A Reactive GRASP for a Commercial Territory Design Problem with Multiple Balancing Requirements

Roger Z. Ríos-Mercado[1]
Graduate Program in Systems Engineering
Universidad Autónoma de Nuevo León
AP 111 – F, Cd. Universitaria
San Nicolás de los Garza, NL 66450, México
E-mail: *roger@mail.uanl.mx*

Elena Fernández
Department of Statistics and Operations Research
Universitat Politècnica de Catalunya
Jordi Girona 1-3, C5-208, Campus Nord
Barcelona 08034, Spain
E-mail: *e.fernandez@upc.es*

September 2006
Revised July 2007

[1]Corresponding author

Abstract

In this paper we present a Reactive GRASP approach to a commercial territory design problem motivated by a real-world application in a beverage distribution firm. The mathematical framework includes, as planning criteria, minimizing a measure of territory dispersion, balancing the different node activity measures among territories and territory contiguity. The proposed GRASP approach incorporates several features such as reactivity, by allowing self-adjustment of the restricted candidate list quality parameter, and filtering, which avoids executing the local search phase in unpromising bad solutions generated by the construction phase. The algorithm has been tested in several data sets. The results show the effectiveness of the proposed approach. It was observed that the reactivity, and the filtering proved very useful in terms of feasibility with respect to the balancing constraints, and find more robust solutions when tested over the basic GRASP. The local search scheme proved to be very effective as well. Moreover, the proposed approach obtained solutions of much better quality (both in terms of its dispersion measure and feasibility with respect to the balancing constraints) than those found by the firm method in relatively fast computation times.

Keywords: combinatorial optimization, territory design, multiple balancing requirements, metaheuristics, reactive GRASP

1 Introduction

The sales territory design problem (TDP) may be viewed as the problem of grouping small geographic sales coverage units (SCUs) into larger geographic clusters, called sales territories, in a way that the territories are acceptable (or optimal) according to relevant planning criteria. This problem belongs to the family of districting problems that have a broad range of applications like political districting and the design of sales and services territories. The recent paper by Kalcsics, Nickel, and Schröder [14] is an extensive survey on approaches to territory design that gives an up to date state of the art and an unifying approach to the topic.

The problem addressed in this paper is motivated by a real-world application from a beverage distribution firm in the city of Monterrey, Mexico, and may be viewed as a sales TDP, where rather than placing salesmen in territories we are interested in locating centers for the purpose of modeling the compactness measure. To make a clear distinction between a sales TDP and our problem, we call ours a commercial TDP, meaning the interest is placed on providing customers with a commercial service by the firm. Although several sales territory design approaches have appeared in the literature [2, 7, 9, 11, 21], the specific features present in this concrete problem make it very unique, and not addressed before to the best of our knowledge. The firm wishes to partition the area of the city into disjoint territories that are suitable for their commercial purposes. In particular, the firm wants to design territories that are balanced (similar in size) with respect to each of three different activity measures (number of customers, product demand, and workload). There are in addition several requirements that are posed by the firm, which are derived from their planning criteria and from regulations and agreements with the work force. They are: (i) contiguity of each territory, so that SCUs can reach each other by traveling within the territory, (ii) territory compactness, so that customers within a territory are relatively close to each other, and (iii) a fixed number of territories.

The characteristics of the problem together with the fact that the number of territories is given have led us to model the problem as a p-center problem where we are interested in locating p centers, one for each territory. It is clear that for modeling this problem in principle it is not needed to associate a center with each territory. However, this gives us a simple tool for defining a compactness measure and for formulating the contiguity requirements. We chose a center (minsum) objective function rather than a median (sum) one, since this is the type of objective function that is typically used in discrete location when it is sought that no customer be "too" far away from the facility it is assigned to. In our opinion this gives us an appropriate measure for modeling compactness, even if one can always build examples where the median would also be suitable.

In this work, we present and discuss a modeling framework for this NP-hard combinatorial optimization problem, and we propose a solution algorithm based on GRASP (Greedy Randomized Adaptive Search Procedure) to construct high-quality solutions. We have taken a distance-based

dispersion measure of the territories as the objective for the proposed model.

The specification that the territories be simultaneously balanced with respect to all the three measures has been modeled by requiring that each territory be within a threshold of a target value for each activity measure. This is motivated by the fact that for a given instance a solution where all the territories are simultaneously balanced with respect to all three measures may not exist.

GRASP is a well-known metaheuristic that has been widely used for successfully solving many combinatorial optimization problems. Broadly speaking, GRASP usually gives a good compromise between the quality of the obtained results and the required computational effort. For the problem that we address we chose GRASP for several reasons: First, since the problem corresponds to a real application, we were interested in implementing a simple efficient algorithm that could be easily understandable by the firm and required as little tuning as possible, for future applications by the firm to possibly different data. Also, to the best of our knowledge, this type of approach has not been considered before for general territory design problems with distance-based dispersion measures.

It is true that more sophisticated heuristic methods (like Tabu Search, Scatter Search or Path Relinking) can outperform the basic versions of GRASP algorithms, but it is also true that they require more sophisticated data structures and their computational requirements tend to be higher. It is also true that the performance of basic GRASP versions can be improved by incorporating additional features that to some extent take into account specific characteristics of the problem or the history of the search process.

In our case the basic version of GRASP has been improved with several additional features. On the one hand, since the local search phase typically increases considerably the required computation time of any GRASP algorithm, we have included a filtering mechanism according to which the local search phase is only applied to those solutions that, based on the history of the search, seem likely to produce a good quality improved solution. On the other hand, we have implemented a reactive GRASP algorithm that, taking also into account the history of the process, does not require tuning the parameter that regulates the size of the restricted candidate list in the constructive phase, which is crucial to the success of the process. In addition, all the versions of the GRASP that we propose allow for strategic oscillation, since the capacity constraints are relaxed, and incorporated to the objective function via a penalty term.

We have run a series of computational experiments to assess the efficiency of the proposed GRASP algorithms, and to evaluate the behavior of our approach when compared to current industry practices. In particular, in these experiments we have analyzed the effect on the obtained solutions of different threshold values for the balancing constraints. Given that the most time consuming phase of the procedure is its local search, we have also studied the impact of applying a filter to avoid executing this phase in unpromising bad solutions generated by the constuction phase. The obtained results are very satisfactory since we obtain solutions of much better quality

than that of the solutions obtained by the firm method when applied to the same set of instances. The proposed approach runs in relatively small computation times.

The paper is structured as follows. In Sections 2 and 3 we describe the problem and we present the model that we follow, respectively. Section 4 gives an overview of some previous related work. Section 5 gives the elements of our algorithm, and Section 6 describes the experiments that we have run and the obtained results. We end the paper in Section 7, with some conclusions and final comments.

2 Problem Description

The problem is modeled by a graph $G = (V, E)$, where a city block or SCU i is associated with a node, and an arc connecting nodes i and j exists if blocks i and j are adjacent to each other. Now each node $i \in V$ has several associated parameters such as geographical coordinates (c_i^x, c_i^y), and three measurable activities. Let w_i^a be the value of activity a at node i, where $a = 1$ (number of customers), $a = 2$ (product demandd), and $a = 3$ (workload). A territory is a subset of nodes $V_k \subset V$. The number of territories is given by the parameter p. It is required that each node is assigned to only one territory. Thus, the territories define a partition of V. One of the properties sought in a solution is that the territories are balanced with respect to each of the activity measures. So, let us define the size of territory V_k with respect to activity a as: $w^a(V_k) = \sum_{i \in V_k} w_i^a$, $a = 1, 2, 3$. Due to the discrete structure of the problem and to the unique assignment constraint, it is practically impossible to have perfectly balanced territories with respect to each activity measure. To account for this, we measure the balance degree by computing the relative deviation of each territory from its average size μ^a, given by $\mu^a = w^a(V)/p$, $a = 1, 2, 3$. Another important feature is that all of the nodes assigned to each territory are connected by a path contained totally within the territory. In other words, each of the territories V_k must induce a connected subgraph of G. In addition, industry demands that in each of the territories, blocks must be relatively close to each other. One way to achieve this is for each territory to select an appropriate node to be its center, and then to define a distance measure such as $D = \max_{1 \leq k \leq p} \max_{j \in V_k} d_{c(k),j}$, where $c(k)$ denotes the index of the center of territory k so $d_{c(k),j}$ represents the Euclidean distance from node j to center of territory k. So maximizing compactness is equivalent to minimizing this dispersion function D. All parameters are assumed to be known with certainty. The problem can be thus described as finding a p-partition of V satisfying the specified planning criteria of balancing and contiguity, that minimizes the above distance-based dispersion measure.

3 MILP Formulation

Indices and sets

n	number of blocks (SCUs)
p	number of territories
i, j	block indices; $i, j \in V = \{1, 2, \ldots, n\}$
a	activity index; $a \in A = \{1, 2, 3\}$
k	territory index; $k \in K = \{1, 2, \ldots, p\}$
E	edge set of adjacent blocks
N^i	$(= \{j \in V : (i, j) \in E \vee (j, i) \in E\})$ set of nodes which are adjacent to node i; $i \in V$

Parameters

w_i^a	value of activity a in node i; $i \in V$, $a \in A$
d_{ij}	Euclidean distance between i and j; $i, j \in V$
τ^a	relative tolerance with respect to activity a; $a \in A$, $\tau^a \in [0, 1]$

Computed parameters

$w^a(B)$	$(= \sum_{j \in B} w_j^a)$ size of set B with respect to a; $a \in A$, $B \subset V$
μ^a	$(= w^a(V)/p)$ average (target) value of activity a; $a \in A$

Decision variables

In the original problem we are not concerned with territory centers; however, we introduce binary variables based on centers for modeling the dispersion measure.

$$y_i = \begin{cases} 1 & \text{if a territory center is placed at node } i;\ i \in V \\ 0 & \text{otherwise} \end{cases}$$

$$x_{ij} = \begin{cases} 1 & \text{if unit } j \text{ is assigned to territory with center in } i;\ i, j \in V \\ 0 & \text{otherwise} \end{cases}$$

Model

$$\text{Minimize} \qquad f(x, y) = \max_{i,j \in V} \{d_{ij} x_{ij}\} \tag{1}$$

$$\text{subject to} \qquad \sum_{i \in V} x_{ij} = 1 \qquad j \in V \tag{2}$$

$$\sum_{i \in V} y_i = p \tag{3}$$

$$\sum_{j \in V} w_j^a x_{ij} \leq (1+\tau^a)\mu^a y_i \quad i \in V,\ a \in A \tag{4}$$

$$\sum_{j \in V} w_j^a x_{ij} \geq (1-\tau^a)\mu^a y_i \quad i \in V,\ a \in A \tag{5}$$

$$\sum_{j \in \cup_{v \in S} N^v \setminus S} x_{ij} - \sum_{j \in S} x_{ij} \geq 1 - |S| \quad i \in V;\ S \subset V \setminus (N^i \cup \{i\}) \tag{6}$$

$$x_{ij}, y_i \in \{0,1\} \quad i,j \in V \tag{7}$$

Objective (1) measures territory dispersion. Constraints (2) guarantee that each node j is assigned to a territory. Constraint (3) sets the number of territories. Constraints (4)-(5) represent the territory balance with respect to each activity measure as it establishes that the size of each territory must lie within a range (measured by tolerance parameter τ^a) around its average size. In particular, the upper bound balancing constraints (4) also assure that if no center is placed at i, no customer can be assigned to it. Constraints (6) guarantee the connectivity of the territories. These constraints, proposed by Drexl and Haase [7], are similar to the constraints used in routing problems to guarantee the connectivity of the routes. Note that, as usual, there is an exponential number of such constraints.

The model can be viewed as a vertex p-center problem with multiple capacity constraints, and with additional constraints (5) and (6). Given that even the uncapacitated vertex p-center problem is $\mathcal{NP}$-hard [15], it follows that our commercial TDP is also $\mathcal{NP}$-hard.

4 Related Work

The grouping of small geographical units into larger geographical clusters according to some specified planning criteria is referred to in the literature as districting or territory design. Many authors have investigated districting problems and developed models and algorithms in several contexts, most importantly: design of sales territories [11, 9, 7, 14] and design of political districts [12, 10, 13, 16, 3, 1]. Other applications include turfing in telecommunications [19], police districting [5], districting for salt spreading operations [17], home-care districting [2], commercial territory design for commodity distribution [20] In Caro et al. [4], there is a very good review of school redistricting applications. Here, we present a review of the most relevant work in territory design.

4.1 Sales Territory Design

Hess and Samuels [11] address a sales TDP with workload balancing constraints, and compactness maximization criteria. As a compactness criteria they used squared Euclidean distances. They

present a heuristic based on a location-allocation scheme where a linear transportation problem is used to solve the assignment phase. Then, splits are resolved by means of a "tie breaking" heuristic which assigns an area to the territory with the maximum of the area's activity. In their empirical work, they find that a rate of $n/p \geq 20$ is more adequate for finding territories with activity measure within 10% of average. Connectivity was not considered.

Zoltners and Sinha [21] present the first review of sales territory design models. They develop a framework for sales territory alignment and several properties, which are incorporated into a general sales territory model.

Fleischmann and Paraschis [9] address a sales territory design problem arising in a German company for consumer goods. They formulate the problem as a MILP and develop a procedure based on a location-allocation approach. Specifically they have to allocate 168 sales agents in 1,400 postal areas. Planning criteria: (a) balancing workload (25% tolerance), (b) compact districts. To ensure (b), they use weighted squared Eucliden distances as a minimization criterion. 70,000 customers are aggregated into 1,400 postal areas for simplifying. No connectivity was considered. They solve a case study.

Drexl and Haase [7] study the solution of sales force deployment which involves the solution of four interrelated subproblems namely: sales force sizing, salesman location, sales territory alignment, and sales resource allocation. They present a MILP model for the problem of maximizing revenue subject to connectivity and profit-related constraints for their subproblems. They propose an approximation method based on solving successively a series of MILPs.

More recently, Kalcsics, Nickel and Schröder [14] present an extensive review of territory design problems. In their work, they identify common features to many territory design problems, introduce a basic framework and present in detail two approaches for solving this model: a classical location-allocation method based on previous work, and a computational geometry-based method. They present results that assess the efficiency of the latter for large-scale practical problems. The model they address does not consider contiguity constraints though. They wrap up by highlighting extensions to the basic model.

4.2 Political Districting

Hess et al. [12] present a location-allocation heuristic for political districting under population equality, compactness, and contiguity considerations. They seek to minimize the sum of squared distances between each unit and its district center. In their empirical work, their heuristic was applied to instances of up to 35 territories and 299 nodes, taking less than a minute of CPU to find feasible solutions.

Garfinkel and Nemhauser [10] present an enumerative algorithm for finding optimal solutions to a political districting problem under contiguity, compactness, and limited population deviation

requirements. In their empirical work, they were able to solve problems with 40 or fewer units.

Hojati [13] addresses a political districting problem in the city of Saskatoon, Canada. He uses a three-stage approach where Lagrangian relaxation is used first to determine district centers, then an assignment problem is solved for allocating population units to districts, and then a sequence of capacitated transportation problems are solved for dealing with splits. His approach, applied to a 42-unit 5-district problem found more compact partitions with less splits than the existing districting plan implemented by the city.

Mehrotra, Johnson, and Nemhauser [16] address the problem of political redistricting from a column generation perspective, and present a heuristic based on branch and price for a case study in the state of California. They used their method to attempt to solve a 51-node problem with 6 districts. Their method compared favorable with respect to a clustering heuristic as was able to deliver solutions within 2% of target of the balancing constraint.

Bozkaya, Erkut, and Laporte [3] address a political districting problem subject to side constraints such as contiguity, population equality, compactness, and socio-economic homogeneity. They develop a tabu search procedure and use it to find solutions to a real-world case in Edmonton, Canada, with 828 basic units in 19 districts. Their procedure integrates several of the criteria into a single-value objective function. Their results indicate that the algorithm produced better maps than the existing one, while improving at the same time some of the other constraints.

Bação, Lobo, and Painho [1] apply a genetic algorithm to a particular political districting problem in Portugal by using several measures as objective functions. The authors claim their method was applied to a 93-district problem in Portugal (they do not mention the total number of units), and show their results for the region of Lisbon (a 7-district subproblem).

4.3 Other Districting Problems

D'Amico et al. [5] present a simulated annealing algorithm to the problem of redistricting or redrawing police command boundaries. They model the problem as a constrained graph-partitioning problem involving partitioning of a police jurisdiction into command districts subject to constraints of contiguity, compactness, convexity, and size. Since the districting plan affects urban emergency services, they also consider quality-of-service constraints. They tested their method in a case study in Buffalo, New York, Police Department, where they were able to significantly reduce the officer workload disparity while maintaining current levels of response time in a 409-node network and 5 districts.

Blais, Lapierre, and Laporte [2] describe a districting study undertaken for a local community health clinic in Montréal. In their problem, a territory had to be partitioned into six districts and five disticting critera had to be met: indivisibility of basic units, respect for borough boundaries, connectivity, visiting personnel mobility, and workload equilibrium. The last two criteria are com-

bined into a single objective function and the problem was solved by a tabu search technique that iteratively moves a basic unit to an adjacent district or swaps two basic units between adjacent districts.

More recently, Vargas-Suárez, Ríos-Mercado, and López [20] address a related commercial TDP with a variable number of territories p, using as an objective a weighted function of the activity deviations from a given goal. No compactness was considered. A basic GRASP was developed and tested in a few instances obtaining relatively good results.

5 Solving the TDP by GRASP

GRASP [8], a fairly well-known metaheuristic that captures good features of both pure greedy algorithms and random construction procedures, has been widely used for successfully solving many combinatorial optimization problems. This type of approach has not been considered before for general territory design problems as far as we know.

A GRASP is an iterative process in which each major iteration consists typically of two phases: construction and post-processing. The construction phase attempts to build a feasible solution S, and the post-processing phase attempts to improve it. When a feasible solution is successfully found in phase one, phase two is typically a local search within suitable neighborhoods with the aim of improving the objective function value. In our particular case, the construction phase does not necessarily terminate with a feasible solution, since the solution found may violate constraints (3) and (4)-(5). Thus, both an adjustment phase, that modifies the current solution so as to satisfy (3), and a post-processing phase, that attempts to improve the solution quality and to reduce the total relative infeasibility with respect to (4)-(5), are developed. Algorithm 1 illustrates a generic GRASP implementation in pseudocode. The algorithm takes as an input an instance of the TDP, the maximum number of GRASP iterations, the restricted candidate list (RCL) quality parameter α, and the number of teritories, and returns a solution S^{best}. Note that in Step 2, q is the number of territories found in partition S after the construction phase is done.

The motivation for GRASP in this particular application stems from the fact that it seems more appealing than current state-of-the-art approaches based on two-stage location-allocation algorithms for handling the connectivity constraints (6). By handling these constraints within a construction heuristic such as GRASP, the connectivity is always kept so it remains to appropriately address the balancing constraints (4)-(5). In the next sections, we describe in detail each of the GRASP basic components and some advanced features such as filtering and reactivity incorporated in the heuristic.

5.1 Construction Phase

In the construction phase, at a given iteration we consider a partial territory and attempt to either allocate an unassigned node to it or to "close" the current territory and "start" a new one. For favoring contiguity, when a new territory is started, the first node is an unassigned one with the smallest degree. When assigning a node that is not the first one in a territory, a greedy function that weighs both a distance-based dispersion measure and the relative violation of the balance constraints (4)-(5) is used. Let V_k be the current territory being built. We denote by $f(V_k) = \max_{i,j \in V_k} d_{ij}$ its corresponding dispersion measure (as dictated by the objective function). Recall that $w_a(V_k) = \sum_{i \in V_k} w_i^a$ is referred to as the size of V_k with respect to activity a, $a \in A$.

For a candidate node v, we define its greedy function as

$$\phi(v) = \lambda F_k(v) + (1 - \lambda) G_k(v), \tag{8}$$

where

$$F_k(v) = \left(\frac{1}{d_{\max}}\right) f(V_k \cup \{v\}) = \left(\frac{1}{d_{\max}}\right) \max\left\{ f(V_k), \max_{j \in V_k} d_{vj} \right\},$$

accounts for the original objective function, and

$$G_k(v) = \sum_{a \in A} g_k^a(v),$$

with $g_k^a(v) = (1/\mu^a) \max\{w^a(V_k \cup \{v\}) - (1+\tau^a)\mu^a, 0\}$, accounts for the sum of relative infeasibilities for the balancing constraints. Here, $d_{\max} = \max_{i,j \in V}\{d_{ij}\}$ is used for normalizing the objective function. Note that $g_k^a(v)$ represents the infeasibility with respect to the upper bound of the balance constraint for activity a, and these two factors are weighted by a parameter λ in function (8). Algorithm 2 shows the pseudocode of the construction procedure.

Line 7 in Algorithm 2 builds the restricted candidate list (RCL) by value α. In line 10, the criteria for closing the current territory is checked. If met, that is, a balance constraint upper bound has been violated, the current territory is "closed" and a new one is "started". Note that in fact, this threshold is adjusted by a parameter $\rho > 0$ which allows for further flexibility in succeeding stages. A value of $\rho < 1$ allows, for instance, to close a territory having a relatively small size. This could allow, however, a violation of constraints (4) when merging territories in the adjustment phase. Note that at this point, we are not concerned with the lower bound balancing constraints (5) because it does not have any impact when adding nodes to a territory, i.e. regardless of which node is chosen, its lower bound violtion is never worse. This situation changes, however, in the local search when all territories have been built.

5.2 Adjustment Phase

Procedure `ConstructGreedyRandomizedSolution()` does not necessarily return a feasible solution. In particular, constraints (3) and (4)-(5) may not be satisfied. To address this issue, a two-step

post-processing phase is applied. First, if (3) is not met, i.e., the number of territories q found in the construction phase is different from p, we either merge territories ($q > p$) or split territories ($q < p$) in procedure `Adjustment()`. The merging operation consists of iteratively considering a territory of smallest size and merging it with its smallest neighboring territory. This reduces the number of connected territories by one at each iteration. This is iteratively repeated until $q = p$. The splitting operation consists of taking a territory of largest size, and splitting it into two connected territories (by recursively applying the same algorithm to the subgraph induced by this territory with $p = 2$). This increases the number of territories by one at each iteration, so the procedure is performed iteratively until $q = p$. Note that the merging operation can be done very efficiently, while the splitting operation is itself another TDP problem. However, the nature of the construction phase makes merging more likely to be applied than splitting. In fact, in the empirical evaluation of the procedure, we have found that the splitting operation is required in less than 0.4% of the cases.

5.3 Local Search

After this adjustment step, a post-processing phase consisting of a local search is performed. Procedure `PostProcessing()` attempts both to recover feasibility of constraints (4)-(5) and to improve the objective function value. In this local search, a merit function that weighs both infeasibility with respect to (4)-(5) and the objective function value is used. In fact, this function is similar to the greedy function used in the construction phase with the exception that now the sum of relative infeasibilities takes into consideration both lower and upper bound violation of the balancing constraints. Specifically, for a given partition $S = \{V_1, \ldots, V_p\}$, its merit function $\psi(S)$ is given by

$$\psi(S) = \lambda F(S) + (1 - \lambda)G(S)$$

where

$$F(S) = \left(\frac{1}{d_{\max}}\right) \max_{k=1,\ldots,p} \left\{ \max_{i,j \in V_k} d_{ij} \right\},$$

and

$$G(S) = \sum_{k=1}^{p} \sum_{a \in A} g^a(V_k),$$

with $g^a(V_k) = (1/\mu^a) \max\{w^a(V_k) - (1+\tau^a)\mu^a, (1-\tau^a)\mu^a - w^a(V_k), 0\}$, being the sum of the relative infeasibilities of the balancing constraints.

We use a neighborhood $N(S)$ made up of all solutions reachable from S by moving a basic unit i from its current territory $t(i)$ to a neighbor district $t(j)$, where j is the corresponding basic unit in territory $t(j)$ adjacent to i, without creating a non-contiguous solution. Such a move is denoted by $move(i, j)$ and is illustrated in Figure 3, where $move(i, j)$ is represented by arc (i, j) (depicted in bold). Note that $move(i, j)$ is allowed only if $V_{t(j)} \cup \{i\}$ is connected (which is always the case if arc (i, j) exists), and $V_{t(i)} \setminus \{i\}$ remains connected. In practice an additional stopping criteria, such

as *limit_moves*, is added to avoid performing the search for a relatively large amount of time. So the procedure stops as soon a local optima is found or the number of moves exceeds *limit_moves*. In our empirical work both a first improving and best improving rules were evaluated.

5.4 Filtering

The most time consuming phase of the GRASP is certainly the local search. To speed up the algorithm we consider the use of a filter to avoid executing the local search in unpromising bad solutions generated by the previous phase. To achieve this, we store the average value $\bar{\beta}$ of the ratio $(\psi(S) - \psi(S'))/\psi(S)$, i.e., the average reduction obtained by the local search phase (S') with respect to the solution found in the previous phase (S). After the first 100 iterations, we make use of this information to decide whether or not each constructed solution is submitted to the local search. The idea is based on the rationale that if some reasonable threshold applied to the cost of the constructed solution leads to a value much higher than the cost of the best solution found so far, it is unlikely that the local search could produce a better solution than the current best. We use $\beta(1 - \bar{\beta})$ as a threshold, where β is an algorithmic parameter that weighs the tightness of this threshold. A high (low) value for β implies that the threshold would be satisfied less (more) often and thus the local search is performed less (more) often. Typical values of β are in the [0,1] range, where $\beta = 0$ corresponds to the extreme case where the threshold is always satisfied so the local search is always performed. This idea is similar to the one proposed by Prais and Ribeiro [18], except that they used a fixed value of $\beta = 0.9$. In our case, we use values of β in the (0.4,0.9) range. Results are reported in Section 6 (experiment C). The pseudocode of `GRASP_filter()` is depicted in Algorithm 4.

5.5 Reactive GRASP

The RCL quality parameter α is basically the only parameter to be calibrated in a practical implementation of a GRASP. Feo and Resende [8] have discussed the effect of the choice of the value of α in terms of solution quality and diversity during the construction phase and how it impacts the outcome of a GRASP. In a Reactive GRASP approach [6, 18] this α is self-adjusted according to the quality of the solutions previously found.

Instead of using a fixed value for the parameter α, which determines what elements are placed in the RCL at each iteration of the construction phase, the procedure randomly selects this value α from a discrete set $A = \{\alpha_1, \ldots, \alpha_m\}$ containing m predetermined acceptable values. Using different values of α at different iterations allows for building different RCLs, possibly leading to the construction of different solutions which would never be built if a single, fixed value of α was used. Let p_i denote the probability associated with the choice of α_i, for $i = 1, \ldots, m$. Initially, $p_i = 1/m, i = 1, \ldots, m$, corresponding to a uniform distribution. Then these probabilities are

periodically updated using information collected during the search. Different strategies for this update can be explored.

At any GRASP iteration, let A_i be the average value of the solutions obtained with $\alpha = \alpha_i$ in the construction phase. The probability distribution is periodically updated every *update_period* iterations (we use *update_period* = 200 in our implementation) as follows. Compute first $q_i = (1/A_i)^\delta$ for $i = 1, \ldots, m$, and then update the new values of the probabilities by normalization of the q_i as $p_i = q_i/(\sum_j q_j)$. Note that the smaller the A_i, the higher the corresponding p_i. Consequently, in the next block of iterations, the values of α that lead to better solutions have higher probabilities and are more frequently used in the construction phase. The exponent δ may be used and explored to differently atenuate the updated values of the probabilities. In our case, we use $\delta = 8$. The pseudocode of the Reactive GRASP is shown in Algorithm 5.

6 Empirical Work

In this section we present experimental results obtained with a C++ implementation of the GRASP for this territory design problem. The procedure was compiled with the Sun C++ compiler workshop 8.0 under the Solaris 9 operating system and run on a SunFire V440. For the experiments, randomly generated problems based on real-world data provided by the industrial partner were generated. Two different data sets are considered, namely DS and DT.

Data set DS: For this set each instance topology was randomly generated as a planar graph in the $[0, 100] \times [0, 100]$ plane. Then, the three node activities were generated from a uniform distribution in the [4,20], [15,400] and [15,100] ranges for number of customers, product demand, and workload, respectively. In addition, four different types of instances according to parameter τ_a are considered. Recall that this parameter sets the allowable deviation from the target of the balancing constraints. We denote these sets as DS30, DS20, DS10, and DS05, corresponding to a τ_a value of 0.30, 0.20, 0.10, and 0.05, respectively. In this sense, DS30 and DS20 are referred to as the loosely constrained sets, and DS10 and DS05 as the tightly constrained sets.

Data set DT: For this set we consider randomly generated maps on the $[0, 500] \times [0, 500]$ plane. Then, each of the three node activities were generated from a non-uniform symmetric distribution in the following way. Each node in our problem represents the aggregated information of 68 blocks in the original graph. So in our problem, each node is the sum of 68 independent non-uniform symmetrically distributed random variables. The number of customers, in each block, is generated as Pr(Number of customers $= r_1$) is 0.15 for $r_1 = 0, 3$ and 0.35 for $r_1 = 1, 2$. The product demand in each block with at least one customer is generated as Pr(Demand $= r_2$) is 0.01 for $r_2 = 1, 12$, 0.03 for $r_2 = 2, 11$, 0.06 for $r_2 = 3, 10$, 0.10 for $r_2 = 4, 9$, 0.12 for

$r_2 = 5, 8$, and 0.18 for $r_2 = 6, 7$. The workload is generated the same way. Naturally, for a given block $r_1 = 0$ implies $r_2 = r_3 = 0$. Again, four different type of instances, named DT30, DT20, DT10, and DT05 are generated corresponding to a τ_a value of 0.30, 0.20, 0.10, and 0.05, respectively.

There are two sources for a randomly generated instance. One is the generation of the map, that is, the graph topology, and the other is the generation of the node attributes. There are two reasons for generating our data set the way we did. As far as the topology is concerned, due to proprietary reasons, the firm has not allowed us yet to use its city map. On the other hand, this allows us to test our method over a more general scenario, and not making it too problem specific. In addition, given that our basic geographic units are city blocks, and one can argue that blocks are somewhat uniformly distributed over an urban setting, the generation of city blocks by a uniform distribution is reasonable. Regarding the node attribute generation, using a uniform distribution generates instances with larger variance, so again, this sets the goal of testing the method over a more general situation. Data set DT represents more closely that of the specific real-world application. Data set DS comes from the data set used by [20] in a similar problem, and, as stated before, is intended to prove the robustness of the proposed method. For each of these set types, 20 different instances of size $n = 500$ and $p = 10$ were generated. The choice of size is justified because this resembles the one currently used in the specific real-world application. Throughout the evaluation, the GRASP is run with *limit_iterations* = 1000 unless otherwise stated.

Experiment A

In this part of the work, we study the sensitivity of the algorithm with respect to the choice of the parameter ρ, which is used within the GRASP construction phase, for deciding when to close (stop allocating new nodes to it) a currently active territory and start a new one (Section 5.1). To this effect, we run the GRASP with no local search phase, and we measure the quality of the weighted objective function (ψ), the distance-based measure (F), and the degree of infeasibility (G). Results over twenty 500-node instances both for loose and tight balancing constraints are reported in Table 1 and Table 2, respectively. In this study, the GRASP parameter α is set to 0.3.

The results in Table 1 show that for loosely constrained instances the value $\rho = 0.8$ gave the best results for all three evaluated measures: weighted objective function, distance-based measure, and the degree of infeasibility. In particular, all the solutions obtained with the value $\rho = 0.8$ were feasible for all the instances in DS30 and DT30, and all but one instance in DS20 and DT20 (in which the deviation from feasibility was really small). For instances in DS30 and DT30, the second best choice was $\rho = 0.6$, whereas for instances in DS20 and DT20 the second best choice was $\rho = 1$. These results indicate that for very loosely constrained instances, small values of ρ give better results, whereas when instances become somewhat more constrained, higher values of

$\rho = 0.6$ yield better results. This tendency is, in fact, confirmed by the results in Table 2 where we can see that for tightly constrained instances the value $\rho = 1$ gave the best results for DS05 and DT05. It is worth noting that even for really tightly constrained instances, like the ones in set DS05, the constructive phase generates solutions where, on the average, the sum of relative infeasibility is very small, indicating that most of these solutions are feasible and when not their deviation from feasibility is small. As for the CPU times, both with loosely constrained instances and for tightly constrained instances, higher values of ρ are more time consuming. This is certainly due to the fact that for each territory that is being constructed higher values of ρ require more iterations of the constructive phase until the threshold value to close that territory is reached.

Since this experiment shows that feasibility is not an issue in loosely constrained instances, in all remaining we have used the more challenging tightly constrained instances in DS10, DT10, DS05, and DT05, and the value of the parameter ρ is fixed to one for DS10, DS05, and DT05, and fixed to 0.8 for DT10.

Experiment B

To investigate the behavior of the local search procedure, we implemented two versions, each with a different strategy for exploring the neighborhood. The first version corresponds to a first found scheme. That is, the procedure examines the neighborhood of a current solution in a random fashion, one element at a time, and then makes a move to the first improving neighbor. The second version is a steepest-descent approach, which examines the entire neighborhood and moves to the best neighbor. These two versions are referred to as FF (first found) and BN (best neighbor). We set the GRASP parameters *limit_iterations* $= 500$, $\alpha = 0.3$, and *limit_moves* $= 100$. Computational results on sets DS10, DS05, DT10, and DT05 are presented in Table 3.

As can be seen, the results in Table 3 indicate that the two versions of the local search indeed provide with considerably improved solutions. This is true for all three evaluated measures: weighted objective function, distance-based measure, and the degree of infeasibility. The reductions in the sum of relative infeasibilities are specially relevant, since the improvement obtained with the local search is in all cases above 74%. In particular, all the solutions found with the FF strategy for the DS10 and DT10 instances were feasible, and the sum of infeasibilities was 0.01 for the most infeasible solution generated for any of for the DS05 instances, and less than 3.67×10^{-4} for the worst DT05 instance. Moreover, the FF strategy gave also the best results for the two other measures, and, at the same time, it was also less time consuming than BN. Hence, given that this strategy gave the best results in the remaining experiments we have used the FF strategy for the local search phase.

Experiment C

As seen in the previous experiment, local search can be quite time consuming and, on the average, consumes more than one third of the total time of the algorithm. As stated in Section 5.4, one way to speed up the procedure is to use a filter that attempts to avoid applying the local search phase to unpromising bad solutions generated by the construction phase. To do this, at every iteration we store the average value $\bar{\beta}$, $(0 \leq \bar{\beta} \leq 1)$, of the ratio $(\psi(S) - \psi(S'))/\psi(S)$. The value $\bar{\beta}$ is the average reduction obtained by the local search phase with respect to the solution obtained in the construction phase. After the first 100 iterations, we make use of this information to decide whether or not each constructed solution is submitted to the local search phase. The idea is based on the premise that if some reasonable threshold applied to the cost of the constructed solution leads to a value significantly higher than the cost of the best solution found so far, it is unlikely that the local search could yield a better solution than the current best. To investigate this, we use as threshold the value $\beta \times (1 - \bar{\beta})$, with $0 \leq \beta \leq 1$, and we analyze different values of β: 0.0 (which means no filtering), 0.6, and 0.8.

The fixed parameters for this experiment are *limit_iterations* = 1000, *limit_moves* = 200, and $\alpha = 0.3$. Results for DS10 and DS05 are shown in Table 4 and Table 5, respectively. In these tables different columns correspond to different values of β, and the value of β is given in brackets at the heading of the column.

The results depicted in both tables clearly indicate that the use of a filter to limit the number of local search phases that are applied, is really worthwhile, since results that are very similar in terms of quality can be obtained with a much smaller work load. In particular, the results in Table 4 show that for the instances in DS10, when $\beta = 0.8$, i.e., on the average local search is applied on about 20% of the iterations, the obtained solutions are similar in terms of feasibility and only slightly worse in terms of the objective function, with a CPU time reduction of 30% of the total time, or 52%, if we measure the time reduction with respect to the one required by the local search phase. The results for DS05 given in Table 5 lead to similar conclusions. It is remarkable that for so tightly constrained instances, an average reduction of the 60% on the number of times that the local search is applied ($\beta = 0.6$) gives results that are nearly the same as when local search is applied at all iterations. Results for instances DT10 and DT05 are very similar.

Experiment D

This experiment is focused on the appropriate choice of the quality parameter α in the construction phase. Recall that this is the parameter that regulates the size of the Restricted Candidate List. The purpose of this experiment is twofold: on the one hand, to evaluate the algorithmic performance as a function of the value of α; and, on the other hand, to analyze the effectiveness of a reactive version of the algorithm when the value of α instead of being fixed as a parameter, is self-adjusted,

based on the history of the search.

For evaluating the performance of the algorithm in terms of α, we set *limit_iterations* = 1000, *limit_moves* = 200, and $\beta = 0$, and we compare the results obtained with the different values of $\alpha \in \{0.1, 0.2, 0.3, 0.4, 0.5\}$, on data sets DS10, DS05, DT10, and DT05. A summary of the results is displayed in Table 6. For each solution, its average relative deviation is computed with respect to the best known feasible solution as $100 \times (f_{\text{heuristic}} - f_{\text{best}})/f_{\text{best}})$. Thus, the results for DS05 include only those for which a feasible solution was found (18 out of 20 instances). As can be seen, for the instances in DS10, the best results were obtained with $\alpha = 0.2$, both in terms of quality and feasibility, whereas for instances in DS05, $\alpha = 0.4$ gave the best results, in terms of feasibility and average quality, even if the worst relative gap is the largest one. Possibly, $\alpha = 0.3$ is a better compromise in this sense. In any case, the obtained results indicate that as the instances become more tightly constrained and, thus, more difficult to solve in terms of feasibility, a higher value of α can be preferred. Similar behavier is observed for data sets DT10 and DT05. For instance, the best results are observed for lower values of $\alpha = 0.1$, whereas the best results for DT05 are observed when $\alpha = 0.3$.

To conclude this subsection, in Tables 7 to 10 we summarize the numerical results obtained with the Reactive GRASP (R-GRASP) for the same set of instances DS10, DS05, DT10, and DT05, respectively. In any of these tables, for a given instance and a fixed value of α, the corresponding entry depicts the relative gap of the best solution obtained for this instance with this value of α with respect to the best known feasible solution for that instance (obtained with any of the tested α's). The column under the heading RG gives the relative gaps of the best feasible solution obtained with R-GRASP (again with respect to the best known feasible solution), whereas the value $\bar{\alpha}$ gives the value of α that gave the best solution in R-GRASP. No significant differences in the running times between the GRASP and R-GRASP could be observed, so time is not displayed. The results of both tables indicate that R-GRASP is a very suitable option. Note that even if, on the average, R-GRASP is not the best option neither for DS10 nor for DS05, it is really a good compromise and it has the clear advantage that no tuning is needed to find out the better option for the parameter α. For instance, in DS10, it is observed that lower values of α seem to deliver better solutions; however, this statement does not hold for DS05. In fact, these lower values of α failed to find feasible solutions in some of the DS05 instances. In this regard, the Reactive GRASP exhibits a more robust behavior. Same behavior is observed for DT10 and DT05. In DT10, a best value of $\alpha = .01$ turns out to be the worst in DT05, and viceversa. In both cases, the Reactive GRASP results are more robust. In fact, the Reactive GRASP method find more best solutions in DT05.

Experiment E

In this part of the work, we perform a comparison of the proposed approach with current pratice. To this end we set up the experiment as follows. We apply a method that reflects the current standard for building solutions. The procedure works in a constructive fashion building one territory at a time. In this phase, no attention is paid to the balancing constraints, but only to the compactness measure. The method builds territories where each territory has about the same number of nodes (given by n/p). Although this method is not too bad on delivering feasible solutions to the less restricted problems (such as DS030), it is certainly extremely limited when addressing the more restricted problems. Thus, in an attempt to improve these solutions, we also apply the local search phase to each instance.

Table 11 shows a comparison, in terms of the feasibility issue, of the solutions found by the firm's method (Firm) and those found by adding the local search (F+LS). As can seen, the local search has a significant improving effect by considerably reducing the average infeasibility, and finding about 50% of feasible solutions which contrasts with the 0% figure by the current practice.

Next, we compare the firms plus with local search (F+LS) with our Reactive GRASP (R-G) implementation. Figures 6 to 9 show a comparison between the two in data sets DS10, DS05, DT10, and DT05, respectively, in terms of the value of the dispersion function. As can be seen, the proposed approach outperforms the current practice by a significant margin. Note that there are only two cases were a better solution was found by Firm+LS; however, these two rendered infeasible solutions with respect to the balancing constraints.

Finally, Table 12 shows a comparison between these two methods in terms of their relative infeasibility. Once again, we can see the proposed apporach behaves in a significantly more robust way as it was able to deliver feasible instances to all 80 DS instances tested, but one.

7 Conclusions

In this paper we have presented a GRASP approach to a sales territory design problem with multiple node balance requirements. The problem that is motivated by a real-world application in the beverage industry, includes several planning criteria such as compactness, balancing among territories, contiguity, and connectivity. Each of the GRASP components was fully evaluated over a range of instances randomly generated according to real-world scenarios. The reactive GRASP with filtering was very successfull as it was able to obtain feasible solutions to 159 out of 160 instances tested on the eight data sets considered (given in data set DS05).

In particular, the local search scheme produced a significant improvement over the quality of the instances found in the construction phase, producing average improvements of 75-90% with respect to the construction phase, and doing an excellent job on recovering feasibility with respect to the

balancing constraints. The reactive component of GRASP was found very robust. and proved a very valuable asset over the costly calibration of the quality parameter α.

When compared to current practice, it is clear from the empirical evidence that the proposed approach finds solutions of significantly better quality, dominating entirely to the best solutions found by the firm's method. In addition, the method showed a very robust behavior in finding feasible solutions in a relatively small computational effort. The behavior of the method when used to solve the instances of data set DT was even better. Recall that these instances follow a nonuniform symmetric distribution.

Several issues remain to be investigated. While the dominance of the proposed approach is clear, the development of a lower bound would help assessing the overall quality of the solutions. The MILP formulation has an exponential number of connectivity constraints, so special care must be taken to appropriately address this issue. A problem with 500 nodes has 250,000 binary variables. To this end, it seems possible trying to exploit the underlying structure of the embedded multicapacitated vertex p-center problem. This study is currently being pursued by the authors.

So far we have developed both construction and local search procedures, which in turn are at the core of more sophisticated metaheuristics such as tabu search and scatter search, that could be worthwhile the effort.

Acknowledgments: The research of the first author was supported by grant SAB2004–0092 of the Spanish State Secretary for Universities and Research under its Visiting Scholar Program and by grant 48499–Y of the Mexican National Council for Science and Technology under its Basic Research Program. The second author is supported by grant MTM2006–14961–C05–01 of the Inter-Ministerial Spanish Commission of Science and Technology, respectively. We are also very grateful to two anonymous referees, whose comments and criticism helped improved the quality of this work.

References

[1] F. Bação, V. Lobo, and M. Painho. Applying genetic algorithms to zone design. *Soft Computing*, 9(5):341–348, 2005.

[2] M. Blais, S. D. Lapierre, and G. Laporte. Solving a home-care districting problem in an urban setting. *Journal of the Operational Research Society*, 54(11):1141–1147, 2003.

[3] B. Bozkaya, E. Erkut, and G. Laporte. A tabu search heuristic and adaptive memory procedure for political districting. *European Journal of Operational Research*, 144(1):12–26, 2003.

[4] F. Caro, T. Shirabe, M. Guignard, and A. Weintraub. School redistricting: Embedding GIS tools with integer programming. *Journal of the Operational Research Society*, 55(8):836–849, 2004.

[5] S. J. D'Amico, S.-J. Wang, R. Batta, and C. M. Rump. A simulated annealing approach to police district design. *Computers & Operations Research*, 29(6):667–684, 2002.

[6] H. Delmaire, J. A. Díaz, E. Fernández, and M. Ortega. Reactive GRASP and tabu search based heuristcs for the single source capacitated plant location problem. *INFOR: Information Systems and Operational Research*, 37(3):194–225, 1999.

[7] A. Drexl and K. Haase. Fast approximation methods for sales force deployment. *Management Science*, 45(10):1307–1323, 1999.

[8] T. A. Feo and M. G. C. Resende. Greedy randomized adaptive search procedures. *Journal of Global Optimization*, 6(2):109–133, 1995.

[9] B. Fleischmann and J. N. Paraschis. Solving a large scale districting problem: A case report. *Computers & Operations Research*, 15(6):521–533, 1988.

[10] R. S. Garfinkel and G. L. Nemhauser. Solving optimal political districting by implicit enumeration techniques. *Management Science*, 16(8):B495–B508, 1970.

[11] S. W. Hess and S. A. Samuels. Experiences with a sales districting model: Criteria and implementation. *Management Science*, 18(4):998–1006, 1971.

[12] S. W. Hess, J. B. Weaver, H. J. Siegfeldt, J. N. Whelan, and P. A. Zitlau. Nonpartisan political redistring by computer. *Operations Research*, 13(6):998–1006, 1965.

[13] M. Hojati. Optimal political districting. *Computers & Operations Research*, 23(12):1147–1161, 1996.

[14] J. Kalcsics, S. Nickel, and M. Schröder. Toward a unified territorial design approach: Applications, algorithms, and GIS integration. *Top*, 13(1):1–74, 2005.

[15] O. Kariv and S. L. Hakimi. An algorithmic approach to network location problems. I: the p-centers. *SIAM Journal on Applied Mathematics*, 37(3):513–538, 1979.

[16] A. Mehrotra, E. L. Johnson, and G. L. Nemhauser. An optimization based heuristic for political districting. *Management Science*, 44(8):1100–1113, 1998.

[17] L. Muyldermans, D. Cattryse, D. Van Oudheusden, and T. Lotan. Districting for salt spreading operations. *European Journal of Operational Research*, 139(3):521–532, 2002.

[18] M. Prais and C. C. Ribeiro. Reactive GRASP: An application to a matrix decomposition problem in TDMA traffic assignment. *Journal on Computing*, 12(3):164–176, 2000.

[19] M. Segal and D. B. Weinberger. Turfing. *Operations Research*, 25(3):367–386, 1977.

[20] L. Vargas-Suárez, R. Z. Ríos-Mercado, and F. López. Usando GRASP para resolver un problema de definición de territorios de atención comercial. In M. G. Arenas, F. Herrera, M. Lozano, J. J. Merelo, G. Romero, and A. M. Sánchez, editors, *Proceedings of the IV Spanish Conference on Metaheuristics, Evolutionary and Bioinspired Algorithms (MAEB)*, pages 609–617, Granada, Spain, September 2005. In Spanish.

[21] A. A. Zoltners and P. Sinha. Toward a unified territory alignment: A review and model. *Management Science*, 29(11):1237–1256, 1983.

Table 1: Evaluation of GRASP parameter ρ on loose instances (DS30, DS20, DT30, DT20)

500-node instances		DS30			DS20		
Statistic		$\rho = 1.0$	$\rho = 0.8$	$\rho = 0.6$	$\rho = 1.0$	$\rho = 0.8$	$\rho = 0.6$
Weighted objective	Best	0.39	0.05	0.16	0.36	0.06	0.73
$\psi(S)$	Average	1.14	0.05	0.33	0.88	0.06	1.00
	Worst	1.46	0.06	0.40	1.15	0.06	1.37
Objective (dispersion)	Best	25.91	23.88	30.11	23.63	24.46	27.80
$F(S)$	Average	29.41	25.07	32.81	31.33	26.44	33.07
	Worst	36.67	26.94	36.84	43.47	29.01	38.43
Sum of relative infeasibilities	Best	0.46	0.00	0.12	0.44	0.00	0.94
$G(S)$	Average	1.53	0.00	0.36	1.16	0.00	1.32
	Worst	2.00	0.00	0.47	1.53	0.01	1.85
Time (sec)	Phase 1	119.65	94.30	74.45	104.15	83.50	66.30
	Phase 2	78.60	63.60	72.30	70.00	61.55	69.10
	Total	198.25	157.90	146.75	174.15	145.05	135.40

500-node instances		DT30			DT20		
Statistic		$\rho = 1.0$	$\rho = 0.8$	$\rho = 0.6$	$\rho = 1.0$	$\rho = 0.8$	$\rho = 0\ .6$
Weighted objective	Best	1.11	0.05	0.06	1.07	0.05	0.58
$\psi(S)$	Average	1.49	0.06	0.16	1.40	0.06	0.90
	Worst	1.76	0.07	0.44	1.76	0.06	1.24
Objective (dispersion)	Best	125.81	115.06	137.47	124.44	120.74	132.35
$F(S)$	Average	153.20	128.16	166.28	153.74	126.08	159.73
	Worst	222.31	145.22	196.62	186.45	133.50	192.75
Sum of relative infeasibilities	Best	1.49	0.00	0.00	1.44	0.00	0.73
$G(S)$	Average	2.03	0.00	0.12	1.90	$< 10^{-4}$	1.19
	Worst	2.37	0.00	0.50	2.42	$< 10^{-3}$	1.65
Time (sec)	Phase 1	125.45	97.20	76.10	126.55	98.30	77.20
	Phase 2	84.40	67.55	72.30	85.30	68.45	73.20
	Total	209.85	164.75	148.40	211.85	166.75	150.40

Table 2: Evaluation of GRASP parameter ρ on tight instances (DS10, DS05, DT10, DT05)

500-node instances		DS10		DS05		DT10		DT05	
Statistic		$\rho = 1.0$	$\rho = 0.8$	$\rho = 1.0$	$\rho = 0.8$	$\rho = 1.0$	$\rho = 0.8$	$\rho = 1.0$	$\rho = 0.8$
$\psi(S)$	Best	0.17	0.25	0.16	1.22	0.64	0.08	0.34	0.98
	Average	0.37	0.70	0.22	1.93	0.91	0.24	0.52	1.31
	Worst	0.52	1.00	0.34	2.35	1.04	0.40	0.70	1.71
$F(S)$	Best	25.31	25.20	28.53	28.00	135.61	136.86	129.13	130.58
	Average	30.64	32.44	33.11	34.37	157.65	160.14	157.34	157.86
	Worst	36.10	44.13	39.56	52.75	218.13	193.70	199.08	189.20
$G(S)$	Best	0.16	0.25	0.13	1.62	0.82	0.01	0.38	1.31
	Average	0.44	0.89	0.21	2.65	1.20	0.24	0.65	1.78
	Worst	0.65	1.31	0.37	3.26	1.39	0.47	0.88	2.33
Time (sec)	Phase 1	95.55	78.05	91.10	75.20	96.65	79.15	91.10	75.20
	Phase 2	64.95	65.85	62.20	67.65	65.85	66.75	62.20	67.65
	Total	160.50	143.90	153.30	142.85	162.50	145.90	153.30	142.85

Table 3: Evaluation of local search on tight instances

500-node instances		DS10		DS05		DT10		DT05	
Statistic		FF	BN	FF	BN	FF	BN	FF	BN
$\psi(S)$	Best	0.05	0.06	0.06	0.07	0.05	0.06	0.05	0.06
	Average	0.05	0.06	0.06	0.09	0.06	0.06	0.06	0.07
	Worst	0.06	0.07	0.09	0.14	0.06	0.06	0.06	0.07
$F(S)$	Best	23.42	25.51	23.83	24.63	117.23	125.19	118.91	132.15
	Average	24.61	28.22	27.75	30.66	125.45	133.74	125.60	143.60
	Worst	25.71	41.58	35.54	35.83	133.09	141.64	131.82	165.21
$G(S)$	Best	0.00	0.00	0.00	0.01	0.00	0.00	0.00	0.00
	Average	0.00	0.00	0.00	0.03	0.00	$< 10^{-3}$	$< 10^{-4}$	$< 10^{-2}$
	Worst	0.00	0.01	0.01	0.11	0.00	$< 10^{-2}$	$< 10^{-3}$	$< 10^{-2}$
Rel. LS improvement	Average	92.96	86.35	88.53	77.37	94.15	89.60	88.53	77.60
Time (sec)	Phase 1	48.15	48.65	46.30	46.65	44.45	43.35	50.45	50.60
	Phase 2	32.00	32.20	30.85	30.90	35.95	34.95	34.15	34.00
	Phase 3	46.55	52.45	49.50	53.55	47.00	62.55	48.30	55.55
	Total	124.70	133.30	126.65	131.10	127.40	150.85	132.90	140.15

Table 4: Evaluation of filtering on instances DS10

500-node instances		Filtering G_F(β)		
Statistic		GRASP	G_F(0.6)	G_F(0.8)
Objective (dispersion)	Best	23.60	23.72	24.04
$F(S)$	Average	24.53	24.58	24.97
	Worst	25.62	25.33	26.46
Sum of relative infeasibilities	Best	0.00	0.00	0.00
$G(S)$	Average	0.00	0.00	0.00
	Worst	0.00	0.00	0.00
Number of infeasible solutions		0	0	0
Local search	Min	1000	270	146
(number of times invoked)	Average	1000	515	244
	Max	1000	675	289
Relative LS improvement	Average	95.37	94.20	93.34
Time (sec)	Phase 1	95.25	94.60	97.00
	Phase 2	64.75	64.85	64.85
	Phase 3	100.15	41.15	19.60
	Total	260.15	200.60	181.45

Table 5: Evaluation of filtering on instances DS05

500-node instances		Filtering G_F(β)			
Statistic		GRASP	G_F(0.4)	G_F(0.6)	G_F(0.8)
Objective (dispersion)	Best	24.82	23.56	24.09	23.83
$F(S)$	Average	26.59	26.66	27.20	26.85
	Worst	31.20	31.40	34.99	30.91
Sum of relative infeasibilities	Best	0.00	0.00	0.00	0.00
$G(S)$	Average	0.00	0.00	0.00	0.01
	Worst	0.01	0.02	0.01	0.04
Number of infeasible solutions		2	2	2	8
Local search	Min	1000	392	219	153
(number of times invoked)	Average	1000	641	272	197
	Max	1000	904	329	224
Relative LS improvement	Average	93.02	91.17	88.03	85.00
Time (sec)	Phase 1	90.80	90.80	90.80	90.80
	Phase 2	62.85	62.85	62.85	62.85
	Phase 3	113.75	64.10	24.20	17.90
	Total	267.40	217.75	177.85	171.55

Table 6: Evaluation of quality parameter α.

500-node instances	DS10				
Statistic	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$
Average relative deviation from best	2.19	1.49	2.58	2.72	3.42
Worst relative deviation from best	6.92	5.31	5.81	5.79	10.44
Number of infeasible solutions	0	0	0	0	0
Number of best solutions	7	6	5	3	2
500-node instances	DS05				
Statistic	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$
Average relative deviation from best (*)	3.52	3.89	3.46	3.03	4.89
Worst relative deviation from best (*)	12.06	9.18	9.15	13.40	9.87
Number of infeasible solutions	2	2	1	0	0
Number of best solutions	5	3	4	7	2
(*) Excludes instances d500-03 and d500-06					
500-node instances	DT10				
Statistic	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$
Average relative deviation from best	2.22	2.35	2.90	2.77	2.36
Worst relative deviation from best	8.13	6.33	7.28	8.48	7.09
Number of infeasible solutions	0	0	0	0	0
Number of best solutions	7	4	2	2	6
500-node instances	DT05				
Statistic	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$
Average relative deviation from best	2.53	1.75	1.60	2.51	2.24
Worst relative deviation from best	8.78	4.99	5.87	6.37	5.22
Number of infeasible solutions	0	0	0	0	0
Number of best solutions	5	6	7	2	2

Table 7: Reactive GRASP comparison on DS10.

	GRASP					R-GRASP	
Instance	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	RG	$\bar{\alpha}$
d500-01	6.30	1.52	0.99	6.17	7.94	0.00	0.2
d500-02	0.00	2.41	2.73	0.29	1.51	4.93	0.1
d500-03	4.96	0.00	2.41	2.01	1.25	1.25	0.4
d500-04	0.00	2.01	3.87	3.25	7.06	4.55	0.1
d500-05	2.84	0.72	2.86	5.79	0.00	2.84	0.3
d500-06	3.34	0.31	1.49	2.91	4.67	0.00	0.1
d500-07	0.00	0.00	0.00	3.24	1.39	1.24	0.1
d500-08	3.24	3.19	3.49	2.05	2.93	0.00	0.2
d500-09	0.00	3.67	4.80	4.66	6.21	2.62	0.1
d500-10	0.00	1.50	0.00	5.11	4.01	0.00	0.2
d500-11	1.24	0.00	3.00	5.41	4.58	3.73	0.2
d500-12	5.51	3.78	4.60	0.00	10.44	7.78	0.3
d500-13	2.96	2.96	0.00	3.84	3.60	3.84	0.2
d500-14	6.92	0.00	5.55	2.12	4.01	3.97	0.5
d500-15	5.31	5.31	0.00	1.16	1.20	0.60	0.3
d500-16	2.27	0.91	4.97	2.07	0.00	1.25	0.4
d500-17	5.95	4.54	6.69	4.21	6.22	0.00	0.4
d500-18	0.00	1.83	1.83	0.18	1.09	1.50	0.1
d500-19	0.00	2.76	5.81	2.76	7.24	5.40	0.3
d500-20	0.71	0.00	4.29	4.76	0.75	0.24	0.3
Average	2.58	1.87	2.97	3.10	3.81	2.29	
Worst	6.92	5.31	6.69	6.17	10.44	7.78	
No. best	7	5	4	1	2	5	

Table 8: Reactive GRASP comparison on DS05.

	GRASP					R-GRASP	
Instance	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	RG	$\bar{\alpha}$
d500-01	6.20	3.39	3.53	5.59	6.98	0.00	0.1
d500-02	8.23	0.00	6.12	10.55	8.17	5.64	0.4
d500-03	F	F	F	33.10	0.00	F	0.5
d500-04	0.00	4.80	2.74	9.02	4.81	4.15	0.3
d500-05	2.37	1.00	3.93	0.73	3.07	0.00	0.3
d500-06	F	F	0.00	9.56	14.28	7.93	0.5
d500-07	12.06	4.90	0.00	5.26	6.34	1.94	0.2
d500-08	0.14	1.42	3.49	3.01	2.51	0.00	0.1
d500-09	4.83	2.02	9.15	0.00	7.44	7.99	0.2
d500-10	5.56	3.42	2.85	0.00	3.43	10.68	0.1
d500-11	0.00	3.13	2.97	5.11	4.72	4.61	0.1
d500-12	3.45	7.77	6.00	0.00	8.05	0.19	0.3
d500-13	6.52	2.99	0.00	2.43	4.98	2.93	0.3
d500-14	0.04	3.97	1.57	1.26	0.78	0.00	0.2
d500-15	8.01	9.18	0.00	1.45	0.00	6.14	0.4
d500-16	0.00	6.52	6.60	13.40	8.32	6.52	0.4
d500-17	1.54	6.73	3.85	0.00	5.77	1.61	0.2
d500-18	5.33	8.44	4.68	0.00	9.87	2.63	0.3
d500-19	3.00	4.73	5.21	0.00	2.75	10.69	0.5
d500-20	0.49	0.00	3.93	1.18	4.46	2.86	0.1
Average (*)	3.77	4.13	3.70	3.28	5.14	3.81	
Worst (*)	12.06	9.18	9.15	13.40	9.87	10.69	
No. best	4	3	3	7	3	4	

F: Failed to find a feasible solution

(*) Excludes instances d500-03 and d500-06

Table 9: Reactive GRASP comparison on DT10.

	GRASP					R-GRASP	
Instance	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	RG	$\bar{\alpha}$
dt500-01	3.60	3.60	0.00	2.55	2.63	2.93	0.4
dt500-02	4.88	11.26	8.87	9.68	8.13	0.00	0.2
dt500-03	8.13	3.92	7.28	8.48	0.00	6.06	0.3
dt500-04	1.28	0.00	1.28	0.00	0.99	1.28	0.4
dt500-05	7.00	0.57	0.83	0.00	2.75	4.88	0.5
dt500-06	4.02	1.42	5.39	0.71	0.00	5.78	0.4
dt500-07	3.54	3.91	4.76	4.07	0.00	0.32	0.3
dt500-08	0.00	4.49	4.88	5.83	3.57	1.21	0.2
dt500-09	0.00	1.15	0.18	3.10	1.38	4.07	0.1
dt500-10	0.00	4.54	4.33	2.66	6.26	6.23	0.5
dt500-11	0.22	3.55	2.63	2.10	0.00	0.23	0.5
dt500-12	7.09	0.00	6.27	1.69	7.09	8.97	0.5
dt500-13	5.57	4.57	3.82	6.06	5.35	0.00	0.3
dt500-14	0.00	0.28	3.14	2.34	0.92	3.14	0.4
dt500-15	3.29	2.75	4.21	4.62	8.47	0.00	0.4
dt500-16	0.00	6.33	4.25	6.07	7.01	8.62	0.1
dt500-17	0.84	0.00	0.04	0.37	0.16	1.83	0.2
dt500-18	0.00	1.77	3.33	2.42	4.23	1.87	0.2
dt500-19	2.86	0.94	2.14	3.74	0.00	3.05	0.2
dt500-20	3.65	3.65	2.03	0.78	0.00	2.77	0.1
Average	2.80	2.93	3.48	3.46	2.95	3.16	
Worst	8.13	11.26	8.87	9.68	8.47	8.97	
No. best	6	3	1	2	6	3	

Table 10: Reactive GRASP comparison on DT05.

	GRASP					R-GRASP	
Instance	$\alpha = 0.1$	$\alpha = 0.2$	$\alpha = 0.3$	$\alpha = 0.4$	$\alpha = 0.5$	RG	$\bar{\alpha}$
dt500-01	0.00	0.82	2.19	2.70	2.50	0.26	0.3
dt500-02	0.30	1.85	0.69	4.37	0.76	0.00	0.5
dt500-03	2.57	3.90	1.61	2.35	4.08	0.00	0.4
dt500-04	6.13	4.71	4.35	0.00	0.90	3.10	0.1
dt500-05	2.48	0.62	0.62	0.62	3.01	0.00	0.1
dt500-06	0.96	0.00	0.92	0.09	3.64	0.92	0.4
dt500-07	8.24	4.70	0.00	5.43	5.22	5.22	0.5
dt500-08	0.95	0.66	0.00	2.62	2.04	1.42	0.2
dt500-09	4.82	4.10	1.28	2.59	5.76	0.00	0.2
dt500-10	2.39	0.00	3.04	1.40	2.77	2.12	0.3
dt500-11	0.00	1.27	3.44	6.37	2.66	5.09	0.5
dt500-12	3.12	0.48	3.73	4.38	4.82	0.00	0.1
dt500-13	4.75	0.42	2.56	4.81	1.25	0.00	0.2
dt500-14	0.00	1.38	0.14	3.42	1.71	2.68	0.5
dt500-15	8.78	4.99	0.00	1.20	2.88	4.70	0.5
dt500-16	0.07	0.00	1.04	3.50	1.04	3.67	0.4
dt500-17	2.20	2.54	0.00	2.56	3.15	2.60	0.2
dt500-18	5.69	2.24	5.87	2.75	0.00	2.85	0.3
dt500-19	2.00	2.97	2.25	2.73	0.00	0.23	0.4
dt500-20	0.00	2.18	3.08	1.04	1.51	3.46	0.2
Average	2.77	1.99	1.84	2.75	2.48	1.92	
Worst	8.78	4.99	5.87	6.37	5.76	5.22	
No. best	4	3	4	1	2	6	

Table 11: Feasibility effect of local search on firm's solutions.

500-node instances		DS10		DS05		DT10		DT05	
Statistic		Firm	F+LS	Firm	F+LS	Firm	F+LS	Firm	F+LS
Sum of relative	Best	0.076	0.000	0.905	0.000	0.000	0.000	0.189	0.000
infeasibilities	Average	1.418	0.001	2.084	0.028	1.375	0.000	2.109	0.001
	Worst	2.810	0.021	3.420	0.131	2.730	0.000	4.130	0.010
Infeasible solutions		20	5	20	18	18	0	20	5

Table 12: Feasibility comparison.

500-node instances		DS10		DS05		DT10		DT05	
Statistic		F+LS	R-G	F+LS	R-G	F+LS	R-G	F+LS	R-G
Sum of relative	Best	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
infeasibilities	Average	0.001	0.000	0.028	0.001	0.000	0.000	0.001	0.000
	Worst	0.021	0.000	0.131	0.010	0.000	0.000	0.010	0.000
Infeasible solutions		5	0	18	1	0	0	5	0

function `GRASP` (*limit_iterations*, α, ρ, p)
Input: *limit_iterations* := GRASP iteration limit; α := GRASP RCL quality parameter; ρ := territory closure parameter; p := Number of territories.
Output: A feasible assigment S^{best}.

0 $S^{\text{best}} \leftarrow \emptyset$;
1 **for** ($l = 1, \ldots,$ *limit_iterations*) **do**
2 $S \leftarrow$ `ConstructGreedyRandomizedSolution(`α`, `ρ`)`; $q \leftarrow |S|$;
3 **if** ($q \neq p$) **then** $S \leftarrow$ `Adjustment(`S`)`;
4 $S \leftarrow$ `PostProcessing(`S`)`;
5 **if** (S better than S^{best}) **then** $S^{\text{best}} \leftarrow S$;
6 **endfor**;
7 **return** S^{best};
end `GRASP`

Algorithm 1: A GRASP pseudocode for TDP.

function `ConstructGreedyRandomized` (α, ρ)

Input: α := GRASP RCL quality parameter; ρ := territory closure parameter.

Output: A feasible assigment S.

0 $q \leftarrow 1$; $\bar{V} \leftarrow V$;
1 $V_q \leftarrow \{v\}$, where $v \in \arg\min\{|N^i| : i \in \bar{V}\}$;
2 $\bar{V} \leftarrow \bar{V} \setminus \{v\}$;
3 **while** ($\bar{V} \neq \emptyset$) **do**
4 $N(V_q) \leftarrow$ set of neighbors of V_q;
5 Compute $\phi(v)$ in eq. (8) for all $v \in N(V_q)$;
6 $\Phi_{\min} \leftarrow \min_v\{\phi(v)\}$; $\Phi_{\max} \leftarrow \max_v\{\phi(v)\}$;
7 RCL $\leftarrow \{j \in N(V_q) : \phi(j) \in [\Phi_{\min}, \Phi_{\min} + \alpha(\Phi_{\max} - \Phi_{\min})]$;
8 Choose $v \in$ RCL randomly;
9 $V_q \leftarrow V_q \cup \{v\}$; $\bar{V} \leftarrow \bar{V} \setminus \{v\}$;
10 **if** ($N(V_q) = \emptyset$ **or** $w^a(V_q) > \rho(1 + \tau^a)\mu^a$ for any a) **then**
11 $q \leftarrow q + 1$;
12 $V_q \leftarrow \{v\}$, where $v \in \arg\min\{|N^i| : i \in \bar{V}\}$;
13 $\bar{V} \leftarrow \bar{V} \setminus \{v\}$;
14 **endif**;
14 **endwhile**;
15 **return** $S = \{V_1, \ldots, V_q\}$;
end `ConstructGreedyRandomized`

Algorithm 2: The GRASP construction pseudocode.

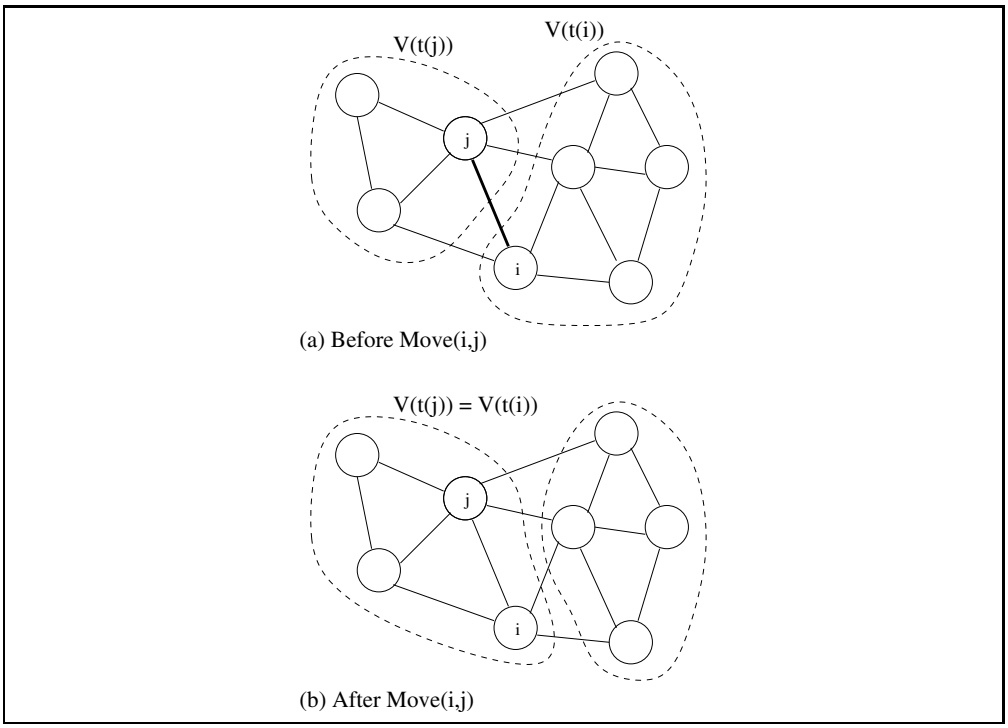

Figure 3: A feasible move.

function `GRASP_filter` (*limit_iterations*, α, ρ, β, p)
Input: *limit_iterations* := GRASP iteration limit; α := RCL quality parameter; ρ := territory closure parameter; β := filtering parameter; p := Number of territories.
Output: A feasible assigment S^{best}.

0 $S^{\text{best}} \leftarrow \emptyset$;
1 $sum \leftarrow 0$; $times \leftarrow 0$;
2 **for** ($l = 1, \ldots,$ *limit_iterations*) **do**
3 $S \leftarrow$ `ConstructGreedyRandomizedSolution(`α`, `ρ`)`; $q \leftarrow |S|$;
4 **if** ($q \neq p$) **then** $S \leftarrow$ `Adjustment(`S`)`;
5 **if** (($l < 100$) **or** ($l \geq 100$ **and** $\beta(1-\bar{\beta})\psi(S) < \psi(S^{\text{best}})$)) **then**
6 $S' \leftarrow$ `PostProcessing(`S`)`;
7 **if** ($\psi(S') < \psi(S^{\text{best}})$) **then** $S^{\text{best}} \leftarrow S'$;
8 $times \leftarrow times+1$;
9 $sum \leftarrow sum+(\psi(S) - \psi(S'))/\psi(S)$;
10 $\bar{\beta} \leftarrow sum/times$;
11 **endif**;
12 **endfor**;
13 **return** S^{best};
end `GRASP_filter`

Algorithm 4: Pseudocode of the GRASP with filter.

function `GRASP_reactive` (*limit_iterations*, *update_period*, $\mathcal{A}$, ρ, β, δ, p)
Input: *limit_iterations* := GRASP iteration limit; *update_period* := Reactive GRASP update period; $\mathcal{A} = \{\alpha_1, \ldots, \alpha_m\}$:= GRASP RCL quality parameter set; ρ := territory closure parameter; β := GRASP filtering parameter; δ := Parameter for atenuating probabilities; p := Number of territories.
Output: A feasible assigment S^{best}.

```
0   S^best ← ∅;
1   sum ← 0; times ← 0;
2   sum_i ← 0; p_i ← 1/m; n_i ← 0;
3   for (l = 1, ..., limit_iterations) do
4         Randomly select α = α_i from A using p_i's;
5         S ← ConstructGreedyRandomizedSolution(α, ρ); q ← |S|;
6         if (q ≠ p) then S ← Adjustment(S);
7         if ( (l < 100) or (l ≥ 100 and β(1 − β̄)ψ(S) < ψ(S^best)) ) then
8               S' ← PostProcessing(S);
9               if (ψ(S') < ψ(S^best)) then S^best ← S';
10              times ← times+1;
11              sum ← sum+(ψ(S) − ψ(S'))/ψ(S);
12              β̄ ← sum/times;
13              n_i ← n_i + 1;
14              sum_i ← sum_i + F(S');
15        endif;
16        if (l mod update_period = 0) then
17              A_i ← sum_i/n_i;
18              q_i ← (1/A_i)^δ;
19              p_i ← q_i/(∑ q_i);
20        endif
21  endfor;
22  return S^best;
end GRASP_reactive
```

Algorithm 5: Pseudocode of the Reactive GRASP.

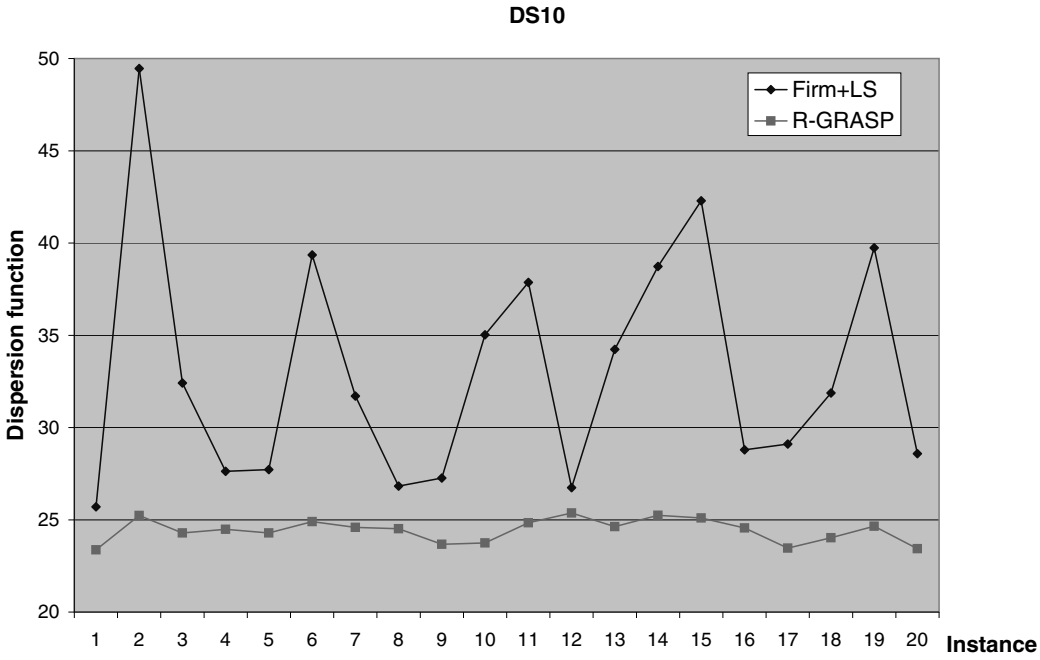

Figure 6: Comparison of dispersion function on DS10.

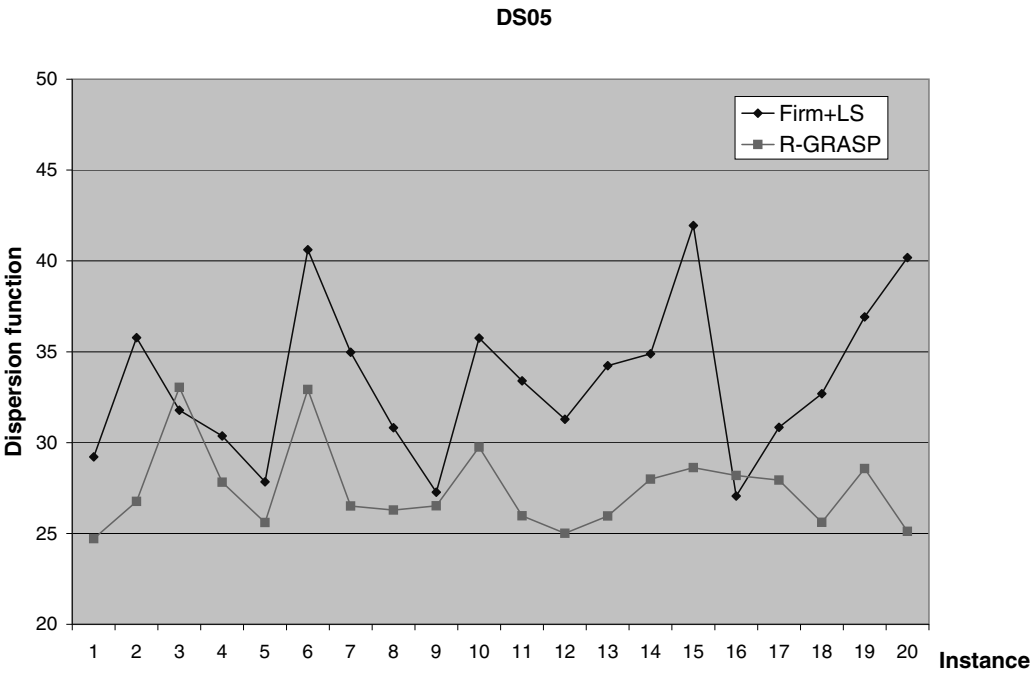

Figure 7: Comparison of dispersion function on DS05.

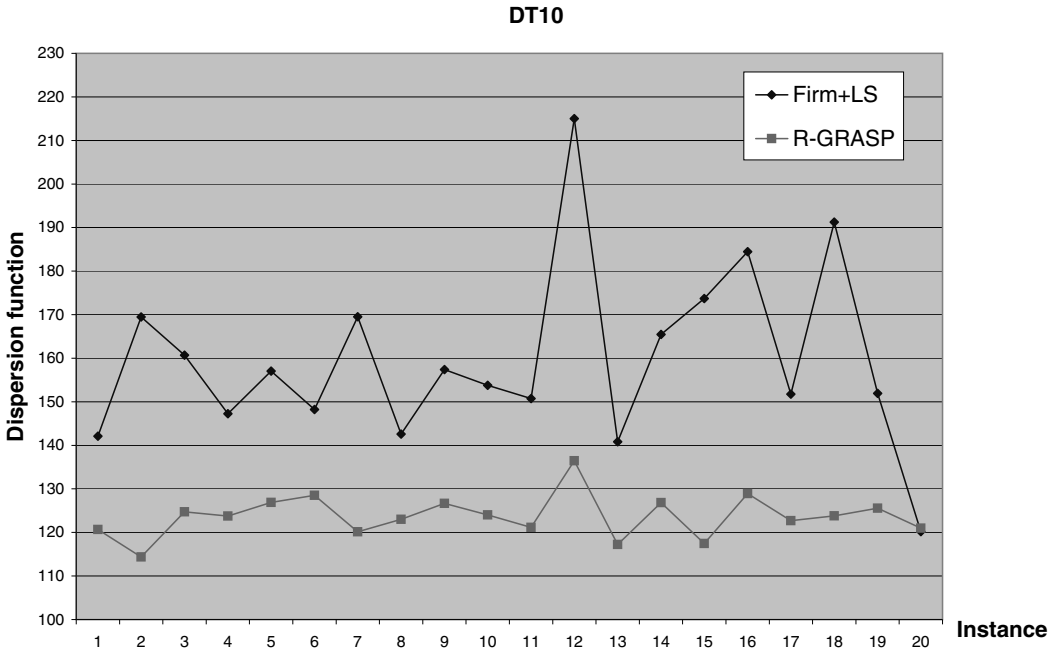

Figure 8: Comparison of dispersion function on DT10.

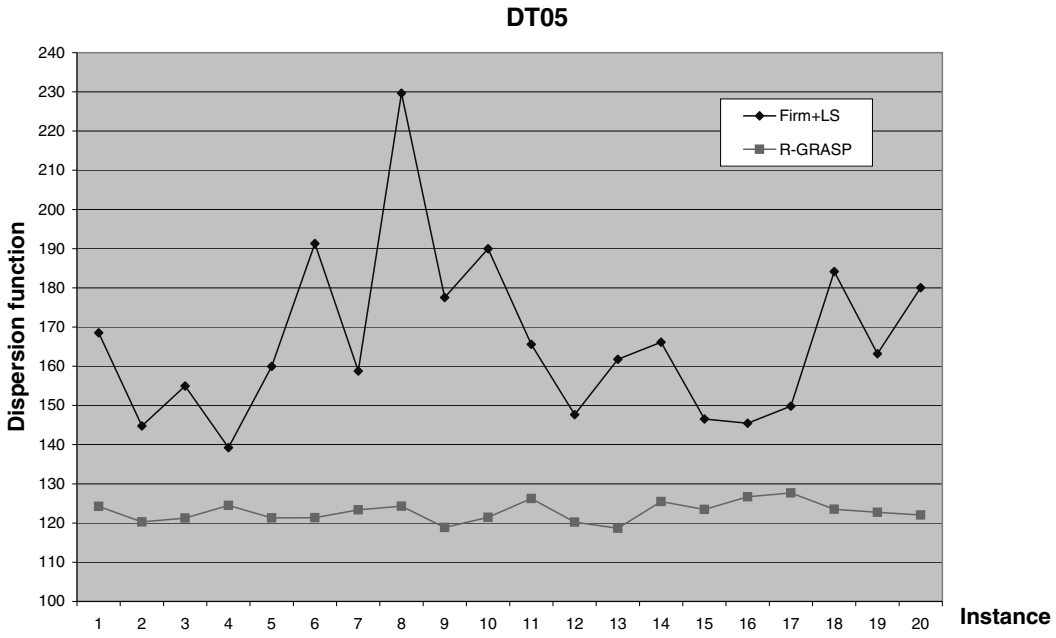

Figure 9: Comparison of dispersion function on DT05.