]. Lingas[19] asserted that with degenerate holes, the MCRP may be NP-hard. However, Soltan and Gorpinevich [32] found a $O(n^{1.5}\log n)$ time solution. For hole-free case, Liou, Tan and Lee [22, 21] can do as fast as $O(n\log\log n)$ while a lower bound $O(n\log n)$ is proved also by them [22] for general case.

Compared with the MERP, why the MCRP is so easy? We next give a little explanation.

We first consider the case of nonexistence of degenerate holes. In this case, if any cut line touching a concave vertex would remove its concavity. Therefore, if no two concave vertices can be connected by a *chord* (a rectilinear line segment, not on the boundary, connecting two concave vertices), then an optimal solution can be easily obtained in the following way: Iteratedly, for each concave vertex, removal its concavity by touching a line segment and extending this segment until meet an existing cut line or boundary. Note that a rectilinear polygon without concave vertices must be a rectangle. When the input polygon has no hole, the rectangular partition is done.

For the hole-free case, since removal each concave vertex results in one more area, the total number of rectangles in the final partition is $c+1$ where c is the number of concave vertices. For input with h holes, since removal each concave vertex woud either result in one more area or reduce one hole, the number of rectangles in the optimal rectangular partition is $c+1-h$.

If there exist chords, then we must find a maximum set of disjoint chords. This is a maximum matching problem on the chord graph with concave vertices as vertices and chords as edges. Therefore, the time complexity depends on algorithm for solving maximum matching problem in the chard graph.

When degenerate holes exist, there exist concave vertice of 360°, which make the problem more complicated since a line segment incident to such a concave vertex may not remove its concavity.

It is worth mentioning that when the MCRP extends to the 3-dimensional space, the problem becomes NP-hard [5].

References

[1] S. Arora, Polynomial-time approximation schemes for Euclidean TSP and other geometric problems, *Proc. 37th IEEE Symp. on Foundations of Computer Science,* (1996) pp. 2-12.

[2] S. Arora, Nearly linear time approximation schemes for Euclidean TSP and other geometric problems, I*Proc. 38th IEEE Symp. on Foundations of Computer Science,* (1997) pp. 554-563.

[3] S. Arora, Polynomial-time approximation schemes for Euclidean TSP and other geometric problems, *Journal of the ACM* 45 (1998) 753-782.

[4] Mihaela Cadei, Xiaoyan Cheng, Xiuzhen Cheng, and Ding-Zhu Du, A tale in guillotine cut, in Proc. of Workshop on Novel Approaches to Hard Discrete Optimization, 2002

[5] V.J. Dielissen and A. Kaldewaij, Rectangular partition is polynomial in two dimensions but NP-complete in three, *Information Processing Letters* 38 (1991) 1-6.

[6] D.Z. Du, F.K. Hwang, M.T. Shing and T. Witbold: Optimal routing trees, *IEEE Transactions on Circuits* 35 (1988) 1335-1337.

[7] D.-Z. Du, L.-Q. Pan, and M.-T. Shing, Minimum edge length guillotine rectangular partition, Technical Report 0241886, Math. Sci. Res. Inst., Univ. California, Berkeley, 1986.

[8] D.-Z. Du, D.F. Hsu, and K.-J Xu, Bounds on guillotine ratio, *Congressus Numerantium* 58 (1987) 313-318.

[9] D.-Z. Du, On heuristics for minimum length rectangularpartitions,Technical Report, Math. Sci. Res. Inst., Univ. California, Berkeley, 1986.

[10] D.-Z. Du and Y.-J. Zhang: On heuristics for minimum length rectilinear partitions, *Algorithmica,* 5 (1990) 111-128.

[11] T. Gonzalez and S.Q. Zheng, Bounds for partitioning rectilinear polygons, *Proc. 1st Symp. on Computational Geometry,* 1985.

[12] T. Gonzalez and S.Q. Zheng, Improved bounds for rectangular and guillotine partitions, *Journal of Symbolic Computation* 7 (1989) 591-610.

[13] A. Gorpinevich and V. Soltan, Simple algorithms of the partition of rectilinear regions into rectangles in VLSI engineering, *Bull. Acad. Sci. Mold. SSR. Ser. Phis.–Techn, and Math. Sci.* No. 2 (1989) 19-25 (Russian).

[14] L. Ferrari, P.V. Sancar and J. Sklansky, Minimal rectilinear partitions of digitized blocks, *Comput. Vision Graphics Image Process* 28 (1984) 58-71.

[15] H. Imai and T. Asano, Efficient algorithm for geometric graph search problems, *SIAM J. Comput* 15 (1986) 478-494.

[16] N.M. Kornienko, G.V. Matveev, N.N. Metelsky, and R.I. Tyshkevich, On tiling of polygons, *Izv. Akad. Nauk. BSSR. Ser. Phis.-Math. Sci.* No 2 (1979) 25-29 (Russian).

[17] C. Levcopoulos, Fast heuristics for minimum length rectangular partitions of polygons, *Proc 2nd Symp. on Computational Geometry,* 1986.

[18] A. Lingas, R. Y. Pinter, R. L. Rivest, and A. Shamir, Minimum edge length partitioning of rectilinear polygons, *Proc, 20th Allerton Conf. on Comm. Control and Compt.,* Illinos, 1982.

[19] A. Lingas, The power of non-rectilinear holes, Lecture Notes on Computer Science, Vol. 140 (Springer-Verlag, 1982) 369-383.

[20] A. Lingas, Heuristics for minimum edge length rectangular partitions of rectilinear figures, *Proc. 6th GI-Conference,* Dortmund, January 1983 (Springer-Verlag).

[21] W.T. Liou, J.J.-M. Tan, and R.C.T. Lee, Minimum rectangular partition problem for simple rectilinear polygons, *IEEE Trans. on Computer-Aided Designs* 9:7 (1990) 720-733.

[22] W.T. Liou, J.J.-M. Tan, and R.C.T. Lee, Minimum partioning simple rectilinear polygons in $O(n \log \log n)$-time, 1989, manuscript.

[23] W. Lipski, E. Lord, F. Luccio, C. Mugnai, and L. Pagli, On two-dimensional data organization II, *Fund. Inform.* 2 (1979) 245-260.

[24] W. Lipski, Finding a Manhattan path and related problems, *Networks* 13 (1983) 399-409.

[25] W. Lipski, An $O(n \log n)$ Manhattan path algorithm, *Information Processing Letters* 19 (1984) 99-102.

[26] B. Lu and L. Ruan, Polynomial time approximation scheme for the rectilinear Steiner arborescence problem, *Journal of Combinatorial Optimization* 4 (2000) 357-363.

[27] J.S.B. Mitchell, Guillotine subdivisions approximate polygonal subdivisions: A simple new method for the geometric k-MST problem. *Proc. 7th ACM-SIAM Symposium on Discrete Algorithms,* (1996) pp. 402-408.

[28] J.S.B. Mitchell, A. Blum, P. Chalasani, S. Vempala, A constant-factor approximation algorithm for the geometric k-MST problem in the plane *SIAM J. Comput.* 28 (1999), no. 3, 771–781.

[29] J.S.B. Mitchell, Guillotine subdivisions approximate polygonal subdivisions: Part II - A simple polynomial-time approximation scheme for geometric k-MST, TSP, and related problem, *SIAM J. Comput.* 29 (1999), no. 2, 515–544.

[30] J.S.B. Mitchell, Guillotine subdivisions approximate polygonal subdivisions: Part III - Faster polynomial-time approximation scheme for geometric network optimization, preprint, 1997.

[31] T. Ohtsuki, Minimum dissection of rectilinear region, *Proceedings of IEEE Symposium on Circuits and Systems,* 1982, pp.1210-1213.

[32] V. Soltan and A. Gorpinevich, Minimum dissection of a rectilinear polygon with arbitrary holes into rectangles, *Discrete and Computational Geometry* 9 (1993) 57-79.

HANDBOOK OF COMBINATORIAL OPTIMIZATION (VOL 5)
D.-Z. Du and P. Pardalos (Eds.) pp. 329-369

Connected Dominating Set in Sensor Networks and MANETs

Jeremy Blum Min Ding Andrew Thaeler Xiuzhen Cheng
Department of Computer Science
The George Washington University, Washington, DC 20002
E-mail: {blumj,minding,athaeler,cheng}@gwu.edu

Contents

1 Introduction

Recently developed classes of wireless networks have blurred the distinction between the network infrastructure and network clients. Sensor networks, for example, consist of one or more base stations and a large number of inexpensive nodes, which combine sensors and low power wireless radios. Due to limited radio range and battery power, most nodes cannot communicate directly with a base station, but rather rely on their peers to forward messages to and from base stations. Likewise, in mobile ad hoc networks (MANETs), the routing of messages is also performed by ordinary nodes. In fact, a MANET typically has no network infrastructure, therefore all routing and network management functions must be performed by ordinary nodes.

The key to scalability and efficiency in traditional computer networks is the organization of the network infrastructure into a hierarchical structure. However, due to the lack of a network infrastructure, sensor networks and MANETs are inherently flat. In order to achieve scalability and efficiency, new algorithms have emerged that rely on a virtual network infrastructure, which organizes ordinary nodes into a hierarchy. The construction of this infrastructure is the primary application of Connected Dominating Sets (CDSs) in wireless networks.

The utility of CDSs in wireless ad hoc networks has been demonstrated in protocols that perform a wide range of communication functions. CDSs have formed an underlying architecture used by protocols including media access coordination [8, 43, 64]; unicast [35, 33, 34, 76], multicast/broadcast [51, 52, 67, 73, 78, 79, 80], and location-based routing [36]; energy conservation

[19, 38, 63, 72, 82]; and topology control [37, 38]. CDS can also be used to facilitate resource discovery in MANET [45, 48].

In this chapter, we are going to survey the CDS construction techniques proposed in the context of sensor networks and MANETs. Sections 3 and 4 address details of centralized and distributed algorithms respectively. Theoretically any centralized algorithm can be implemented in a distributed fashion, with the tradeoff of higher protocol overhead. We are going to examine several centralized algorithms and their corresponding distributed implementations in detail. Distributed or even localized algorithms are very important for sensor networks and MANETs. CDS must be constructed efficiently to be applicable in a mobile or large scale network. Due to the dynamism of wireless links and nodal mobility, algorithms should rely on limited knowledge of the current network topology.

Note that the design goals of different algorithms vary based on the needs of the protocols making use of the CDS. When designing a CDS algorithm, one must take the following parameters into consideration: performance bounds, degree of localization, time and message complexities, and stability with respect to nodal movement. We are going to analyze these algorithms in Section 5. We will provide a toolbox of techniques (Subsection 5.1) elicited from the examination of CDS construction algorithms, and present guidelines (Subsection 5.2) to aid in the selection of particular techniques based on the design goals of an application.

Actually many works seek a *minimum connected dominating set (MCDS)* in unit-disk graphs as their major design goal. Thus performance bounds is their primary design parameter. The rationale of this problem formulation can be justified as follows. The foot print of an ad hoc network with fixed transmission range for each host can be modelled by a unit-disk graph [25]. And minimizing the cardinality of the computed CDS can help to decrease the control overhead since broadcasting for route discovery [47, 60] and topology update [30] is restricted to a small subset of nodes [25]. Therefore *broadcast storm problem* [59] inherent to global flooding can be greatly decreased.

Other works seek a connected dominating set that provides good resource conservation property [19, 76]. Thus performance bound is not their primary consideration. Instead, the hop count of communication path between nodes is taken into consideration for load balance [19] and power conservation [78, 79]. In the following section (Section 2), we will present network model and useful definitions needed for algorithm elaboration. We also will give an overview of CDS applications.

2 Network Model and CDS Applications

The following section provides background information for the analysis of CDS applications in ad hoc wireless networks. We first present a mathematical model for the networks under consideration and introduce useful terminologies and definitions from graph theory. Then we sketch the various wireless network applications utilizing connected dominating sets.

2.1 Network Model

An ad hoc wireless network can be represented by a graph $G(V, E^t)$ comprised of a set of vertices V and time-varying edges E^t. For each pair of vertices $u, v \in V$, $(u, v) \in E^t$ if and only if the nodes u and v are within communication range. Due to nodal movement, the topology of the network is dynamic, as reflected by E^t.

Given omni-directional antennae, the communication range of a node in a wireless network is typically modelled as a disk centered at the node with radius equal to the transmission range of the radio. Consequently, when transmission range is fixed for all nodes, the network has the property of a unit-disk graph (UDG), where an edge exists if and only if two nodes have inter-nodal distance less than or equal to 1 unit (the fixed communication range). Many of the CDS algorithms use the properties of UDG's to prove their performance bounds.

Each node v has an associated set of nodal properties. Typical properties include the following:

- ID_v, the unique ID for node v.
- loc_v^t, the location of node v at time t.
- $velocity_v^t$, the velocity vector for node v at time t.

A number of definitions from graph theory are used in this chapter. Figure 1 can help to illustrate the following concepts:

- *Open Neighbor Set,* $N(u) = \{v \mid (u, v) \in E\}$, is the set of nodes that are neighbors of u. In Figure 1, the open neighbor set of e is $\{d, f, g\}$.
- *Closed Neighbor Set,* $N[u] = N(u) \cup \{u\}$, is the set of neighbors of u and u itself. In Figure 1, the closed neighbor set of e is $\{d, e, f, g\}$.
- *Maximum Degree,* Δ, is the maximum count of edges emanating from a single node. The maximum degree of the graph in Figure 1 is three, and occurs at nodes c, e, and g.

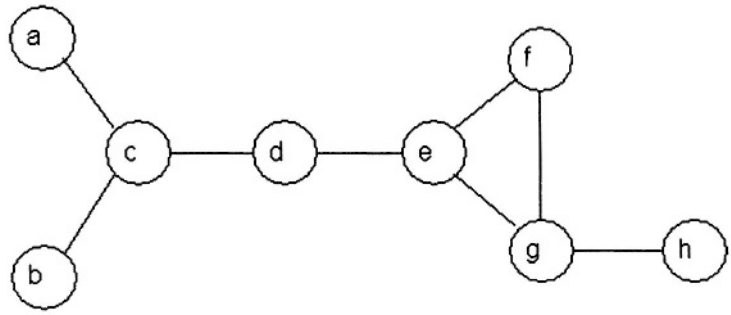


Figure 1: Representation of a wireless network with eight nodes as a graph.

- *Independent Set,* is a subset of V such that no two vertices within the set are adjacent in V. For example, $\{a, b, f, h\}$ is an independent set in Figure 1.

- *Maximal Independent Set (MIS),* is an independent set such that adding any vertex not in the set breaks the independence property of the set. Thus, any vertex outside of the maximal independent set must be adjacent to some node in the set. The previous independent set $\{a, b, f, h\}$ must have node d added to become an MIS.

- *Dominating Set, S,* is defined as a subset of V such that each node in $V - S$ is adjacent to at least one node in S. Thus, every MIS is a dominating set. However, since nodes in a dominating set may be adjacent to each other, not every dominating set is an MIS. Finding a minimum-sized dominating set or MDS is NP-Hard [41].

- *Connected Dominating Set (CDS), C,* is a dominating set of G which induces a connected subgraph of G. One approach to constructing a CDS is to find an MIS, and then add additional vertices as needed to connect the nodes in the MIS. A CDS in Figure 1 is $\{c, d, e, g\}$.

- *Minimum Connected Dominating Set* (MCDS) is the CDS with minimum cardinality. Given that finding minimum sized dominating set is NP-Hard, it should not be surprising that finding the MCDS is also NP-Hard [41]. In Figure 1, $\{c, g\}$ is a minimum connected dominating set.

- *Weakly Connected Dominated Set (WCDS), S,* is a dominating set such that $N[S]$ induces a connected subgraph of G. In other words, the

subgraph weakly induced by S is the graph induced by the vertex set containing S and its neighbors. Given a connected graph G, all of the dominating sets of G are weakly connected. Computing a minimum WCDS is NP-Hard [41].

- *Steiner Tree,* is a minimum weight tree connecting a given set of vertices in a weighted graph. After finding an MIS, connecting the nodes together could be formulated as an instance of the Steiner Tree problem. Like many of the other problems that arise in CDS construction, this problem is NP-Hard [41].

2.2 Applications of CDS in Wireless Networks

Sensor networks and MANETs have unique characteristics that require the development of protocols specific to them. For efficiency reasons, many of these protocols first organize the network through the construction of dominating sets. These protocols address media access, routing, power management, and topology control.

At the Link Layer, clustering can increase spatial reuse of the spectrum, minimize collisions, and provide Quality of Service (QoS) guarantees [8, 43, 64, 65]. Correspondingly, the nodes in the dominating set can coordinate with one another and use orthogonal spreading codes in their neighborhoods to improve spatial reuse with code division spread spectrum techniques [42]. Furthermore, these nodes can coordinate access to the wireless media by their neighbors for QoS or collision avoidance purposes.

As first noted by Ephremedis *et al.,* a CDS can create a virtual network backbone for packet routing and control [40]. Messages can be routed from the source to a neighbor in the dominating set, along the CDS to the dominating set member closest to the destination node, and then finally to the destination. This is termed *dominating set based routing* [33, 76], or *Backbone based rouing* [34], or *spine based routing* [35, 66]. Restricting the routing to the CDS results in a significant reduction in message overhead associated with routing updates [18]. Furthermore, the dominating set can be organized into a hierarchy to further reduce control message overhead [56, 64, 65].

A CDS is also useful for location-based routing. In location-based routing, messages are forwarded based on the geographical coordinates of the hosts, rather than topological connectivity. Intermediate nodes are selected based on their proximity to the message's destination. With this scheme, it is possible for a message to reach a local maximum, where it has been sent

to an intermediate node whose neighbors are all further from the destination than itself. In this case, the routing must enter a recovery phase, where the route may backtrack to find another path. However, if messages are only forwarded to nodes in the dominating set, the inefficiency associated with this recovery phase can be greatly reduced [36].

The efficiency of multicast/broadcast routing can also be improved through the utilization of CDSs. A big problem in multicast/broadcast routing is that many intermediate nodes unnecessarily forward a message. Nodes often hear the same message multiple times. This is the *broadcast storm problem* [59]. If the message is routed along a CDS, most of the redundant broadcasts can be eliminated [25, 51, 52, 67, 73, 78, 79, 80].

Nodes in a wireless network often have a limited energy supply. CDSs play an important role in power management. They have been used to increase the number of nodes that can be in an sleep mode, while still preserving the ability of the network to forward messages [19, 38, 82]. They have also been used to balance the network management requirements to conserve energy among nodes [63, 72, 77, 78, 79, 80].

In large-scale dense sensor networks, sensor topology information extraction can be handled by CDS construction [37, 38]. Other than routing, the virtual backbone formed by dominating set can also be used to propagate "link quality" information for route selection for multimedia traffic [64], or to serve as database servers [49], etc.

3 Centralized CDS Construction

The first instance of a dominating set problem arose in the 1850's, well before the advent of wireless networks [70]. The objective of the five queens problem is to find the minimum number of queens that can be placed on a chessboard such that all squares are either attacked or occupied by a queen. This problem was formulated as a dominating set of a graph $G(V,E)$, with the vertices corresponding to squares on the chessboard, and $(u, v) \in E$ if and only if a queen can move from the square corresponding to u to the square corresponding to v.

MCDS in general graphs was studied in [41], in which a reduction from the *Set Cover Problem* [41] to the MCDS problem was shown. This result implies that for any fixed $0 < \epsilon < 1$, no polynomial time algorithm with performance ratio $\leq (1-\epsilon)H(\Delta)$ exists unless $NP \subset DTIME[n^{O(\log \log n)}]$ [54], where Δ is the maximum degree of the input graph and H is the harmonic function. The MCDS remains NP-hard [29] for unit-disk graphs.

MCDS in unit-disk graphs has constant performance ratio, as proved by [4, 16, 23, 69]. A polynomial time approximation scheme for computing a MCDS in unit-disk graphs has been developed by Cheng *et al.* in [26]. A significant impact of this result is that a MCDS in unit-disk graphs can be approximated to any degree if computing time is permitted. Note that heuristics proposed for unit-disk graphs work well for general graphs, but their performance analysis is unapplicable. Thus in this section and next, we will focus on algorithm description and skip the corresponding performance analysis. Note that we intentionally omit the fact that these algorithms are proposed in either unit-disk graphs or general graphs because they are actually applicable in both graph models. Also note that we are going to use either the name of the first author or all authors' names of the paper to represent the algorithm.

In the following we will focus on centralized CDS construction algorithms. Distributed heuristics will be discussed in Section 4.

3.1 Guha and Khuller's Algorithm

In 1998, Guha and Khuller proposed two CDS construction strategies in their seminal work [44], which contains two greedy heuristic algorithms with bounded performance guarantees. In the first algorithm, the CDS is grown from one node outward. In the second algorithm, a WCDS is constructed, and then intermediate nodes are selected to create a CDS. The distributed implementations of both algorithms were provided by Das *et al.* in [33], which will be addressed in Subsection 4.2. Many algorithms designed latter [23, 69] are motivated by either of these two heuristics. We sketch the procedures in the following.

The first algorithm begins by marking all vertices white. Initially, the algorithm selects the node with the maximal number of white neighbors. The selected vertex is marked black and its neighbors are marked gray. The algorithm then iteratively scans the gray nodes and their white neighbors, and selects the gray node or the pair of nodes (a gray node and one of its white neighbors), whichever has the maximal number of white neighbors. The selected node or the selected pair of nodes are marked black, with their white neighbors marked gray. Once all of the vertices are marked gray or black, the algorithm terminates. All the black nodes form a connected dominating set. This algorithm yields a CDS of size at most $2(1 + H(\Delta)) \cdot |OPT|$, where H is the harmonic function, and OPT refers to an optimal solution – that is, a minimum connected dominating set.

For example, consider the graph in Figure 2. Initially, either node c, or e,

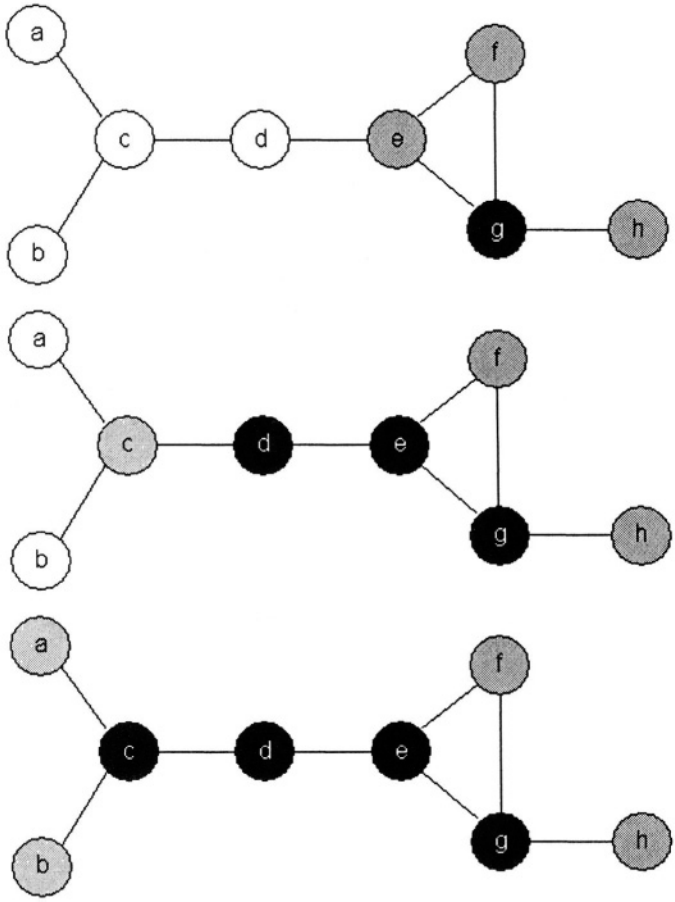


Figure 2: An example of Guha's first algorithm.

or g could be marked since they have maximal degree. Node g is arbitrarily picked from these candidates and marked black. All its neighbors are then marked gray. We now consider the gray nodes e, f, h, and the pair of gray and white nodes (d, e). Of all these, the pair (d, e) covers the most number of white neighbors - two. So we mark both d and e black. Finally, we consider the gray node c and the pairs (c, a) and (c, b). All these candidates have the same number of white neighbors. Therefore the single node c is selected. Now all the nodes are either black or gray, and the set of nodes in black $\{c, d, e, g\}$ forms a CDS.

The second algorithm also begins by coloring all nodes white. A *piece* is defined to be either a connected black component, or a white node. The algorithm contains two phases. The first phase iteratively selects a node that causes the maximum reduction of the number of pieces. In other words, the greedy choice for each step in the first phase is the node that can decrease the maximum number of pieces. Once a node is selected, it is marked black and its white neighbors are marked gray. The first phase terminates when no white node left. After the first phase, there exists at most $|OPT|$ number of connected black components. The second phase constructs a Steiner Tree that connects all the black nodes by coloring chains of two gray nodes black. The size of the resulting CDS formed by all black nodes is at most $(3 + ln(\Delta)) \cdot |OPT|$.

Figure 3 shows an example of the second algorithm. First, node g is marked as it is one of the nodes with the maximum number of white neighbors. Next, node c is marked because it can reduce the maximum number of pieces compared with any other node. Now the first phase ends as there is no white node left. In the second phase, a Steiner Tree is constructed by adding nodes d and e to connect nodes c and g.

3.2 Ruan's Algorithm

The potential function used in the second algorithm of Guha and Khuller [44] is the number of pieces. Each step seeks maximum reduction in the number of pieces in the first phase. By modifying the potential function, Ruan *et al.* [62] proposes a one-step greedy approximation algorithm with performance ratio at most $3+ln(\Delta)$. This algorithm also requires each node to be colored white at the beginning. If there exists a white or gray node such that coloring it black and its white neighbors gray would reduce the potential function, then choose the one that causes maximum reduction in the potential function.

The potential function plays a critical rule in this algorithm. It is defined

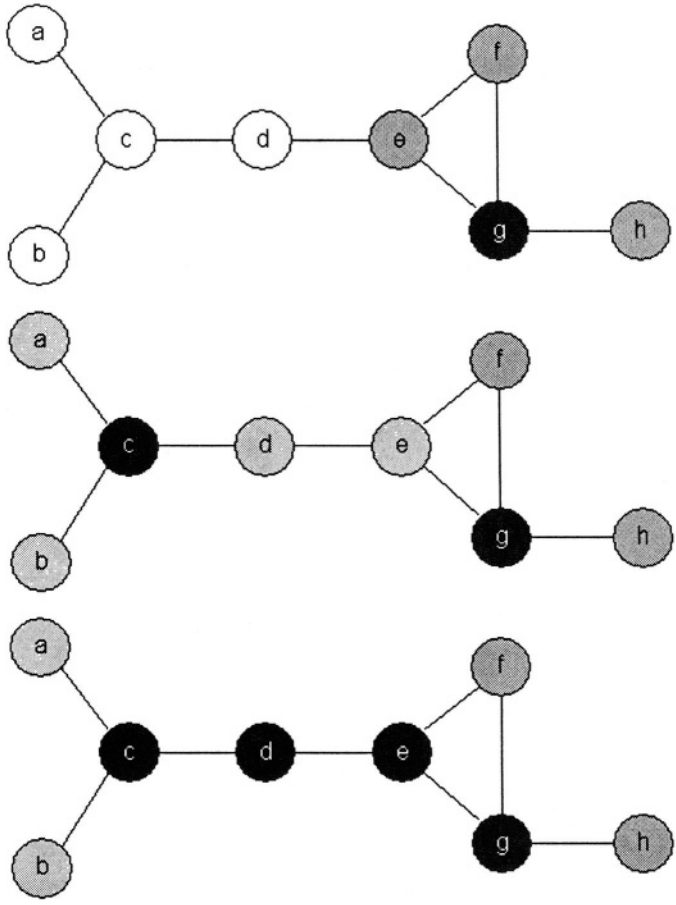


Figure 3: An example of Guha's second algorithm.

in the following way in [62]. Given a connected graph $G(V, E)$, define $p(C)$ be the number of connected black components in the subgraph induced by $C \subset V$. Let $D(C)$ be the set of all edges incident to vertices in C. Define $q(C)$ be the number of connected components in the subgraph $G(V, D(C))$. Then the potential function is defined to be $f(C) = p(C) + q(C)$.

In the greedy algorithm, let C be the set containing all black nodes. Thus initially $f(C) = |V|$ since $C = \phi$. The first step chooses a node x with maximum degree. Every other step selects a node x such that $f(C) - f(C \cup \{x\})$ is maximized. Color node x black and color all its white neighbors gray. The algorithm ends when $f(C) = 2$, where C is the resultant CDS.

3.3 Cheng's Greedy Algorithm

In [24], Cheng *et al.* propose a greedy algorithm for MCDS in unit-disk graphs. Compared to the many heuristics discussed in Subsection 4.3, this algorithm relies on an MIS but the resultant CDS may not contain all the elements in the MIS.

Assume initially all nodes are colored white. The construction of a CDS contains four phases. In the first phase, an MIS is computed and all its members are colored red. In the second phase, a node that can decrease the maximum number of pieces is selected, where a piece is either a red node, or a connected black component. This node is colored black and all its non-black neighbors are colored gray. When the second phase is over, we still have some white nodes left. The third phase will compute a spanning tree for each connected component in the subgraph reduced by all white nodes. Connect each tree to the nearest black component with black nodes accordingly. All non-leaf tree nodes are colored black while leaf nodes are colored gray. The last phase will seek chains of two gray nodes to connect disjoint black components.

The motivation of Cheng's algorithm are two fold. First, the greedy choice in Guha and Khuller's second algorithm [44] is the one that can decrease the maximum number of pieces, where a piece is either a connected black component, or a white node. Second, a unit-disk graph has at most 5 independent neighbors. Thus intuitively one can choose the greedy choice that can connect to as many independent nodes as possible. In other words, the node to be colored black at each step will try to cover more uncovered area, if we model vertices in a unit-disk graph as nodes in a flat area. Unfortunately Cheng's algorithm does not have a solid performance analysis.

3.4 Min's Algorithm

Recently Min *et al.* [58] propose to use a *Steiner tree with minimum number of Steiner nodes (ST-MSN)* [20, 39, 53] to connect a maximal independent set. This algorithm contains two phases. The first phase constructs an MIS with the following property: every subset of the MIS is two hops away from its complement. Color all nodes in the MIS black; color all other nodes gray. In the second phase, a gray node that is adjacent to at least three connected black components is colored black in each step. If no node satisfying this condition can be found, a gray node that is adjacent to at least two connected black components will be colored black. This algorithm has performance ratio 6.8 for unit-disk graphs. Its distributed implementation is sketched in Subsection 4.6.

ST-MSN in Euclidean plane is NP-hard [53]. A 3-approximation algorithm for ST-MSN in Euclidean plane is proposed in [20] and is extended to unit-disk graphs by Min *et al.* in [58]. Since the size of any MIS is at most $3.8 \cdot |OPT| + 1.2$ [81] in unit-disk graphs, where OPT is any MCDS, the computed CDS by Min's algorithm has size at most $6.8 \cdot OPT$ [58].

3.5 Butenko's Algorithm

The heuristic proposed in [14, 15] is pruning-based. In other words, the connected dominating set S *is* initialized to the vertex set of graph $G(V, E)$, and each node will be examined to determine whether it should be removed or retained. Assume all nodes in S are colored white at the beginning. Define the *effective degree* of a node to be its white neighbors in S. Consider a white node $x \in S$ with minimum effective degree. If removing x from S makes the induced graph of S disconnected, then retain x and color it black. Otherwise, remove x from S. At the same time, if x does not have a black neighbor in S, color its neighbor with maximum effective degree in S black. Repeat this procedure until no white node left in S. This algorithm has time complexity $O(|V| \cdot |E|)$. It does not have a performance analysis. We will discuss its distributed implementation in Subsection 4.4.2.

4 Distributed CDS Construction

For sensor networks and MANETs, distributed CDS construction is more effective due to the lack of a centralized administration. On the other hand, the large problem size (e.g. a sensor network may contain hundreds of thousands of sensors) also prohibits the centralized CDS computation. In this

section, we survey a variety of distributed approaches that seek to balance the competing requirements of complexity, running time, stability, and overhead. Note that many published results contain algorithms that differ very little from each other. To be more focus, we decide only to cover the major distributed CDS construction techniques.

In wireless networks, CDS problems have been formulated in a number of ways, depending on the needs of the particular application. These formulations can be classified into WCDSs, non-localized CDSs, localized CDSs, and stable CDSs with nodal mobility. In this chapter, we choose to use a different classification containing the following categories based on the CDS construction techniques: WCDS, greedy CDS, MIS based CDS, pruning based CDS, multipoint forwarding based CDS, Steiner tree based CDS, and stable CDS. In the following, we will examine each category and explore example algorithms in detail. We also will sketch the maintenance of the computed CDS if available.

4.1 WCDS Construction

In a WCDS, the vertex set is partitioned into a set of clusterheads and cluster members, such that each cluster member is within radio range of at least one clusterhead. There are two ways of approaching this problem. If nodal location information is known, then the nodes can be clustered geographically. Otherwise, nodes can be clustered based solely on the graph topology. Chen and Liestman [21] propose a series of approximate algorithms for computing a small WCDS to be used to cluster mobile ad hoc networks.

Geographical clustering algorithms create a virtual grid in the geographical region where the network exists. Each cluster comprises all of the nodes in a given grid. The Grid Location Service, for example, uses this grid structure to disseminate the location information of nodes throughout the network. Using the grid structure, the density of nodes holding the location information of other nodes decreases as the distance from the node increases [50]. Geographical Adaptive Fidelity attempts to minimize energy consumption in a network by clustering the nodes based on a grid, and having only one node per grid responsible for routing at any given time [82].

In the first WCDS algorithm for wireless networks, nodes elect their neighbor with the lowest ID as their clusterhead [40]. Nodes learn about the ID's of their current neighbors through periodic beacons that each node broadcasts. Whenever a node with a lower ID moves into range of a clusterhead, it becomes the new clusterhead.

Highest Connectivity Clustering presents an improvement over Lowest ID Clustering by reducing the number of clusters [43]. In addition to periodically broadcasting its ID, each node also broadcasts the size of N(u), its open neighbor set. A node becomes clusterhead if it is the most highly connected among its neighbors. When nodes are moving, this selection criterion is less stable than Lowest ID Clustering, since ID's do not change, but nodal degree changes often.

One potential problem with Lowest ID Clustering is its unfairness. Nodes in the dominating set may be burdened with additional responsibilities. To be more equitable in clusterhead selection, the Min-Max D-Cluster algorithm selects nodes based on ID, but attempts to elect both low and high ID nodes [7]. Another interesting feature of the algorithm is the creation of d-hop dominating sets where each node is at most d hops from a node in the dominating set.

A serious problem with the Lowest ID Clustering is that clusterhead selection can be very unstable. The left side of Figure 4 displays a cluster headed by Node 2. As shown on the right side, once Node 1 moves into radio range, Node 2's cluster will be disbanded and replaced by three clusters. This reorganization is unnecessary, since Node 1 could have simply joined the existing cluster.

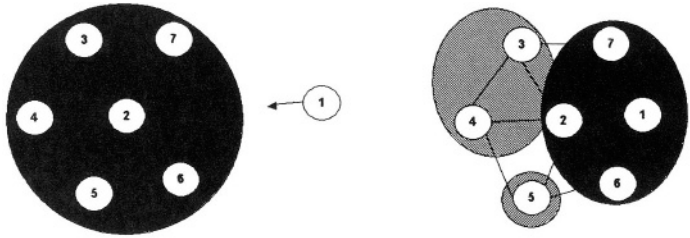


Figure 4: Instability of Lowest ID Clustering. As shown on the left, Node 1 is moving into range of Node 2's cluster. Once Node 1 is in radio range, Node 2's cluster will be reorganized.

In order to address this instability, the clustering algorithm introduced by the Clusterhead-Gateway Switch Routing Protocol preserves clusterhead elections under all but two conditions [27]. The set of clusterheads changes only when two clusterheads come into contact with each other, or when a node loses contact with all clusterheads.

4.2 Greedy CDS Construction

Das *et al.* [35, 33, 66] propose the distributed implementations of the two greedy algorithms given by Guha and Khuller in [44].

The first algorithm grows a CDS from one node with maximum degree. Thus the first step involves selecting the node with the highest degree. Therefore a node must know the degree of all nodes in the graph. On the other hand, each iterative step selects either a one- or two-edged path emanating from the current CDS, thus the nodes in the CDS must know the number of unmarked neighbors for all nodes one and two hops from the CDS. These two requirements force the flooding of degree information in the network. This algorithm generates a CDS with approximation ratio of $2H(\Delta)$ in $O(|C|(\Delta + |C|))$ time, using the $O(n|C|)$ messages, where the harmonic function $H(\Delta) = \sum_{i=1}^{\Delta} 1/i \leq \ln(\Delta) + 1$, n is the total number of vertices, and C represents the generated CDS.

The second algorithm first computes a dominating set and then selects additional nodes to connect the set. In order to locally select nodes for the dominating set in the first stage, an unmarked node compares its effective degree, the number of unmarked neighbors, with the effective degrees of all its neighbors in two-hop neighborhood. The greedy algorithm iteratively adds the node with maximum effective degree to the dominating set. The first stage terminates when a dominating set is achieved. The second stage connects the obtained components using a distributed minimum spanning tree algorithm, with the goal of adding as few nodes as possible. To do this, each edge is assigned a weight equal to the number of endpoints not in the dominating set. At the end, the interior nodes in the resulting spanning tree compose a connected dominating set. This algorithm has time complexity of $O((n + |C|)\Delta)$, and message complexity of $O(n|C| + m + n\log(n))$. It approximates the MCDS with a ratio of $2H(\Delta) + 1$, where m is the cardinality of the edge set.

Das *et al.* [35, 33, 66] handle CDS maintenance differently in the case of single-node movement versus multiple-node movement. If a single node moves, the CDS can be updated locally. When more than one node moves, the moves can be treated as many single-node moves if they have no overlapping neighborhoods. However, an entirely new CDS computation may be needed in the case of overlapping neighborhoods.

4.3 MIS Based CDS Construction

Algorithms in this category compute and connect an MIS. But how an MIS is computed and connected differs from algorithm to algorithm. One can compute an MIS based on either single leader [2, 3, 4, 5, 13, 16, 23, 69] or multiple leaders [6, 25]. The MIS can be connected after the construction is over [2, 3, 4, 5, 6, 13, 16, 23, 25, 69], or one can compute and connect an MIS simultaneously [23, 25]. Note that single leader based MIS construction needs a leader-election algorithm, which takes $O(n \log n)$ messages [10, 28]. Nodes with maximum degree or id among all neighbors can serve as leaders in multiple leader based MIS construction, thus the corresponding algorithms have lower message complexity [6, 25].

Note that the algorithm provided by [58] also relies on an MIS. But we will address it in detail in Subsection 4.6, as the major contribution in [58] is the exploitation of a Steiner tree with minimum number of Steiner points to connect an MIS.

4.3.1 Alzoubi and Wan's Single Leader Algorithm

Alzoubi and Wan's algorithms [4, 5, 69] utilize the properties of unit-disk graphs (UDGs) to prove their performance bounds. By definition, an MIS is adjacent to every node in the graph. Due to the geographic constraints imposed by a UDG, a node is adjacent to at most five independent neighbors [55]. Therefore, an arbitrary MIS can contain no more than five times the number of nodes in the minimum-sized MIS. This observation forms the basis of the proof of the performance bounds for all algorithms in [4, 5, 69].

Alzoubi *et al.* [4, 5, 69] provide two versions of an algorithm to construct the dominating set for a wireless network. In both algorithms, they first employ the distributed leader election algorithm [28] to construct a rooted spanning tree from the original network topology. Then, an iterative labelling strategy is used to classify the nodes in the tree to be either black (dominator) or gray (dominatee), based on their ranks. The rank of a node is the ordered pair of its level (number of hops to the root of the spanning tree) and its ID.

The labelling process begins from the root node and finishes at the leaves. The node with the lowest rank marks itself black and broadcasts a DOMINATOR message. The marking process then continues according to the following rules:

- If the first message that a node receives is a DOMINATOR message, it marks itself gray and broadcasts a DOMINATEE message.

- If a node received DOMINATEE messages from all its lower rank neighbors, it marks itself black and sends a dominator message.

The marking process finishes when it reaches the leaf nodes. At that time, the set of black nodes form an MIS, incorporating alternate levels of the spanning tree. Since the nodes are on alternating levels of the spanning tree, the distance between any subset of the MIS and its complement is exactly two hops away.

The final phase connects the nodes in the MIS to form a CDS, using INVITE and JOIN messages. Initially, the root joins the CDS and broadcasts an INVITE message. The INVITE message is relayed to all two-hop neighbors out of the current CDS. When a black node receives the INVITE message for the first time, it joins the dominating tree together with the gray node, which relayed the message. It then initiates an INVITE message. The process terminates when all the black nodes join the CDS.

This algorithm has time complexity of $O(n)$, and message complexity of $O(n\log(n))$. The resulting CDS has a size of at most $8opt + 1$.

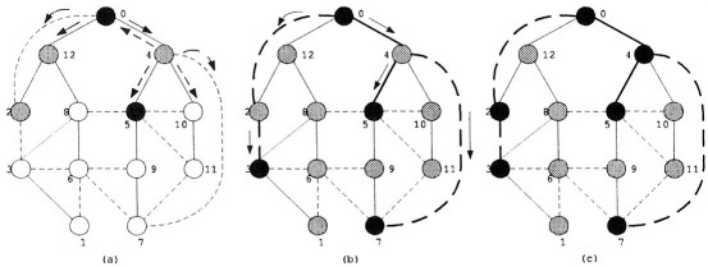


Figure 5: An example of Alzoubi and Wan's single leader algorithms for CDS construction.

Figure 5 illustrates the action of these algorithms. In this graph, node 0 is the root of the spanning tree that is constructed by using the leader election algorithm. The solid lines represent the edges of the rooted spanning tree, and the dashed lines represent other edges in the UDG. Node 0 is marked black first and broadcasts a DOMINATOR message (solid arrows in Figure 5(a)). After receiving this message, nodes 2, 4, and 12 are marked gray and broadcast DOMINATEE messages. (For simplicity, only the DOMINATEE messages from node 4 are shown as the dashed arrows in Figure 5(a)). Then node 5 is selected to be a DOMINATOR as it has received DOMINATEE messages from all its lower rank neighbors (node 4 only). Figure 5(b) shows the colors of the nodes when the labelling process finishes. The final process

builds the dominating tree from the root. The INVITE message (solid arrow in Figure 5(b)) is sent from node 0, and it is relayed to its two-hop black neighbors 3, 5 and 7. These black nodes join the dominating tree, as well as their relaying gray nodes 2 and 4. The thick links in Figure 5(b) illustrate the edges in the final dominating tree. All the nodes in the tree form a connected dominating set. The improved approach in [5] merges the MIS construction with the dominating tree building processes. As shown in Figure 5(c), node 2 is colored black when it first receives a DOMINATOR message from its child 3, and node 4 is marked black for the same reason. Finally, all the black nodes form a connected dominating set.

4.3.2 Cheng's Single Leader Algorithm

Cheng *et al.* [13, 16, 25, 23] present two algorithms for growing a connected dominating set from a leader node. Compared with the work of Alzoubi *et al.* [4, 5, 69], they introduce a new ***active*** state for vertices to describe the current labelling set of vertex nodes. With the help of this new concept, either cost-aware or degree-aware optimization can be achieved.

Their first algorithm is cost-aware. Each host has a local cost, which serves as the selection criterion together with its ID. At the beginning, all vertices are in initial state with white color. The leader starts the algorithm by marking itself black and becoming a dominator. A white node goes to be a dominatee (gray) if one of its neighbors becomes a dominator. A non-active white vertex changes to status ***active*** if one of its neighbors becomes a dominatee. Its color still keeps white. Then, an ***active*** node with the smallest cost among all its active neighbors will compete to be a dominator. Its minimum cost gray parent also changes to serve as its dominator (black), ensuring the connectivity of the dominating tree. Finally, all black leaf nodes can change back to be dominatees (gray). This process terminates when all nodes are colored gray or black, and all the black nodes form a connected dominating set.

This algorithm has the time complexity of $O(n)$, and the message complexity of $O(n \log n)$, which is dominated by leader election. The performance ratio is $8opt + 1$, the same as that of the algorithm in [4].

An example that illustrates the application of Cheng's single leader algorithm is shown in Figure 6. There are 9 hosts and 12 links. We assume host IDs are the costs. Host 0 is the leader. In the beginning, node 0 is colored black, serving as a dominator. Nodes 1 and 5 are then colored gray. Nodes 2 and 6 become active. Their competition results in the winner of node 2. Node 2 colors itself black and invites node 1 to be its dominator.

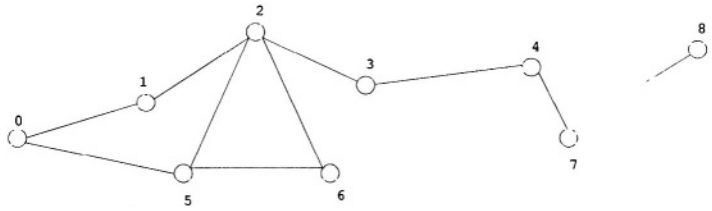


Figure 6: An example of unit-disk graph G containing 9 hosts and 12 links.

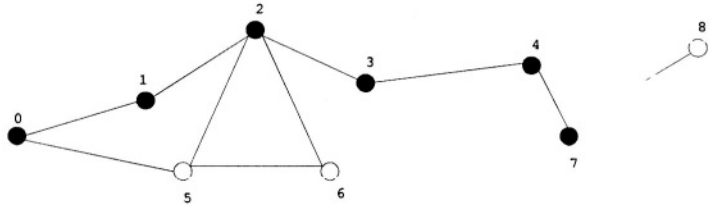


Figure 7: The computed connected dominating set from Cheng's single leader algorithm contains hosts {0,1,2,3,4,7}. The optimal solution contains {1,2,3,4,7}

Therefore node 3 is colored gray and node 4 becomes active. This process continues until no white nodes left. The result is demonstrated in Figure 7. All black nodes form a CDS.

Note that in this algorithm, a dominating tree is grown from the leader. The resultant CDS contains two subsets: one changes color from white to black directly, and the other colors themselves from white to gray then to black. If the determination of the second subset is delayed until the first subset is constructed, and the criterion for changing color from gray to black is based on the number of black neighbors a gray node has, the degree-based algorithm described in [23] is obtained. This algorithm has performance ratio 8. However, since effective degree information (number of white neighbors) needs to be updated during the algorithm execution, the degree-aware algorithm takes higher number of messages compared to the cost-aware algorithm, even though their time and message complexities are the same. Min and Du [57] extends the second algorithm to consider reliable virtual backbone construction in MANET.

4.3.3 Alzoubi's Multiple Leader algorithm

Single leader based CDS construction has message complexity $\Omega(n\ log\ n)$, which is dominated by leader election. In single leader CDS construction, the dependence on a rooted tree renders the maintenance of the CDS extremely difficult. To decrease the protocol overhead caused by the large volume of message exchange, and to improve the structure of the computed CDS, multiple leader based algorithms [6, 25] are proposed. Compared with single leader based CDS construction algorithms, the maintenance of the CDS constructed based on multiple leaders may be faster, as it does not rely on a spanning tree. In this subsection, we will study the algorithm introduced by Alzoubi *et al.* [6]. In next subsection, the algorithm provided by Cheng *et al.* [25] will be elaborated.

The algorithm proposed by Alzoubi *et al.* [6] constructs a CDS in a UDG with size at most $192 \cdot |OPT| + 48$. The message complexity is $O(n)$. This algorithm does not use a rooted spanning tree. Initially all the nodes are candidates. Whenever the ID of a node becomes the smallest among all of its one-hop neighbors, it will change its status to dominator. Consequently, its candidate neighbors become dominatees. After all nodes change status, each dominator node identifies a path of at most three hops to another dominator with larger ID. The candidate nodes on this path become connectors. All dominators and connectors compose a connected dominating set.

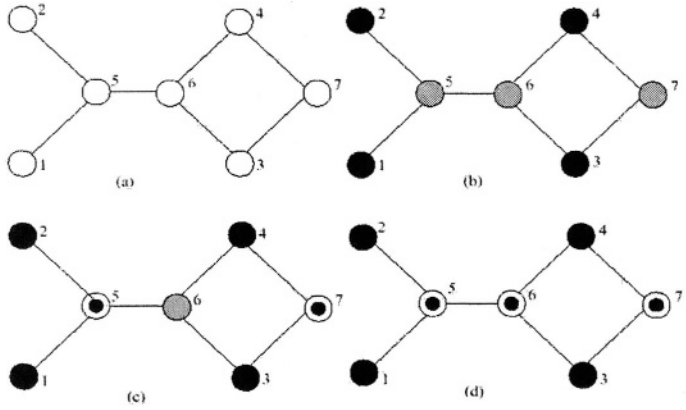


Figure 8: An example of Message-Optimal CDS Construction [6].

The example in [6] can illustrate the execution of this algorithm. As shown in Figure 8(b), nodes 1, 2, 3 and 4 declare themselves to be domi-

nators in the beginning. Then node 5 declares itself as a connector on the paths from 1/2 to 3/4, so do nodes 6 and 7. Finally all the connectors and dominators form a connected dominating set.

4.3.4 Cheng's Multiple Leader Algorithm

Cheng's multiple leader algorithm [25] is motivated by [6]. Initially each node with smallest ID among its one-hop neighbors becomes a leader. A forest with each tree rooted at a leader is constructed first. Then a chain of one or two nodes are computed to connect neighboring trees. This algorithm has linear time and message complexities, and generates a CDS with size at most $147 \cdot |OPT| + 33$. The performance analysis explores the neighboring property of nodes in a tree, thus the algorithm achieves better performance ratio compared to the multiple leader algorithm in [6]. This algorithm achieves linear time and message complexities.

4.3.5 Single Leader vs. Multiple Leader

Although single leader based CDS construction algorithms provide better performance bounds, their non-localized construction may render them unusable. Since each algorithm has a time complexity of $O(n)$, nodes may move during CDS construction such that the result of the algorithm is not a CDS. Thus, the ultimate goal of the CDS construction is to develop truly localized algorithms that have constant time complexity. Since the worst-case time complexity of the two multiple leader based algorithms [6, 25] is $O(n)$ due to the MIS or forest construction, these algorithms are not purely localized in a strict sense.

Nonetheless, multiple leader based algorithms in [6, 25] do represent progress in message complexity - they achieve the optimal message complexity of $O(n)$. Message complexity of a CDS construction algorithm contributes to the protocol overhead. It also plays an important role in the effectiveness and efficiency analysis of the protocol. Single leader based CDS construction algorithms [4, 5, 69, 23, 16] have message complexity $O(n \log n)$. The direct distributed implementation of Guha and Khuller's algorithms have message complexities $O(n|C|)$ and $O(n|C| + m + nlog(n))$ [35, 33, 66]. These algorithms have better performance ratio. Therefore as a trade-off the performance ratios of [6] and [25] are much higher.

The two truly localized algorithms are presented by Wu and Li in [76], and by Adjih, Jacquet, and Viennot in [1], which will be discussed in Subsections 4.4 and 4.5. Both algorithms need two-hop neighborhood information.

Wu and Li's algorithm first creates a larger CDS by selecting more nodes than needed, then prunes the set of selected nodes to get a CDS with smaller size. The algorithm proposed by Adjih, Jacquet, and Viennot relies on a multipoint relaying set.

4.4 Pruning Based CDS Construction

There are two pruning-based CDS construction algorithms [14, 76] proposed in the context of ad hoc and sensor networks. We will study them in the following subsections.

4.4.1 Wu and Li's Algorithm

Wu *et al.*'s work [76, 31] proposes a completely localized algorithm to construct CDS in general graphs. Initially all vertices are unmarked. They exchange their open neighborhood information with their one-hop neighbors. Therefore each node knows all of its two-hop neighbors. The marking process uses the following simple rule: any vertex having two unconnected neighbors is marked as a dominator. The set of marked vertices form a connected dominating set, with a lot of redundant nodes. Two pruning principles are provided to post-process the dominating set, based on the neighborhood subset coverage. A node u can be taken out from S, the CDS, if there exists a node v with higher ID such that the closed neighbor set of u is a subset of the closed neighbor set of v. For the same reason, a node u will be deleted from S when two of its connected neighbors in S with higher IDs can cover all of u's neighbors. This pruning idea is generated to the following general rule [32]: a node u can be removed from S if there exist k connected neighbors with higher IDs in S that can cover all u's neighbors. Wu *et al.* extend their work to calculate power-aware connected dominating sets [72, 77], by considering the power property for all the nodes as a criterion for the post pruning.

This idea is also extended to directed graphs. Due to differences in transmission ranges, or the hidden terminal problem [68] in wireless networks, some links in an ad hoc network may be unidirectional. In order to apply this algorithm to a directed graph model, neighboring vertices of a certain node are classified into a dominating neighbor set and an absorbent neighbor set in terms of the directions of the connected edges [70]. Figure 10 illustrates the dominating and absorbent neighbor sets of vertex u. In this case, the objective is to find a small set that is both dominating and absorbent for a given directed graph. The original marking process is adapted

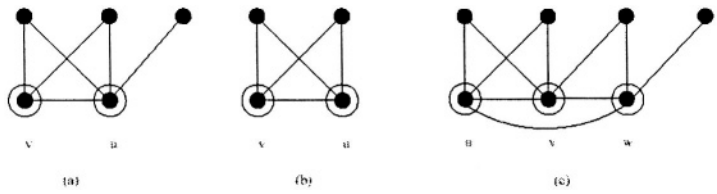


Figure 9: Examples of two pruning principles to eliminate redundant nodes [72].

as follows: a node u is added into the dominating and absorbent set, when there exists a node w in its dominating set and another node v in its absorbent set which can only be connected via u. Then similar post-process principles are used to delete the redundant nodes in the resulted dominating and absorbent set.

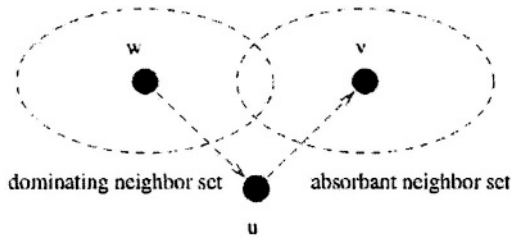


Figure 10: Dominating (absorbent) neighbor set of vertex u [70].

The performance ratio for Wu's algorithm is proved to be $O(n)$ [69]. For the post-processing, the nodes need to know their two hop neighbors. Thus the algorithm has time complexity of $O(\Delta^3)$ and message complexity of $\Theta(m)$, which indicate that the maintenance of the CDS constructed by Wu's algorithm is relatively easier. In fact one major advantage of Wu and Li's algorithm is its locality of maintenance. Mobile hosts may switch off or on at any time for power efficiency. In addition to this switching, CDS maintenance may be required due to mobile hosts' movements. Compared to other CDS algorithms, Wu's algorithm only requires neighbors of a node to update their status when it changes switching status or location.

4.4.2 Butenko's algorithm

The centralized version of this algorithm [14, 15] is discussed in Subsection 3.5. Its distributed implementation has higher message complexity because global connectivity needs to be checked when examining each node. This can be done through any distributed spanning tree algorithm [9].

The algorithm starts from the node with minimum degree, which can be found by modified leader-election algorithms in [28]. Let u be the node under consideration at the current step. If removing u causes the CDS disconnected, then we color u black. u then selects its non-black neighbor with minimum effective degree (the number of non-black neighbors within the set) for consideration in next step. If it is OK to remove u, u will select a neighbor with minimum effective degree, if u does not have a black neighbor, for next step. If u does have a black neighbor v, then v will choose its neighbor with minimum effective degree for next step. This procedure will be continued until all nodes have been examined.

4.5 Multipoint Relaying Based CDS Construction

Multipoint relaying is a technique to allow each node u select a minimum forwarding set [61, 51,17] from $N(u)$ to cover $N(N(u))$. Finding a *multipoint relay set (MRS)* with minimum size is NP-Complete [61]. Refs. [61] and [51] independently design different $\log n$ factor greedy heuristics where n is the total number of hosts. Ref. [17] provides a sophisticated approximation algorithm with constant performance ratio 6. Multipoint relaying is mainly used for flooding control to decrease protocol overhead [46]. Recently Adjih, Jacquet, and Viennot [1] propose a localized heuristic to generate a CDS based on multipoint relaying. Their idea is sketched below.

Each node first compute a multipoint relay set, a subset of one-hop neighbors that can cover all the two-hop neighbors. The a CDS is constructed by two rules. The first rule puts all nodes with smallest ID among their neighbors into the CDS; The second rule puts a node into the CDS if the node is a member of the MRS of its smallest ID neighbor. Wu [71] enhanced the first rule by selecting the node that has at least two disconnected neighbors and has the smallest ID among all its neighbors. Chen and Shen [22] claims that by considering node degree instead of ID, the constructed CDS may have smaller size. The correctness of this heuristic is proved in [1] but no performance analysis is available.

4.6 Steiner Tree Based CDS Construction

The second algorithm proposed in Guha and Khuller [44] is Steiner tree based. Its distributed implementation [35, 33, 34] applies a spanning tree to approximate the computation of Steiner points. The single leader based algorithms proposed in [2, 3, 4, 5, 16, 23, 69] also compute Steiner points to connect the MIS. These algorithms have been discussed in Subsections 4.2 and 4.3, thus they will not be repeated here. In this subsection, we will study how the centralized algorithm proposed in [58] is implemented in a distributed fashion.

As mentioned by Subsection 3.4, Min's CDS construction algorithm applies Steiner tree with minimum number of Steiner points to connect an MIS with the following property: any subset of the MIS has hop distance two with its compliment. We need to consider two phases in the distributed implementation: the construction of the MIS and the computation of the Steiner points.

Cheng *et al.* [25] and Wan *et al.* [69] both imply heuristics to compute an MIS in a distributed fashion. Here we rephrase the simple idea based on [25]. Assume all nodes are white initially. In the first step, color the node with smallest id black. Color all its one-hop neighbors gray; color its two-hop neighbors yellow. At each step, choose a yellow node with smallest id among all its yellow neighbors and color it black. Color its one-hop white/yellow neighbors gray; color its two-hop white neighbors yellow. Repeat this procedure until all nodes are colored either black or gray. All black nodes form an MIS.

The distributed computation of the Steiner tree with minimum number of Steiner points is non-trivial, as demonstrated by [58]. The idea is stated below: each gray node u (a node not in the MIS) keeps track of the neighboring connected black components. Its *competitor* set includes all its gray neighbors and all the gray nodes adjacent to its neighboring connected black components. u becomes a Steiner point if and only if it is adjacent to the maximum number of black components among all its competitors. It is clear that this distributed implementation has very high message complexity.

4.7 Proactively Considering Nodal Mobility in CDS Construction

Most of the previous algorithms have attempted to address nodal movements by incorporating a maintenance phase in which the CDS can be reconstructed when nodes move. However, in the case of high nodal mobility,

this reactive approach may not yield a stable infrastructure. A number of DS algorithms have been developed that attempt to proactively utilize mobility information in an attempt to create stable clusters.

The Mobility Adaptive Clustering Algorithm, for example, selects slow moving nodes for inclusion in a WCDS [11]. When nodes move randomly, a fast moving clusterhead is likely to encounter another clusterhead sooner than a slow moving one. Furthermore, the open neighbor sets of fast moving nodes will exhibit more change than those of slow moving nodes. Therefore, this algorithm selects slow moving nodes, which are more likely to have stable links.

Mobility-Based Clustering also considers nodal movement in the creation of a WCDS [8]. This WCDS approach has three notable features:

- It uses relative mobility information to create clusters.
- It allows cluster members to be L hops from the clusterhead, where $L \geq 1$, in order to increase the stability of clusters.
- Rather than seeking a minimal WCDS, it allows for clusterheads to come into contact with each other if the clusterheads are going in different directions.

In this WCDS protocol, neighbors periodically exchange their position and velocity information with their neighbors. A node calculates the relative velocity and the relative mobility between itself and all of its neighbors. Relative mobility is an average of the magnitude of the relative velocity vector in the recent past. Once relative mobility has been measured, there is an initial cluster creation where, a node elects as its clusterhead its neighbor with the lowest id, whose relative mobility falls below a user-defined threshold. As this step is performed, a clusterhead which has been elected, may join another cluster, with the two clusters then merging into one, subject to the L-hop rule. Cluster maintenance is done sparingly, only when a node loses contact with its clusterhead, and encounters a node whose relative mobility is less than the threshold.

(α,t)-Clusters takes a similar approach in the creation of a WCDS. It requires that nodes in a cluster be mutually reachable after time t with a probability of α [56]. In their simulations, nodes move in a random walk-based mobility. They derive the probability that two nodes are mutually reachable based on this mobility pattern, and current mobility information.

The Clustering for Open Inter-vehicle communication Networks (COIN) algorithm arises because clustering protocols designed random mobility patterns perform poorly for inter-vehicle communication [12]. This algorithm

was tested with nodal mobility generated from microscopic vehicular traffic simulators. Vehicular mobility is highly constrained by the layout of road network, by traffic control devices, and by surrounding vehicles. Vehicle movement is characterized by high rates of speed, producing very high relative velocities. Driver behavior has a significant impact of mobility patterns both in the near term and in the long term. In the near term, vehicle movements vary dramatically based on individual lane changing, braking, and passing behaviors. In the long term, mobility is affected by the variations in the intended destination of a driver. The COIN algorithm enhances the prediction of future mobility by incorporating driver intentions into the prediction algorithm. The destination of a driver could be gleaned either from an on-board route guidance system, or through a statistical analysis of previous trips on the current roadway. The paper also identifies a source of instability in clustering - oscillatory inter-vehicle distances between vehicles with low relative mobility. Examples of this phenomenon can be found in stop and go traffic, as vehicles pass through four-way stop signs, or as vehicles slow to navigate curves in the roadway. In order to accommodate this phenomenon the algorithm relaxes the condition that no two clusterheads with low relative mobility be within radio contact. Instead, a design parameter, the inter-cluster minimum distance, specifies the minimum distance between two clusterheads.

5 Analysis of Distributed CDS Algorithms

The algorithms presented in the previous section represent an evolution of distributed CDS algorithms designed for use in a variety of wireless applications. This evolution has been driven by both improvements in efficiency and stability and by the design requirements of the consumer applications. The following analysis presents the evolution at these two levels. First, we describe the development of a toolbox of techniques for CDS construction. Then, at a higher level, we elicit the manner in which the needs of the consumer applications dictate the selection of specific techniques from this toolbox.

Since non-localized algorithms rely on global information, their time complexity is at least $O(n)$ as in [33]. Purely localized CDS construction algorithms operate much quicker, with a complexity related to the maximal degree, typically $O(\Delta^3)$ [1, 76]. This quicker execution time comes at a cost of a larger CDS.

5.1 A Toolbox for CDS Construction

5.1.1 WCDS Construction Techniques

WCDS construction plays an important role not only in clustering algorithms, but also in a number of CDS construction algorithms, which begin with the selection of a WCDS. The first WCDS construction algorithm selected the lowest ID neighbor as clusterhead [40]. In newer algorithms, techniques have been developed to reduce the size of the clusterhead set, improve the stability of the clusterhead set, and promote fairness in clusterhead selection.

The size of the WCDS can be reduced by choosing nodes with the highest degree as clusterheads [43]. Although this will reduce the size of the WCDS, if a network with mobile nodes, this may increase WCDS instability since nodal degree varies while ID does not.

Since stability of the WCDS is a key factor in many applications, clusterhead stability has been improved through preservation of clusterhead elections and multi-hop clusters. Clusterhead selection need not change whenever the topology of the network changes; rather, reelections might only occur when clusterheads move into range of each other or when a node moves out of range of all clusterheads [27]. Multi-hop clusters can improve cluster stability by increasing the area covered by a single clusterhead [7].

Fairness may also be a problem with Lowest ID clustering, since the burdens of being a clusterhead are primarily borne by those nodes with lower ID's. However, mitigating this shortcoming should not be done in a way that decreases stability, as in highest connectivity clustering without clusterhead election preservation. For example, Min-Max D-Clusters attempted to address this issue by selecting both low and high id nodes as clusterheads [7].

5.1.2 CDS Construction Techniques

We have discussed example algorithms exploiting the following distributed CDS construction techniques: greedy, MIS based, Steiner tree based, pruning based, and multipoint relaying. We briefly discuss them in the following.

Das *et al.*'s greedy algorithms [35, 33, 66] are the distributed implementations of Guha and Khuller's algorithms in [44]. These heuristics have high message complexity due to the global selection of the greedy choice. They are the first distributed ones proposed for MCDS computation in the context of wireless ad hoc networks. One feature of their scheme is to store the global topology information only in dominating nodes. This reduces access

and update overheads for routing. However, their CDS construction requires two hop neighborhood knowledge. The generated CDS has high approximation ratio and high implementation complexities (in message and time). In addition, it is not clear in their algorithm description how each individual node is informed on when to start the second stage. The CDS maintenance is expensive too, as their approaches need to maintain a spanning tree.

The MIS based algorithms [6, 25, 23, 69] compute and connect an MIS. Their performance analysis in unit-disk graphs take the advantage of the relationship between an MIS and OPT. Algorithms in this category usually have good performance bound and time/message complexities. They only need one-hop neighborhood information. However, the single leader based algorithms [13, 16, 25, 23] require leader election. This drawback makes them difficult to support localized CDS maintenance. Multiple leader based algorithms [6, 25] are optimal in message complexity. Compared to single leader algorithms, they are relatively more practical in local maintenance since they obviates the rooted spanning tree construction.

Pruning based algorithms [14, 76] prune a large CDS. Wu and Li's algorithm [76] is the first purely localized CDS construction heuristic. Butenko *et al.*'s algorithm [14] has high message complexity due to the global connectivity checking.

Wu and Li's algorithm [70, 76, 72] is very simple. The localized property makes the CDS maintenance easier. However, there is no performance analysis in the original paper [76, 72], which incorrectly analyzes the algorithm's time complexity. Ref. [4] corrects the mistake in [76], and proves that Wu and Li's algorithm has a linear performance ratio. This algorithm needs at least two-hop neighborhood information. It is presented based on the general graph model, and is extended to directed graph [70]. This is important for wireless network, as either the disparity of transmission range or the hidden terminal problem in physical wireless networks can cause unidirectional links.

The Steiner tree based CDS construction heuristic in [58] uses a Steiner tree with minimum number of Steiner points to connect a dominating set, usually an MIS. Computing Steiner points requires large number of message exchange.

The multipoint relaying based heuristic [1] is pure localized. This algorithm selects CDS from a multipoint relay set. No complexity analysis for this algorithm in literature.

We summarize the above analysis for major algorithms in the following table.

	Graph modal	Approx. ratio	Time complexity	Msg. complexity	Ngh. info.	Maintenance
[33]-I	general	$2H(\Delta)+1$	$O((n+\|C\|)\Delta)$	$O(n\|C\|+m+n\log(n))$	2-hop	non-local
[33]-II	general	$2H(\Delta)$	$O(\|C\|(\Delta+\|C\|))$	$O(n\|C\|)$	2-hop	non-local
[4]-I	UDG	$8opt+1$	$O(n)$	$O(n\log(n))$	1-hop	non-local
[69]	UDG	$8opt$	$O(n)$	$O(n\log(n))$	1-hop	non-local
[6]	UDG	$192opt+48$	$O(n)$	$O(n)$	1-hop	non-local
[16]	UDG	$8opt$	$O(n)$	$O(n\log(n))$	1-hop	non-local
[25]	UDG	$147opt+33$	$O(n)$	$O(n)$	1-hop	non-local
[76]	general	$O(n)$	$O(\Delta^3)$	$\Theta(m)$	2-hop	local
[58]	UDG	$6.8opt$	-	-	2-hop	non-local
[1]	general	-	$O(\Delta^3)$	$\Theta(m)$	2-hop	local

Table 1: Performance comparison for distributed CDS construction algorithms in [33, 4, 69, 6, 16, 25, 76, 58, 1]. Here n and m are the number of vertices and edges respectively; opt is the size of any optimal MCDS for the given instance; Δ is the maximum degree; $|C|$ is the size of the computed CDS.

5.2 Selection of CDS Construction Methods

The design goals of the application dictate the selection from the different techniques for creating a virtual network infrastructure. Depending on the needs of the application, either a WCDS or CDS will be appropriate. The application requirements will also dictate the balance between bounded performance and fast operation. Finally, requirements for stability will affect the size and characteristics of the dominating set, as well as the node selection methodology.

The selection of a WCDS or a CDS depends on the communication requirements between nodes within the dominating set. Routing applications tend to rely on CDS, since messages must be forwarded along a backbone. However, if the network topology is highly unstable, a WCDS may be a better choice since its smaller size makes it easier to maintain. Likewise, while intra-cluster coordination functions might be managed within a WCDS, inter-cluster coordination is probably more easily handled by a CDS. So if adjacent clusters are using orthogonal codes or different frequency bands, a WCDS could manage media access within each cluster. However, for a system-wide media access coordination, a CDS may be more efficient since it includes nodes needed for clusterhead communication.

In addition to selecting the underlying virtual network infrastructure, the application developer must select the appropriate balance between performance bounds and fast operation. As seen in the previous examples, significantly better performance bounds are achievable if the dominating set is constructed serially as opposed to in parallel. However, the application may not be able to accommodate the additional time required to construct the dominating set. Moreover, with mobile nodes, then by the time the dominating set is constructed, the network topology may have changed such that the result of the algorithm is not a dominating set.

In addition to performance considerations, the stability requirements of the dominating set are also crucial for applications that require long-lasting dominating sets for mobile nodes. The stability of dominating set can be measured either from the perspective of a dominating set or from the perspective of a node outside of the dominating set. One may seek to reduce the rate of CDS change. Alternatively, one may seek to increase the duration that a node is associated with a given node in the dominating set. An example of the latter would be to increase the time that a node is in the cluster of a given clusterhead.

Depending on the definition of stability, different techniques can be employed. If one seeks to promote the stability of the CDS, then the selection

of low mobility nodes as members of the CDS is suitable approach.

However, if one also seeks to promote the association stability between nodes and members of the CDS, then one must create these associations based on the predictions of relative mobility between nodes. Furthermore, these predictions are domain-specific - predictions that reflect random mobility are likely to be unsuitable for domains where mobility is constrained.

A common technique in many of these algorithms is enlarging the size of the dominating set. This achieves greater stability by relaxing the constraints on the number of elements in a dominating set that are within radio range of each other. For the vehicular mobility case, for example, one ends up with at least two dominating sets - one for each direction of vehicle traffic. The overlap in these sets allows for both stability of the CDS, and stability of node association with clusterheads.

6 Conclusions and Discussions

Dominating sets have proven to be an effective construct within which to solve a variety of problems that arise in wireless networks. Applications that use dominating sets include media access coordination, unicast and multicast routing, and energy efficiency.

The needs of these consumer applications drive the design goals of the dominating set construction algorithms. In our review of these algorithms, we have examined a number of these considerations. The most obvious is the choice of target set - a WCDS, CDS, or a d-hop dominating set. Furthermore, the large differences in time and message complexity are the result of tradeoffs between fast operation and bandwidth efficiency versus dominating set size. In addition, these algorithms may need to address nodal mobility, if the dominating set is not transitory. Some algorithms have attempted to address this problem through the creation of localized maintenance routines that seek to avoid CDS recreation through a rerunning of the construction algorithm. Others have also included proactive consideration of nodal mobility in order to promote CDS stability.

As ad hoc wireless networking moves from the research labs to the real world, additional requirements will certainly arise. Issues that will likely arise include the security and survivability of CDS-based applications. In infrastructure-based wireless networks, the infrastructure plays a key role for security purposes. A perimeter is established around the infrastructure, regulating access to the network. A CDS could perform similar actions through forming a virtual network infrastructure. However, significant issues must

first be addressed including the real-time assurance of the trustworthiness of mobile hosts. Similarly, performance of the DS algorithms in the face of Denial of Service and other security attacks must be evaluated in order to ensure robust, survivable network functionality.

References

[1] C. Adjih, P. Jacquet, and L. Viennot, Computing connected dominated sets with multipoint relays, *Technical Report,* INRIA, Oct. 2002.

[2] K. M. Alzoubi, P.-J. Wan, O. Frieder, Distributed Construction of Connected Dominating Set in Wireless Ad Hoc Networks, to appear in *ACM Mobile Networks and Applications.*

[3] K. M. Alzoubi, P.-J. Wan, O. Frieder: Maximal Independent Set, Weakly Connected Dominating Set, and Induced Spanners for Mobile Ad Hoc Networks, *International Journal of Foundations of Computer Science,* Vol. 14, No. 2, pp. 287-303, 2003.

[4] K.M. Alzoubi, P.-J. Wan and O. Frieder, New Distributed Algorithm for Connected Dominating Set in Wireleess Ad Hoc Networks, *Proceedings of the 35th Hawaii International Conference on System Sciences,* Big Island, Hawaii, 2002.

[5] K.M. Alzoubi, P.-J. Wan and O. Frieder, Distributed Heuristics for Connected Dominating Sets in Wireless Ad Hoc Networks, *Journal of Communications and Networks,* Vol. 4, No. 1, Mar. 2002.

[6] K.M. Alzoubi, P.-J. Wan and O. Frieder, Message-Optimal Connected Dominating Sets in Mobile Ad Hoc Networks, *MOBIHOC,* EPFL Lausanne, Switzerland, 2002.

[7] A. D. Amis, R. Prakash, T. H. P. Vuong, and D. T. Huynh, Max-Min D-Cluster Formation in Wireless Ad Hoc Networks, Proc. IEEE INFOCOM, Tel Aviv, Israel, 2000.

[8] B. An and S. Papavassiliou, A mobility-based clustering approach to support mobility management and multicast routing in mobile ad-hoc wireless networks, *International Journal of Network Management,* 11, pp. 387-395, 2001.

[9] H. Attiya and J. Welch, Distributed computing: fundamentals, simulations and advanced topics, McGraw-Hill Publishing Company, 1998.

[10] B. Awerbuch, Optimal Distributed Algorithm for Minimum Weight Spanning Tree, Counting, Leader Election and Related Problems, *Proceedings of the 19th ACM Symposium on Theory of Computing, ACM,* pp. 230-240, 1987.

[11] S. Basagni, Distributed and mobility-adaptive clustering for multimedia support in multi-hop wireless networks, Proc. IEEE 50th Vehicular Technology Conference, Piscataway, NJ, 1999.

[12] J. Blum, A. Eskandarian, and L. Hoffman, Mobility Management of IVC Networks, Proc. IEEE Intelligent Vehicles Symposium, Columbus, OH, 2003.

[13] S. Butenko, X. Cheng, D.-Z. Du, and P.M. Pardalos, On the construction of virtual backbone for ad hoc wireless networks, In S. Butenko, R. Murphey, and P.M. Pardalos, editors, *Cooperative Control: Models, Applications and Algorithms,* pp. 43-54, Kluwer Academic Publishers, 2003.

[14] S. Butenko, X. Cheng, C. Oliveira, and P.M. Pardalos, A new heuristic for the minimum connected dominating set problem on ad hoc wireless networks, In S. Butenko, R. Murphey, and P.M. Pardalos, editors, *Recent Developments in Cooperative Control and Optimization,* pp. 61-73, Kluwer Academic Publishers, 2004.

[15] S. Butenko, C. Oliveira, and P.M. Pardalos, A new algorithm for the minimum connected dominating set problem on ad hoc wireless networks, *Proc. of CCCT'03,* Vol. V, pp.39-44, International Institute of Informatics and Systematics (IIIS), 2003.

[16] M. Cadei, X. Cheng and D.-Z. Du, Connected Domination in Ad Hoc Wireless Networks, *Proc. 6th International Conference on Computer Science and Informatics,* 2002.

[17] G. Calinescu, I. Mandoiu, P.-J. Wan and A. Zelikovsky, Selecting forwarding neighbors in wireless ad hoc networks, *manuscript,* 2001.

[18] Y.-L. Chang and C.-C. Hsu, Routing in wireless/mobile ad-hoc networks via dynamic group construction, *Mobile Networks and Applications,* 5, pp. 27-37, 2000.

[19] B. Chen, K. Jamieson, H. Balakrishnan, and R. Morris, Span: An energy-efficient coordination algorithm for topology maintenance in Ad Hoc wireless networks, *Wireless Networks,* 8, pp. 481-494, 2002.

[20] D. Chen, D.-Z. Du, X. Hu, G.-H. Lin, L. Wang, and G. Xue, Approximations for Steiner trees with minimum number of Steiner points, *Journal of Global Optimization,* Vol. 18, pp. 17-33, 2000.

[21] Y. Chen, A. Liestman, Approximating minimum size weakly-connected dominating sets for clustering mobile ad hoc networks, *ACM MobiHoc,* June 2002.

[22] X. Chen and J. Shen, Reducing connected dominating set size with multipoint relays in ad hoc wireless networks *Proceedings of the 7th International Symposium on Parallel Architectures, Algorithms and Networks,* pp. 539 - 543, May 2004.

[23] X. Cheng, Routing Issues in Ad Hoc Wireless Networks, *PhD Thesis,* Department of Computer Science, University of Minnesota, 2002.

[24] X. Cheng, M. Ding, and D. Chen, An approximation algorithm for connected dominating set in ad hoc networks, *Proc. of International Workshop on Theoretical Aspects of Wireless Ad Hoc, Sensor, and Peer-to-Peer Networks (TAWN),* 2004.

[25] X. Cheng, M. Ding, D.H. Du, and X. Jia, On The Construction of Connected Dominating Set in Ad Hoc Wireless Networks, to appear in Special Issue on *Ad Hoc Networks* of *Wireless Communications and Mobile Computing,* 2004.

[26] X. Cheng, X. Huang, D. Li, W. Wu, and D.-Z.Du, Polynomial-Time Approximation Scheme for Minimum Connected Dominating Set in Ad Hoc Wireless Networks, *Networks,* Vol. 42, No. 4, pp. 202-208, 2003.

[27] C.-C. Chiang and M. Gerla, Routing and Multicast in Multihop, Mobile Wireless Networks, Proc. IEEE ICUPC'97, San Diego, CA, 1997.

[28] I. Cidon and O. Mokryn, Propagation and Leader Election in Multihop Broadcast Environment, *Proc. 12th Int. Symp. Distr. Computing,* pp. 104-119, Greece, Spt. 1998.

[29] B. N. Clark, C. J. Colbourn and D. S. Johnson, Unit disk graphs, *Discrete Mathematics,* Vol. 86, 1990, pp. 165-177.

[30] T. Clausen, P. Jacquet, A. Laouiti, P. Minet, P. Muhlethaler, and L. Viennot, Optimized link state routing protocol, IETF Internet Draft, draft-ietf-manet-olsr-05.txt, October 2001.

[31] F. Dai and J. Wu, An Extended Localized Algorithm for Connected Dominating Set Formation in Ad Hoc Wireless Networks, to appear in *IEEE Transactions on Parallel and Distributed Systems,* 2004.

[32] F. Dai and J. Wu, Distributed dominate pruning in ad hoc wireless networks, *Proc. ICC,* 2003.

[33] Bevan Das and Vaduvur Bharghavan, Routing in Ad-Hoc Networks Using Minimum Connected Dominating Sets, *International Conference on Communications,* Montreal, Canada, June 1997.

[34] Bevan Das, Raghupathy Sivakumar and Vaduvur Bharghavan, Routing in Ad Hoc Networks Using a virtual backbone, *IC3N'97,* pp.1-20, 1997.

[35] Bevan Das, Raghupathy Sivakumar and Vaduvur Bharghavan, Routing in Ad Hoc Networks Using a Spine, *International Conference on Computers and Communications Networks,* Las Vegas, NV. Sept. 1997.

[36] S. Datta and I. Stojmenovic, Internal Node and Shortcut Based Routing with Guaranteed Delivery in Wireless Networks, *Cluster Computing,* 5(2), pp. 169-178, 2002.

[37] Budhaditya Deb, Sudeept Bhatnagar, and Badri Nath, Multi-resolution State Retrieval in Sensor Networks, *First IEEE Workshop on Sensor Network Protocols And Applications (SNPA) ,* Anchorage 2003

[38] M. Ding, X. Cheng, and G. Xue, Aggregation tree construction in sensor networks, *Proc. of IEEE VTC,* 2003.

[39] D.-Z. Du, L. Wang, and B. Xu, The Euclidean bottleneck Steiner tree and Steiner tree with minimum number of Steiner points, *COCOON'01,* pp. 509-518, 2001.

[40] A. Ephremides, J. Wieselthier, and D. Baker, A design concept for reliable mobile radio networks with frequency hopping signaling, *Proceedings of the IEEE,* 75, pp. 56-73, 1987.

[41] M. R. Garey and D. S. Johnson, Computers and Intractability: A guide to the theory of NP-completeness, Freeman, San Frncisco, 1978.

[42] R. De Gaudenzi, T. Garde, F. Giannetti, and M. Luise, DS-CDMA techniques for mobile and personal satellite communications: An overview *IEEE Second Symposium on Communications and Vehicular Technology in the Benelux,* pp.113 - 127, 2-3 Nov. 1994.

[43] M. Gerla and J. T. Tsai, Multicluster, mobile, multimedia, radio network, *ACM/Baltzer Journal on Wireless Networks,* 1, pp. 255-265, 1995.

[44] S. Guha and S. Khuller, Approximation algorithms for connected dominating sets, *Algorithmica,* 20(4), pp. 374-387, Apr. 1998.

[45] A. Helmy, Efficient resource discovery in wireless adhoc networks: contacts do help, Manuscript, 2004.

[46] P. Jacquet, P. Muhlethaler, A. Qayyum, A. Laouiti, L. Viennot and T. Clausen, Optimized link state routing protocol, IETF Internet Draft, draft-ietf-manet-olsr-04.txt, March 2001.

[47] D.B. Johnson and D.A. Maltz, Dynamic source routing in ad hoc wireless networks, *Proc. Mobile Compiling* edited by T. Imielinski and H. Korth, Kluwer Academic Publisher, 1996.

[48] U. Kozat, L. Tassiulas, Service discovery in mobile ad hoc networks: an overall perspective on architectureal choices and network layer support issues, *Ad Hoc Networks Journal,* In press, 2003.

[49] B. Liang and Z.J. Haas, Virtual backbone generation and maintenance in ad hoc network mobility management, *IEEE INFOCOM* 2000, pp.1293-1302.

[50] J. Li, J. Jannotti, D. S. J. D. Couto, D. R. Karger, and R. Morris, A scalable location service for geographic ad hoc routing, Proc. of the sixth annual international conference on Mobile computing and networking, Boston, MA, 2000.

[51] H. Lim and C. Kim, Multicast Tree Construction and Flooding in Wireless Ad Hoc Networks, Proc. 3rd ACM international workshop on Modeling, Analysis and Simulation of Wireless and Mobile Systems, Boston, MA, pp. 61-68, 2000.

[52] H. Lim and C. Kim, Flooding in wireless ad hoc networks, *Computer Communications Journal,* Vol. 24 (3-4), pp. 353-363, 2001.

[53] G.-H. Lin and G.L. Xue, Steiner tree problem with minimum number of Steiner points and bounded edge-length, *Information Processing Letters,* Vol. 69, pp. 53-57, 1999.

[54] C. Lund and M. Yannakakis, On the hardness of approximating minimization problems, *Journal of the ACM,* Vol. 41(5), pp. 960-981, 1994.

[55] M. V. Marathe *et al.,* Simple heuristics for unit disk graphs, *Networks,* vol. 25, pp. 59-68, 1995.

[56] A. B. McDonald and T. F. Znati, Mobility-based framework for adaptive clustering in wireless ad hoc networks, *IEEE Journal on Selected Areas in Communications,* 17, pp. 1466-1487, 1999.

[57] M. Min and D.-Z. Du, A reliable virtual backbone scheme in mobile ad-hoc networks, *manuscript,* 2004.

[58] M. Min, C.X. Huang, S. C.-H. Huang, W. Wu, H. Du, and X. Jia, Improving construction of connected dominating set with Steiner tree in wireless sensor networks, submitted.

[59] S.-Y. Ni, Y.-C. Tseng, Y.-S. Chen and J.-P. Sheu, The broadcast storm problem in a mobile ad hoc network, *Proc. MOBICOM,* Seattle, Aug. 1999, pp. 151-162.

[60] C. Perkins and E. Royer, Ad-hoc on-demand distance vector routing, *Proc. 2nd IEEE Workshop on Mobile Computing Systems and Applications,* pp. 90-100, 1999.

[61] A. Qayyum, L. Viennot, and A. Laouiti, Multipoint relaying for flooding broadcast messages in mobile wireless networks, *HICSS02,* pp. 3866 - 3875, January 2002.

[62] L. Ruan, D.H. Du, X. Jia, W. Wu, Y. Li, and K.-I Ko A greedy approximation for minimum connected dominating sets, *manuscripts,* 2004.

[63] J. Shaikh, J. Solano, I. Stojmenovic, and J. Wu, New Metrics for Dominating Set Based Energy Efficient Activity Scheduling in Ad Hoc Networks, *Proc. of WLN Workshop (in conjunction to IEEE Conference on Local Computer Networks),* pp. 726-735, Oct. 2003.

[64] R. Sivakumar, P. Sinha and V. Bharghavan, CEDAR: a core-extraction distributed ad hoc routing algorithm, *Selected Areas in Communications, IEEE Journal on,* Vol. 17(8), Aug. 1999, pp. 1454 -1465.

[65] P. Sinha, R. Sivakumar and V. Bharghavan, Enhancing ad hoc routing with dynamic virtual infrastructures, *INFOCOM 2001,* Vol. 3, pp. 1763-1772.

[66] R. Sivakumar, B. Das, and V. Bharghavan, An Improved Spine-based Infrastructure for Routing in Ad Hoc Networks, *IEEE Symposium on Computers and Communications,* Athens, Greece, June 1998.

[67] I. Stojmenovic, M. Seddigh, J. Zunic, Dominating Sets and Neighbor Elimination Based on Broadcasting Algorithms in wireless networks, *Proc. IEEE Hawaii Int. Conf. on System Sciences,* 1988.

[68] A. Tanenbaum, *Computer Networks,* Prentice Hall, 1996.

[69] P.-J. Wan, K.M. Alzoubi and O. Frieder, Distributed Construction of Connected Dominating Sets in Wireless Ad Hoc Networks, *IEEE INFOCOM,* 2002.

[70] J. Wu, Extended Dominating-Set-Based Rounting in Ad Hoc Wireless Networks with Undirectional Links, *IEEE Trans. on Parallel and Distributed Systems,* pp. 866-881, Sept. 2002.

[71] J. Wu, An enhanced approach to determine a small forward node set based on multipoint relays *IEEE 58th Vehicular Technology Conference,* Vol. 4, pp. 2774 - 2777, October 2003.

[72] J. Wu, F. Dai, M. Gao, and I. Stojmenovic, On Calculating Power-Aware Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks, *Journal of Communications and Networks,* Vol. 5, No. 2, pp. 169-178, March 2002.

[73] J. Wu and F. Dai, Broadcasting in ad hoc networks based on self-pruning, *IEEE INFOCOM,* 2003.

[74] J. Wu and F. Dai, On Locality of Dominating Set in Ad Hoc Networks with Switch-On/Off Operations, *Journal of Interconnection Networks,* a special issue on *Algorithms for Networks,* Vol. 3, No. 3 & 4, pp. 129-147, Sept. & Dec. 2002.

[75] J. Wu and F. Dai, On Locality of Dominating Set in Ad Hoc Networks with Switch On/Off Operations, *Proc. of the 6th International Conference on Parallel Architectures, Algorithms, and Networks (I-SPAN02),* 2002.

[76] J. Wu and H. Li, On Calculating Connected Dominating Set for Efficient Routing in Ad Hoc Wireless Networks, *Proc. of the Third International Workshop on Discrete Algorithms and Methods for Mobile Computing and Communications,* pp. 7-14 , Aug. 1999.

[77] J. Wu, M. Gao, and I. Stojmenovic, On Calculating Power-Aware Connected Dominating Sets for Efficient Routing in Ad Hoc Wireless Net-

works, *Proc. of International Conference on Parallel Processing (ICPP),* 346-356, 2001.

[78] J. Wu and B. Wu, A Transmission Range Reduction Scheme for Power-Aware Broadcasting in Ad Hoc Networks Using Connected Dominating Sets, *Proc. of 2003 IEEE Semiannual Vehicular Technology Conference (VTC2003-fall),* Oct. 2003.

[79] J. Wu, B. Wu, and I. Stojmenovic, Power-Aware Broadcasting and Activity Scheduling in Ad Hoc Wireless Networks Using Connected Dominating Sets, *Proc. of IASTED International Conference on Wireless and Optical Communication (WOC 2002),* 2002.

[80] J. Wu, B. Wu, and I. Stojmenovic, Power-Aware Broadcasting and Activity Scheduling in Ad Hoc Wireless Networks Using Connected Dominating Sets, *Wireless Communications and Mobile Computing,* a special issue on *Research in Ad Hoc Networking, Smart Sensing, and Pervasive Computing,* Vol. 3, No. 4, pp. 425-438, June 2003.

[81] W. Wu, H. Du, X. Jia, M. Min, C.H. Huang, and X. Cheng, Maximal independent set and minimum connected dominating set in unit disk graphs, *manuscripts,* 2004.

[82] Y. Xu, J. Heidemann, and D. Estrin, Geography-informed energy conservation for Ad Hoc routing, MobiCom 2001, pp.70-84, 2001.

Author Index

Subject Index