Journal of Classification 7:217-240 (1990)

Weight Constrained Maximum Split Clustering

P. Hansen

RUTCOR, Rutgers University

B. Jaumard

GERAD and Ecole Polytechnique de Montréal

K. Musitu

GERAD and McGill University, Montréal

Acknowledgments: Work of the first author was supported in part by AFOSR grants 0271 and 0066 to Rutgers University and was done in part during a visit to GERAD, Ecole Polytechnique de Montréal, whose support is gratefully acknowledged. Work of the second and third authors was supported by NSERC grant GP0036426 and by FCAR grant 89EQ4144. We are grateful to Silvano Martello and Paolo Toth for making available to us their program MTP for the bin-paking problem and to three anonymous referees for comments which helped to improve the presentation of the paper.

Authors' addresses: P. Hansen, Rutgers Center for Operations Research, Hill Center for the Mathematical Sciences, Rutgers University, New Brunswick, New Jersey 08903, USA; B. Jaumard, GERAD and Département de Mathématiques Appliquées, Ecole Polytechnique, Campus de l'Université de Montréal, Case Postale 6079, Succursale "A", Montréal, Québec, H3C 3A7, Canada; and K. Musitu, GERAD, 5255 avenue Decelles, Montréal, Québec, H3T 1V6, Canada.

Abstract: Consider N entities to be classified, with given weights, and a matrix of dissimilarities between pairs of them. The split of a cluster is the smallest dissimilarity between an entity in that cluster and an entity outside it. The single-linkage algorithm provides partitions into M clusters for which the smallest split is maximum. We consider the problems of finding maximum split partitions with exactly M clusters and with at most M clusters subject to the additional constraint that the sum of the weights of the entities in each cluster never exceeds a given bound. These two problems are shown to be NP-hard and reducible to a sequence of bin-packing problems. A $\Theta(N^2)$ algorithm for the particular case $M = N$ of the second problem is also presented. Computational experience is reported.

Résumé: Soient N objets à classifier, avec des poids donnés, et une matrice de dissimilarités entre paires de ces objets. L'écart d'une classe est la plus petite dissimilarité entre un objet de cette classe et un objet en dehors d'elle. L'algorithme du lien simple fournit des partitions en M classes dont le plus petit écart est maximum. Nous étudions comment obtenir des partitions d'écart maximum en exactement M classes ou en au plus M classes sous la contrainte additionnelle que la somme des poids des objets de chaque classe ne dépasse jamais une borne donnée. Nous montrons que ces deux problèmes sont NP-difficiles et réductibles à une séquence de problèmes de mise-en-boites (bin-packing). Nous proposons aussi un algorithme en $\Theta(N^2)$ pour le cas particulier $M = N$ du second problème. Enfin, nous présentons des résultats de calcul.

Keywords: Partition; Single-linkage; Split; Cluster weight; Constrained clustering.

1. Introduction

Many cluster analysis algorithms (Hartigan 1975; Späth 1980; Gordon 1981) partition the entities of a given set into homogeneous and/or well separated classes, called clusters. In some applications it is desirable for operational reasons to impose additional constraints on these clusters. Constrained clustering has been reviewed by Murtagh (1985). The constraints most often considered are bounds on the cardinality of the clusters, and connectivity constraints when regions in the plane are clustered. The former type of constraints are easily taken into account in heuristic algorithms of the exchange type (e.g., Diday et al. 1979); constraints of the latter type may be implemented in agglomerative hierarchical methods by allowing only fusion of regions having a common frontier, as proposed by many authors. This procedure severely restricts the set of feasible fusions, probably more than is required for most partitions.

Cardinality or weight constraints appear naturally in many applications. These include grouping machines into cells or jobs into families in production planning (King and Nakornchai 1982; Waghodekar and Sahu 1983; Kusiak, Vannelli and Kumar 1986), assigning files to storage devices in data management (Flory, Gunther and Kouloumdjian 1978; Bell 1984; Bell, McErlean, Stewart and Arbuckle 1988) and locating obnoxious facilities (Hsu and Nemhauser 1979).

In this paper we consider the criterion (implicitly) optimized by the single-linkage algorithm, i.e., the split of a partition, and study its maximization subject to a constraint on the sum of the weights of the entities in each cluster. This approach includes, as a particular case, maximum-split clustering with cardinality constraints on the clusters. We study both partitions into exactly M clusters and into at most M clusters; the particular case $M = N$ of the latter problem, where N denotes the number of entities of the given set, is shown to be solvable in $\Theta(N^2)$ time by a variant of the single-linkage algorithm.

The weight-constrained-maximum-split clustering problem is closely related to the bin-packing problem. The latter consists of fitting a set of entities of known size into the smallest possible number of identical bins of a given size. In fact, our algorithm involves solving a sequence of bin-packing problems.

The paper is organized as follows. Section 2 gives definitions and mathematical statement of the problem. Its complexity is analyzed in Section 3. A general algorithm is presented in Section 4. The way to solve the bin-packing problems, which arise as subproblems, is studied in Section 5. A polynomial algorithm is proposed for solving the bin-packing problems arising in the case of cardinality constraints on the clusters. An example is presented in Section 6. Computational experience is reported in Section 7, and conclusions are drawn in Section 8.

As the algorithms of this paper focus on the split of the partitions, i.e., on a measure of separation, the optimal partitions obtained should not be expected to be very homogeneous. They may also have clusters with non-contiguous sets of entities (assuming these entities are regions in the plane). The more so as several clusters obtained by the single-linkage method (which will be shown to be building blocks for the optimal constrained partitions) may be joined in a single weight-constrained cluster, which may be even less homogeneous than they are. Ways to take homogeneity and/or contiguity constraints into account in constrained clustering, possibly together with separation, are now under study.

The algorithms of this paper do provide one optimal weight-constrained partition into M or at most M clusters, and not all such optimal partitions. While it is possible to obtain them all with a much larger amount of work (i.e., considering all minimum spanning trees of the graph G associated with the given set of entities, and all optimal solutions of the bin-packing problems solved as subproblems, etc) this does not seem to be desirable. Indeed, when **threshold type** criteria, i.e., criteria whose value depends on a single dissimilarity, are used in clustering the number of optimal solutions can be, and often is, so large that enumerating all of them would be prohibitive.

2. Definitions and Problem Statement

Let $O = \{O_1, O_2, \ldots, O_N\}$ denote a set of $N = |O|$ entities and $D = (d_{kl})$ an $N \times N$ matrix of dissimilarities between pairs of entities. Dissimilarities are real numbers satisfying the usual conditions $d_{kl} \geq 0$, $d_{kk} = 0$ and $d_{kl} = d_{lk}$ for $k,l = 1,2,\ldots,N$. A partition $P_M = \{C_1, C_2, \ldots, C_M\}$ of O into M clusters satisfies the conditions $\bigcup_{j=1}^{M} C_j = O$, $C_j \neq \varnothing$ and $C_i \cap C_j = \varnothing$ for $i,j = 1,2,\ldots,M$ and $i \neq j$. Let Π_M denote the set of all partitions P_M of O into M clusters. A positive real number w_k, called **weight**, is associated with each entity O_k of O. Let $\overline{w}$ be a positive real number, larger than w_k for $k = 1,2,\ldots,N$. A **weight-constrained partition** $\overline{P}_M$ of O is a partition P_M of Π_M such that:

$$\sum_{k \mid O_k \in C_j} w_k \leq \overline{w}$$

for $j = 1,2,\ldots,M$. Let $\overline{\Pi}_M$ denote the set of all weight-constrained partitions $\overline{P}_M$ of O into M clusters. The split $s(C_j)$ of a cluster C_j is the smallest dissimilarity between an entity of O within it and one outside it:

$$s(C_j) = \min_{k,l \mid O_k \in C_j, O_l \notin C_j} d_{kl} \, ,$$

and the split $s(P_M)$ of a partition P_M is the minimum split of its clusters:

$$s(P_M) = \min_j s(C_j) \, .$$

Let $G = (V,E)$ denote a complete graph associated with the dissimilarity matrix D: vertices $v_k \in V$ correspond to entities $O_k \in O$ and edges $\{v_k, v_l\} \in E$ are weighted by the dissimilarities d_{kl} for $k,l = 1,2,\ldots,N$. A result of Rosenstiehl (1967) implies that the split of any cluster, and hence of any partition, is equal to the weight of an edge of any minimum spanning tree of G. This result implies in turn, as noted by Delattre and Hansen (1980), that the single-linkage algorithm maximizes the split of the partitions obtained at all levels of the hierarchy (see also Zahn 1971, Leclerc 1977, and Hubert 1977 for related results).

The **weight-constrained-maximum-split clustering problem**, called **Problem** $P1$ in the sequel, may be formulated:

$$\text{Determine } \bar{P}_M \in \bar{\Pi}_M$$

such that $s(\bar{P}_M)$ is maximum for a given M between 2 and N.

We call **Problem** $P2$ the variant in which the constraint on the number of clusters is weakened:

$$\text{Determine } \bar{P}_L \in \bar{\Pi}_L$$

such that $s(\bar{P}_L)$ is maximum and $L \in \{2,3,\ldots,M\}$ for a given M between 2 and N. It follows from the result of Rosenstiehl (1967) cited above that the values of $s(\bar{P}_M)$ and $s(\bar{P}_L)$, if the partitions $\bar{P}_M$ and $\bar{P}_L$ exist, are equal to dissimilarities associated with edges of a minimum spanning tree T of G. In fact, it will be shown later that the clusters of $\bar{P}_M$ or $\bar{P}_L$ can be obtained by considering some of the single-linkage algorithm ones and unions of them.

3. Complexity

We study in this section the complexity of problems $P1$ and $P2$ defined in Section 2. We first review the main aims and definitions of that theory.

Complexity theory studies the number of computations taken by an algorithm for a given problem to solve an instance of that problem, as a function of the size of this instance. (Technically, one considers an abstract model of a computer, the **Deterministic Turing Machine** and evaluates the size of a problem instance by the length of a binary encoding of its data.) Usually, complexity results are expressed as bounds on the number of computations in the **worst case**. The **average case** is sometimes also studied but is more difficult both to define in a realistic way and to analyze. Results are expressed in terms of order of magnitude. This renders them independent of the computer language or machine used (except for parallel computers which require a different theory).

Recall that a function $f(n)$ is **of the order of** a function $g(n)$, which is noted $O(q(n))$, if there exists a constant c such that $|f(n)| \leq c\,|g(n)|$ for all $n \geq 0$. A **polynomial time algorithm** is an algorithm whose number of computations, or **time complexity function**, is in $O(p(n))$ where n denotes the size of the problem and $p(n)$ is a polynomial in n. The class of problems which can be solved by a polynomial algorithm is denoted by P. Several problems of cluster analysis, such as **maximum split clustering** and **maximum sum-of-splits clustering**, belong to this class. (Note that using a polynomial algorithm does not imply one solves a problem which is in P as: (i) the algorithm may be approximate, i.e., not always find an optimal solution, or guarantee optimality when one is found, and (ii) the problem may be ill defined, i.e., have no explicit objective function.)

When a problem belongs to P, one is interested in finding an implementation of it with the lowest possible complexity, i.e., such that the polynomial $p(n)$ be of lowest possible order. To do this, each step of the algorithm must be analyzed, in terms of elementary operations such as INSERT, DELETE, FIND, MINIMUM, and the **data structure** which best allows to implement it chosen. Many sophisticated data structures, with known time and space complexities for each elementary operation are now available (see, e.g., Aho, Hopcroft and Ullman (1974) for an introduction to that subject). Lowering complexity by finding better algorithms or better implementations of known algorithm often allows to solve much larger problems than before. A well-known example of the former case in cluster analysis is Gower and Ross's (1969) discovery of an $O(N^2)$ algorithm for single-linkage (or maximum split) clustering exploiting properties of the minimum spanning tree of the graph G associated with O. Another example is Benzécri's (1982) use of chains of nearest neighbors to obtain $O(N^2)$ hierarchical agglomerative algorithms for the variance and other criteria when entities are objects in low-dimensional space and Euclidean distance is used as dissimilarity. An example of the latter case is the reduction of the complexity of the hierarchical agglomerative scheme from $O(N^3)$ to $O(N^2 \log N)$ through the introduction of priority queues, by Day and Edelsbrunner (1984). One may also consider lower bounds on the complexity of any algorithm which solves a given problem belonging to P. A problem is said to be in $\Omega(q(n))$ if the complexity of any algorithm to solve it must be at least in $O(q(n))$. For instance, dissimilarities-based clustering problems are in $\Omega(N^2)$ as it is necessary to read or compute the dissimilarities between all pairs of entities. If the complexity $O(p(n))$ of an algorithm for a given problem is of the same order as the lower bound $\Omega(q(n))$, this algorithm is **best possible**, i.e., it uses a number of computations which is lowest possible up to a constant factor. The algorithm is then said to be in $\Theta(p(n))$. The algorithms of Gower and Ross (1969) and of Benzécri (1982) cited above as well as the algorithm of Hansen, Jaumard and Frank (1989) for maximum sum-of-splits clustering are in $\Theta(N^2)$.

Many problems which are not known to be in P, i.e., for which no polynomial algorithm has been found, can be solved by some algorithm in exponential time. It is equivalent to say that they can be solved in polynomial time by an hypothetical computer which can simultaneously carry out any number of computations in parallel. (Technically, they can be solved in polynomial time by a **Nondeterministic Turing Machine**). The class of problems admitting such exponential time algorithms is denoted by NP. It follows from the definitions that P $\subset$ NP. Whether the reverse relation

$P \supset NP$ is true or not is the famous $P = NP$ conjecture. A main reason for interest in this conjecture is that a very large collection of problems (more than 10000 of them) are: (i) known to be in NP; (ii) not known to be in P; and (iii) **polynomially reducible** one to the other. The latter condition means that an instance of one of them can be transformed into an instance of any other in such a way that its size is not increased more than polynomially by this operation. Such problems are said to be NP-complete. Therefore, the existence of a polynomial algorithm for an NP-complete problem would imply the existence of polynomial algorithms for all of them. Despite many efforts, no polynomial algorithm for an NP-complete problem has yet been found and the $P = NP$ conjecture remains open.

Note that not all known problems are in NP, as it has been proved that there exists no algorithm to solve some of them (as for instance polynomial equations in integers, i.e., Hilbert's tenth problem, see Matiasevitch 1973).

Determining whether a new problem is NP-complete or not is very useful to direct efforts. If it can be shown to be in NP, it seems very unlikely that a polynomial algorithm can be found to solve it and building an exponential time algorithm (e.g., a branch-and-bound method) is a reasonable approach. Otherwise it is worth trying to find a polynomial algorithm to solve it and if one is found to lower its complexity as much as possible. Many techniques for proving that a problem is NP-complete have been devised. A very clear exposition of the main approaches is given in Garey and Johnson (1979). Probably the easiest technique is **reduction**. To show by reduction that a problem is in NP it suffices to express it as a particular case of another one known to be in NP. For instance, many clustering problems can be expressed as integer programs (see, e.g., Vinod 1969, and Rao 1971) and INTEGER PROGRAMMING is known to belong to NP. Then to show that a problem is NP-complete it is enough to express another problem known to be NP-complete in the form of the original one. Such techniques will be used below.

In complexity theory, problems are usually expressed in a way which allows to answer them by YES or NO, i.e., as **decision problems**. If an objective function is involved, i.e., if one considers an **optimization problem**, asking for the existence of a solution with a value less than a given number yields a decision problem. If the latter problem is NP-complete, the optimization one is said to be **NP-hard**. One may also be interested in the number of computations needed to find an approximate solution, with a value within $\varepsilon\%$ of the optimal one. It can be shown that if the problem remains NP-complete for instances in which all data are of a magnitude polynomially bounded in the length of the input (in which case it is said not to be a **number problem**) then finding a polynomial algorithm which guarantees to obtain an ε optimal solution would imply $P = NP$. Such problems are said to be **strongly NP-complete** (or NP-complete in the strong sense).

Finally, an NP-complete problem may or may not remain so when further restrictions are imposed on some parameters defining it. The most frequent such restriction is to assume that some parameter is fixed, i.e., does not depend on the size of the problem. For instance, we will show in the sequel of this Section and in Section 5 respectively that CARDINALITY CONSTRAINED MAXIMUM SPLIT CLUSTERING is NP-complete for M unrestricted and polynomial for M fixed. It is equivalent, and more common in complexity theory terminology, to say that M depends on N (a characteristic of problem size) in the former case and is independent of N in the latter.

We now study the complexity of problems $P1$ and $P2$ of Section 2. We first express $P1$ as a decision problem.

Weight-Constrained-Maximum-Split Clustering (WCMSC):

Instance: Set $O = \{O_1, O_2, \ldots, O_N\}$ of $N = |O|$ entities; positive real numbers $w_1, w_2, \ldots, w_N$; $N \times N$ matrix $D = (d_{kl})_{k,l=1,2,\ldots,N}$ of dissimilarities associated with the pairs of entities $\{O_k, O_l\}$ belonging to O; positive integer M; positive real numbers $\bar{w}$ and $\bar{s}$.

Question: Is there a partition $P_M = \{C_1, C_2, \ldots, C_M\}$ of O into M clusters such that $\sum_{k/O_k \in C_j} w_k \le \bar{w}$ and $s(C_j) \ge \bar{s}$ for $j = 1,2,\ldots,M$?

We then have:

Theorem 1. *Weight-Constrained-Maximum-Split Clustering is NP-complete.*

Proof. As WCMSC can easily be expressed as a linear program in 0-1 variables, it is in NP. To show it is NP-complete, we use reduction to the NP-complete problem PARTITION (Garey and Johnson 1979, pp. 223):

Instance: N **positive real numbers** $r_1, r_2, \ldots, r_N$.

Question: Is there a subset I of $I_N = \{1,2,\ldots,N\}$ such that $\sum_{k \in I} r_k = \sum_{k \in I_N \setminus I} r_k$?

Taking $M = 2, \bar{s} = 0, w_k = r_k$ for $k = 1,2,\ldots,N, \bar{w} = \frac{1}{2} \sum_{k=1}^{N} w_k$ and any D reduces the weight-constrained-maximum-split clustering problem to the problem PARTITION. •

Note that this proof is also valid if Problem $P2$ is considered instead of Problem $P1$.

The particular case where $w_k = 1$ for all k corresponds to cardinality constraints on the clusters. We now express $P1$ as a decision problem in that case:

Cardinality-Constrained-Maximum-Split Clustering (CCMSC):

Instance: Set $O = \{O_1, O_2, \dots, O_N\}$ of $N = |O|$ entities; $N \times N$ matrix $D = (d_{kl})_{k,l=1,2,\dots,N}$ of dissimilarities associated with the pairs of entities $\{O_k, O_l\}$ belonging to O; positive integers $M, \overline{w}$; positive real number $\overline{s}$.

Question: Is there a partition $P_M = \{C_1, C_2, \dots, C_M\}$ of O into M clusters such that $|C_j| \le \overline{w}$ and $s(C_j) \ge \overline{s}$ for $j = 1,2,\dots,M$?

In this case, problems $P1$ and $P2$ are solvable in polynomial time for fixed M. Indeed, polynomial algorithms for that case are presented in Section 5. If, however, M depends on N, problems $P1$ and $P2$ are again NP-complete, as we now show:

Theorem 2. *Cardinality-Constrained-Maximum-Split Clustering is strongly NP-complete.*

Proof: As Cardinality-Constrained-Maximum-Split Clustering is a particular case of WCMSC, it is in NP. To show that it is NP-complete, we use reduction to the problem 3-PARTITION (Garey and Johnson 1979, pp. 224) which is NP-complete in the strong sense, i.e., which remains NP-complete even when the largest coefficient is bounded by a polynomial in the input length.

Instance: Set A of $3m$ elements, a bound $B \in \mathbb{Z}^+$ and a size $s(a) \in \mathbb{Z}^+$ for each $a \in A$ such that $\frac{B}{4} < s(a) < \frac{B}{2}$ and $\sum_{a \in A} s(a) = mB$.

Question: Can A be partitioned into m disjoints sets $A_1, A_2, \dots, A_m$ such that for $1 \le i \le m$, $\sum_{a \in A_i} s(a) = B$ (note that each A_i must therefore contain exactly three elements from A)?

Consider a set O of mB entities and the vertex set V of the corresponding graph. Partition V arbitrarily into subsets of size $s(a)$ for all $a \in A$. Build a spanning tree arbitrarily on the vertex set of each subset. The union of these trees forms a forest, possibly with trees reduced to a single vertex. Associate dissimilarities equal to 0 to all edges of these trees. Add edges with dissimilarities equal to 1 until a complete graph is obtained. Set $w_j = 1$ for all j, $\overline{w} = B$, $M = m$, $\overline{s} = 1$. The CCMSC problem so defined admits a constrained partition with a split of 1 if and only if the 3-PARTITION problem has a solution. Moreover, (i) a solution to the CCMSC problem can be obtained in time polynomial in its size (by the algorithm of Section 5), which is in turn polynomial in the length of the input of the 3-PARTITION problem for instances with coefficients $s(a)$ polynomially bounded by this input length; (ii) the length of the CCMSC problem input is longer than that of the 3-PARTITION problem and (iii) the maximum size of a coefficient in the CCMSC problem is polynomially bounded in the length and size of 3-PARTITION. So all conditions (see Garey and Johnson 1979, pp. 101) for a pseudo-polynomial transformation of the former problem into the latter are satisfied. This result implies that Cardinality-Constrained-Maximum-Split Clustering is strongly NP-complete. •

Again, the proof remains valid if Problem $P2$ is considered instead of Problem $P1$.

4. General Algorithm

We now describe an algorithm for weight-constrained-maximum-split clustering, e.g., Problem $P1$, which involves the resolution of a sequence of bin-packing problems. In the next section, we discuss how to solve these latter problems when maximum weight constraints and when cardinality constraints are imposed.

The algorithm makes use of the minimum spanning tree T of G, and of spanning forests $F(s)$ obtained by deleting from T all edges associated with dissimilarities larger than or equal to a given value s (i.e., the current candidate value for the maximum split). Let $V(T)$ denote the set of vertices of the tree T. A spanning forest F of G is thus a set of trees $T_1, T_2, \dots, T_f$ such that $V(T_i) \cap V(T_j) = \emptyset$, for $i,j = 1,2,\dots,f$ and $i \neq j$, and $\bigcup_{i=1}^{f} V(T_i) = V(G)$; we allow $|V(T_i)| = 1$, i.e., the forest F may contain isolated vertices. Each spanning forest $F(s)$ corresponds to a partition of O obtained with the single-linkage algorithm, where each tree of the forest corresponds to a cluster. Weight-constrained clusters will be obtained by joining clusters associated with trees T_j, T_k.. of $F(s)$. (In other words, the partition corresponding to $F(s)$ is that one defined by cutting the single-linkage dendrogram at level $s + \varepsilon$ where ε is a small positive number, and weight-constrained clusters are obtained by joining clusters from this partition.)

The principle of the algorithm is to first find a set of candidate values for the maximum split, i.e., the dissimilarities associated with the edges of a minimum spanning tree T of G, (steps (a) and (b) below) and then to search systematically for the maximum feasible value s^* of the split (step (c)). This is done by testing values s within a dichotomous search procedure (governed by steps (c1) and (c2)). For each tentative value, after $F(s)$ is built (step (c3)) one considers several necessary conditions for feasibility and proceeds to the next value of s if one of them is violated: (i) $F(s)$ should not have too few components, i.e., less than M (step (c4)); (ii) no component of $F(s)$ should have too large a weight, i.e., a weight larger than $\bar{w}$ (step (c5)); (iii) the minimum number of clusters of weight not larger than $\bar{w}$ needed to group all clusters corresponding to $F(s)$ should not exceed M (step (c6)). If these conditions are satisfied and possibly after modifying the partition into clusters of weight not larger than $\bar{w}$, if there are less than M of them (steps (c7) and (c9)), one computes the value of the split of the current feasible partition, updates the best value known for the split and the best known solution and proceeds to the next tentative value.

We now state formally the rules of the algorithm, which we call algorithm WCMSC1:

(a) **Minimum Spanning Tree**
Determine a minimum spanning tree T of G using, e.g., Prim's (1957) algorithm.

(b) **Candidate values for the maximum split**
Rank the $N-1$ dissimilarities associated with the edges of T by order of non-decreasing values. Scan the list thus obtained and eliminate all repeated values. Let $d_1 < d_2 < ... < d_l$ denote the remaining values.

(c) **Dichotomous search for the maximum split**

(c1) **Initialization**
Set HIGH to l, i to 0 , s^* to 0 and BOOL to true.

(c2) **Current candidate value**
If BOOL is true, set LOW to $i+1$ otherwise set HIGH to $i-1$. If HIGH < LOW, s^* is the maximum split value: stop. Set i to $\left[\frac{\text{LOW}+\text{HIGH}}{2}\right]$ (where $[a]$ denotes the largest integer not larger than a) and s to d_i.

(c3) **Spanning forest**
Build the spanning forest $F(s) \equiv \{T_1, T_2, \ldots, T_{f(s)}\}$. Let $W_j = w(T_j) = \sum_{k/v_k \in V(T_j)} w_k$ denote the weight of tree T_j, i.e., the sum of the weights of its vertices. (In case of cardinality constraints $W_j = |V(T_j)|$.)

(c4) **First feasibility test**
If $f(s) < M$ set BOOL to false and return to (c2).

(c5) **Second feasibility test**
If for some $j = 1,2, \ldots, f(s)$, $W_j > \bar{w}$, set BOOL to false and return to (c2).

(c6) **Bin-packing problem**
Solve the bin-packing problem:

$$\text{minimize } z = \sum_{l=1}^{f(s)} y_l$$

$$\text{subject to: } \begin{cases} \sum_{j=1}^{f(s)} W_j x_{jl} \leq \overline{w} \cdot y_l & l = 1,2,\ldots,f(s) \\ \sum_{l=1}^{f(s)} x_{jl} = 1 & j = 1,2,\ldots,f(s) \\ x_{jl} \in \{0,1\} & j,l = 1,2,\ldots,f(s) \\ y_l \in \{0,1\} & l = 1,2,\ldots,f(s). \end{cases}$$

where $x_{jl} = 1$ if entity j (here tree T_j) goes into bin l (here cluster l) and $x_{jl} = 0$ otherwise, $y_l = 1$ if bin l is used and $y_l = 0$ otherwise. If the optimum value z^*, i.e., number of bins used, does exceed M, set BOOL to false and return to (c2).

(c7) **Test on the number of bins**
If $z^* < M$, go to (c9).

(c8) **Split of the current partition** Determine the clusters of the partition P^* corresponding to the optimal solution of the bin-packing problem, i.e.,

$$C_l = \bigcup_{j/x_{jl}=1} \{O_k \mid v_k \in T_j\}, \quad l = 1,2,\ldots,f(s) \text{ for } y_l = 1 .$$

(There are at most M non-empty clusters.) Compute the split $s^* = s(P^*)$ of the partition, i.e., $s^* = \min_{l/y_l=1} s(C_l)$, and store the partition P^* as the incumbent one. Let i denote the index such that $s^* = d_i$, set BOOL to true and return to (c2).

(c9) **Modification of the current partition to obtain M clusters**
Let F^* denote the forest obtained from T by deleting all edges of T whose endpoints belong to different clusters of P^*. Then consider the clusters of P^* which contain more than one connected component of F^*. Assign connected components one at a time to new clusters until a new partition P^* with M clusters is obtained, if possible. If the number of clusters z^* of P^* is still smaller than M when each cluster contains a single connected component of F^*, consider the edges of T joining two trees of $F(s)$ which belong to the same cluster. Rank them by

order of non-increasing dissimilarities. Cut the $M - z^*$ edges with largest dissimilarities, thus obtaining a partition P^* into M clusters. Set s^* to the split $s(P^*)$ of this partition and store it as the incumbent one. Let i denote the index such that $s^* = d_i$, set BOOL to true and return to (c2).

Note that the absence of any feasible partition of O into M clusters is indicated by an optimal value $s^* = 0$. If Problem $P2$ is considered instead of $P1$, algorithm WCMSC1 can readily be used, after deletion of the tests (c4), (c7) and (c9). We then call it algorithm WCMSC2. Algorithm WCMSC1 is illustrated with an example in Section 6.

We then have:

Theorem 3. *Algorithm WCMSC1 (resp. WCMSC2) solves Problem* $P1$ *(resp. Problem* $P2$*) in a finite time.*

Proof: Correctness follows from that, as shown above, there are at most $N - 1$ candidate values for the split of the optimal partition, which are all considered implicitly in the dichotomous search. Then for each candidate value explicitly considered either a partition into M clusters (resp. at most M clusters) is shown not to exist as a necessary condition for its existence does not hold, or one is found and evaluated. The best solution found being updated, an optimal partition for Problem $P1$ (resp. Problem $P2$) is found when the algorithm stops. Finiteness follows from that all tests except (c6) take polynomial time, the bin packing problem of test (c6) can be solved by a non-polynomial but finite algorithm and all tests are performed in sequence unless one proceeds to examine the next candidate value for the optimum split. •

Algorithm WCMSC1 determines a weight-constrained-maximum-split partition P^* after solving at most $\log_2 N$ bin-packing problems. This upper bound is not necessarily reached since the answer to the current problem allows us to eliminate one candidate value at each iteration in addition to cutting the range in two, and because the split $s(P^*)$ of the partition P^* obtained in Step (c6) or Step (c9) may be larger than the current candidate value, thus allowing us to eliminate all candidate values between these two.

Note also that it is not necessary to compare $s(P^*)$ with the value of the incumbent partition as, because of the rules of dichotomous search, the current value of s is not smaller than any previously examined one which led to a feasible partition, and, as explained above, $s(P^*) \geq s$.

In the particular case when $M = N$, Problem $P2$ reduces to maximum-split partitioning subject to weight constraints only. Again algorithm WCMSC1 can be used, this time after deletion of tests (c4), and (c6) to (c9), leading to algorithm WCMSC3. This algorithm uses $O(N^2)$ operations and, as the computation of the minimum spanning tree T requires $\Omega(N^2)$ operations, it is in $\Theta(N^2)$. The number of computations of algorithm WCMSC3 is thus minimum, up to a constant factor.

5. Solving the Bin-packing Problem

Recall that the bin-packing problem consists of fitting a set of entities of known size into the smallest possible number of bins of a given size. The two algorithms of the previous section involve solving a sequence of bin-packing problems (expressed as decision problems, i.e., for given M), with real data in the case of weight constraints and with bounded integer data in the case of cardinality constraints. The bin-packing problem is NP-hard in the former case, as well as in the latter one except for fixed bin capacity (Garey and Johnson 1979, p. 226) or for fixed M, as discussed below. However, for small values of M, many instances of the bin-packing problem are easy to solve, as solutions with

$$\left\lceil \frac{\sum_{k=1}^{N} w_k}{\bar{w}} \right\rceil$$

bins (where $\lceil a \rceil$ denotes the smallest integer not smaller than a) are often easy to find. If this last number is greater than M, the bin packing problem has no solution. Otherwise, if $\sum_{k=1}^{N} w_k$ is substantially smaller than $M\,\bar{w}$, a heuristic often suffices to find a feasible solution, but if $\sum_{k=1}^{N} w_k$ is only slightly smaller than $M\,\bar{w}$, the bin-packing problem becomes a difficult combinatorial one.

Many heuristic algorithms for the bin-packing problem have been devised. Their behavior in the worst case and in the average case has been extensively studied. Coffman, Garey and Johnson (1984) give a detailed survey of this work. An easy-to-use and efficient heuristic is "First-Fit Decreasing" which works as follows: "Rank the entities (here the trees of the forest $F(s)$) by order of non-increasing weights. Introduce each entity in the first bin in which it fits."

Should such a heuristic be unable to provide a feasible solution with M clusters (or with at most M clusters), an exact branch-and-bound algorithm can be used. Martello and Toth (1989, 1990) have derived sharp bounds and reduction procedures for the bin-packing problem. They have used them to design an efficient branch-and-bound algorithm and the corresponding program, which we have used in our computational experiments.

In the case of cardinality constraints, a polynomial algorithm may be used to solve the bin-packing problems, for small values of M. Let us call *profile* of a partition the list of the weights of its clusters, ranked in order of non-increasing values. The following algorithm proceeds to an enumeration of all feasible profiles, for partitions of sets of k trees, $k = 1,2,\ldots,f(s)$, of the current forest $F(s)$.

Algorithm CCBP (Cardinality-Constrained Bin-Packing)

- (a) **Ranking of trees**
  Rank the trees $T_1, T_2, \ldots, T_{f(s)}$ of the current forest in order of non-increasing weights. Reindex them according to this order.
- (b) **Initial profile**
  Let L denote the current list of profiles. Initialize L to P_1 where $P_1(j) = 0, j = 1,2,\ldots,M$. Set k to 1.
- (c) **Profiles of the partitions**
  - (c1) **New profiles from old ones**
    Set L' to $\varnothing$. Consider in turn each profile P_l of L. Consider in turn each distinct value $P_l(j)$ of P_l: if $P_l(j) + w(T_k) < \bar{w}$: (i) define a new profile P_r by adding $w(T_k)$ to $P_l(j)$ and by reordering the components if necessary; (ii) insert profile P_r into L' unless P_2 is identical to a profile already in L'.
  - (c2) **Feasibility test**
    If $L' = \varnothing$, end: the bin-packing problem has no solution.
  - (c3) **New iteration**
    If $k < f(s)$, set L to L', increase k by 1 and return to (c1).
- (d) **Optimality tests**
  - (d1) **Absence of empty clusters**
    Check if L' contains a profile P_r such that $P_r(j) \neq 0$, $j = 1,2,\ldots,M$. If it is not the case, stop: the bin-packing problem has no feasible solution with M non-empty bins.
  - (d2) **Optimal partition**
    Otherwise, an optimal profile has been found and a partition can be deduced by retracing any sequence of additions performed at step (c1) and leading to it.

Step (d2) is easily done if a graph G_1 is built with vertices associated with profiles generated by algorithm CCBP and arcs joining pairs of vertices corresponding to profiles before and after the addition of a tree.

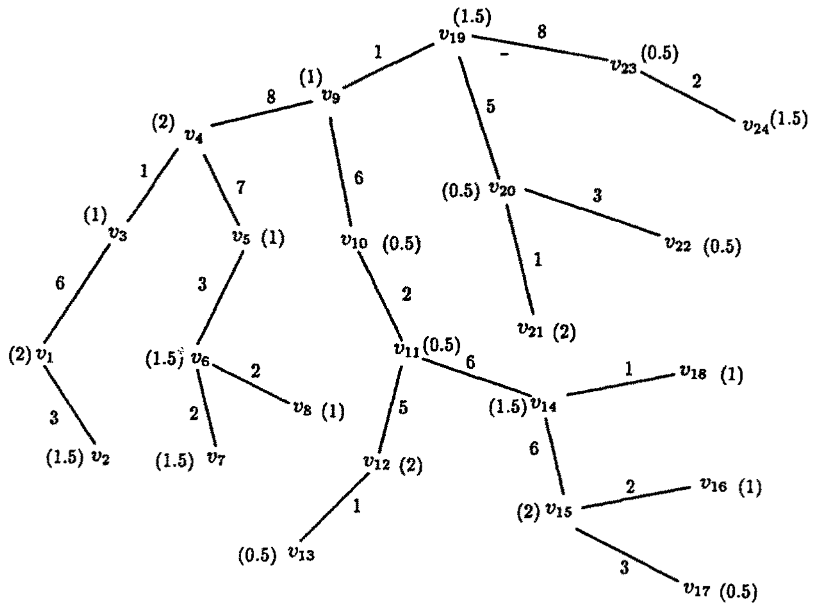

Figure 1. Minimum Spanning Tree T.

If Problem $P2$ is considered, the previous algorithm readily applies after deletion of step (d1).

The number of profiles is bounded by N^M and the number of additions by MN^M. After each addition, reordering of components can be done in $O(\log M)$ time (assuming the ranking is described by double chaining) and checking whether the profile is already in the list L' can be done in $O(M \log N)$ time, using an AVL-tree (Knuth 1973, pp. 451-469). Hence the overall complexity of algorithm CCBP is $O(M^2 N^M \log N)$. Clearly, while CCBP is a polynomial algorithm, it may require a large amount of computing time for large N and $M \geq 3$.

Note that algorithm CCBP remains polynomial if the weights w_i of the entities are polynomially bounded in the input length; however its complexity, which is already large in the case of unit weights, then drastically increases.

6. Example

We consider a set O with 24 entities and seek a maximum-split partition $\bar{P}_3$ of O into 3 clusters with a weight not exceeding 10 (Problem $P1$). The vector of weights of the entities is w = (2, 1.5, 1, 2, 1, 1.5, 1.5, 1, 1, 0.5, 0.5, 2, 0.5, 1.5, 2, 1, 0.5, 1, 1.5, 0.5, 2, 0.5, 0.5, 1.5). A minimum spanning tree T of the graph $G = (V,E)$ associated with O is represented in Figure 1 (it may not be unique but its choice, should there be several, does not affect the results). Dissimilarities which are candidate values for the maximum split are indicated in this figure along the edges of T. Weights of the entities are given between parentheses. The distinct candidate values for the maximum split are 8, 7, 6, 5, 3, 2, and 1.

Applying dichotomous search, we first consider the tentative value 5. The spanning forest $F(5)$ is represented in Figure 2. It contains 10 trees with a weight vector (5, 3, 3.5, 3.5, 3, 2.5, 2, 1, 2.5, 2.5). The First Fit Decreasing heuristic (Johnson 1974) yields a partition into the three clusters $C_1 = \{O(T_1), O(T_3), O(T_8)\}$, $C_2 = \{O(T_4), O(T_2), O(T_5)\}$ and $C_3 = \{O(T_6), O(T_7), O(T_9), O(T_{10})\}$ where $O(T_i) = \{O_j \mid v_j \in V(T_i)\}$. The split of this partition is equal to 5. The next tentative value for the maximum split is 7. The forest $F(7)$ is represented in Figure 3 in full and dotted lines.

As this forest contains a tree T_1 with a weight of 15 > 10, the tentative value 7 is too high. We then consider the tentative value 6. The forest $F(6)$ is represented in Figure 3 in full lines. It contains 8 trees with a weight vector (5.5, 5, 3.5, 3.5, 3.5, 3, 2.5, 2). The First Fit Decreasing heuristic gives a solution with 4 bins, which is infeasible. We therefore consider the bin-packing problem:

$$\min \sum_{l=1}^{8} y_l$$

subject to:

$$5.5x_{1l} + 5x_{2l} + 3.5x_{3l} + 3.5x_{4l} + 3.5x_{5l} + 3x_{6l} + 2.5x_{7l} + 2x_{8l} \le 10y_l \qquad l = 1,2,\ldots,8$$

$$x_{j1} + x_{j2} + x_{j3} + x_{j4} + x_{j5} + x_{j6} + x_{j7} + x_{j8} = 1 \qquad j = 1,2,\ldots,8$$

$$x_{jl} \in \{0,1\} \qquad j,l = 1,2,\ldots,8$$

$$y_l \in \{0,1\} \qquad l = 1,2,\ldots,8,$$

and solve it using the branch-and-bound algorithm of Martello and Toth (1989, 1990). An optimal solution is $y_1^* = y_2^* = y_3^* = 1$, $y_4^* = y_5^* = y_6^* = y_7^* = y_8^* = 0$, $x_{11}^* = x_{71}^* = x_{81}^* = x_{32}^* = x_{42}^* = x_{62}^* = x_{23}^* = x_{53}^* = 1$ and all other $x_{jl}^* = 0$. There is thus a partition into 3 clusters $C_1 = \{O(T_1), O(T_7), O(T_8)\}$ $= \{O_9, O_{14}, O_{18}, O_{19}, O_{20}, O_{21}, O_{22}, O_{23}, O_{24}\}$ of weight 10, $C_2 = \{O(T_3), O(T_4), O(T_6)\} = \{O_3, O_4, O_{10}, O_{11}, O_{12}, O_{13}, O_{15}, O_{16}, O_{17}\}$

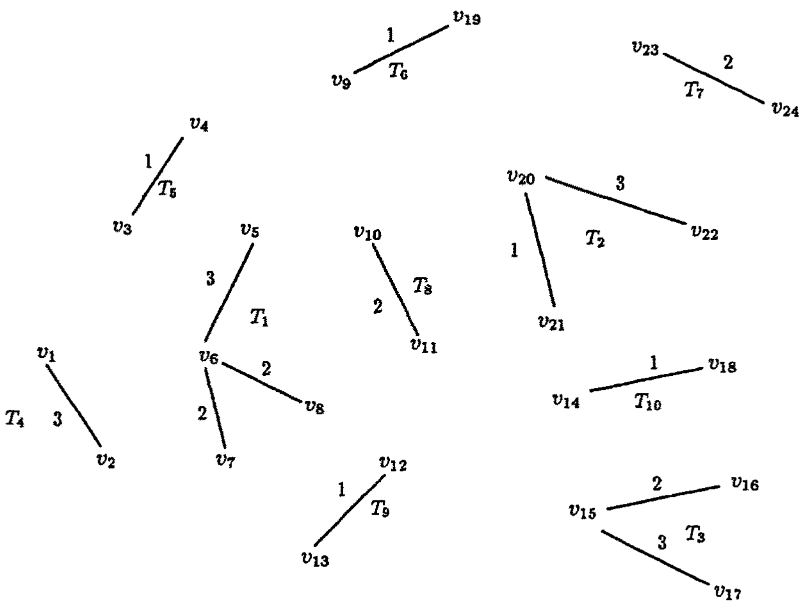

Figure 2. Spanning Forest *F(5)*.

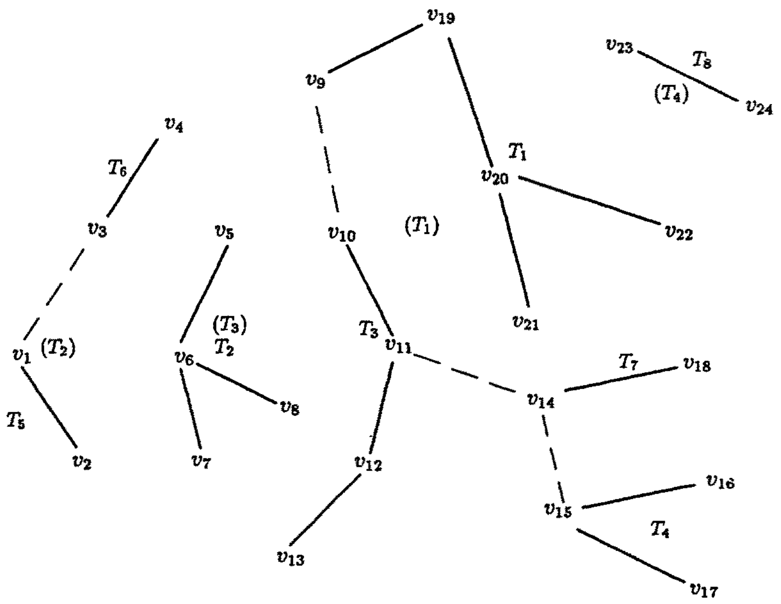

Figure 3. Spanning Forests *F(6)* (full lines) and *F(7)* (full and dotted lines).

of weight 10 and $C_3 = \{O(T_2), O(T_5)\} = \{O_1, O_2, O_5, O_6, O_7, O_8\}$ of weight 8.5, with a split of 6. As there are no more tentative values for the maximum split, this partition is optimal.

To illustrate the use of the enumerative algorithm for bin-packing of Section 5, we consider the same example as above, but with cardinality constraints instead of weight constraints. We bound the number of entities in any cluster by 8. Applying the First Fit Decreasing heuristic to the trees in the forest $F(5)$ again yields an infeasible solution using 4 bins. So we proceed to the enumeration of feasible profiles, after ranking the trees by order of decreasing cardinalities. This enumeration is summarized in Figure 4.

As a profile equal to $\begin{bmatrix}8\\8\\8\end{bmatrix}$ is obtained, the bin-packing problem admits solutions using 3 bins. The partition corresponding to the highest path between $\begin{bmatrix}0\\0\\0\end{bmatrix}$ and $\begin{bmatrix}8\\8\\8\end{bmatrix}$ goes through the profiles $\begin{bmatrix}4\\0\\0\end{bmatrix}$ $\begin{bmatrix}4\\3\\0\end{bmatrix}$ $\begin{bmatrix}6\\4\\0\end{bmatrix}$ $\begin{bmatrix}8\\4\\0\end{bmatrix}$ $\begin{bmatrix}8\\6\\0\end{bmatrix}$ $\begin{bmatrix}8\\8\\0\end{bmatrix}$ $\begin{bmatrix}8\\8\\2\end{bmatrix}$ $\begin{bmatrix}8\\8\\4\end{bmatrix}$ and $\begin{bmatrix}8\\8\\6\end{bmatrix}$. It consists of the clusters $C_1 = \{O(T_2), O(T_3), O(T_4)\} = \{O_1, O_2, O_{15}, O_{16}, O_{17}, O_{20}, O_{21}, O_{22}\}$, $C_2 = \{O(T_1), O(T_5), O(T_6)\} = \{O_3, O_4, O_5, O_6, O_7, O_8, O_9, O_{19}\}$ and $C_3 = \{O(T_7), O(T_8), O(T_9), O(T_{10})\} = \{O_{10}, O_{11}, O_{12}, O_{13}, O_{14}, O_{18}, O_{23}, O_{24}\}$.

The next tentative value for the split is 7 but $F(7)$ contains a tree T_1 with $|V(T_1)| = 14 > 8$, thus the cardinality constraint cannot be satisfied. We consider the last tentative value for the maximum split, i.e., 6. The forest $F(6)$ contains 8 trees with cardinalities $5, 4, 4, 3, 2, 2, 2$ and 2 (See Figure 3). The First Fit Decreasing heuristic provides a solution with 3 bins. The corresponding partition consists of the clusters $C_1 = \{O(T_1), O(T_4)\} = \{O_9, O_{15}, O_{16}, O_{17}, O_{19}, O_{20}, O_{21}, O_{22}\}$, $C_2 = \{O(T_2), O(T_3)\} = \{O_5, O_6, O_7, O_8, O_{10}, O_{11}, O_{12}, O_{13}\}$ and $C_3 = \{O(T_5), O(T_6), O(T_7), O(T_8)\} = \{O_1, O_2, O_3, O_4, O_{14}, O_{18}, O_{23}, O_{24}\}$ and has a split of 6. As this value is the last candidate, it is optimal.

7. Computational results

The algorithms WCMSC1 and WCMSC2 have been implemented in Fortran 77 on a SUN 3/50 system. The example of Section 6 was first solved, with general and with unit weights, in a few hundredths of a second of CPU time in both cases. We then considered data due to Späth (1980, pp. 91-92) on 89 postal zones in Germany. It is easy to think of various technical or administrative reasons why it might be desirable to cluster them, while taking into account a constraint on the total workload in each cluster. Measures of the workload in each zone not being available, the corresponding weights were randomly generated from a uniform distribution on [1,500]. The results

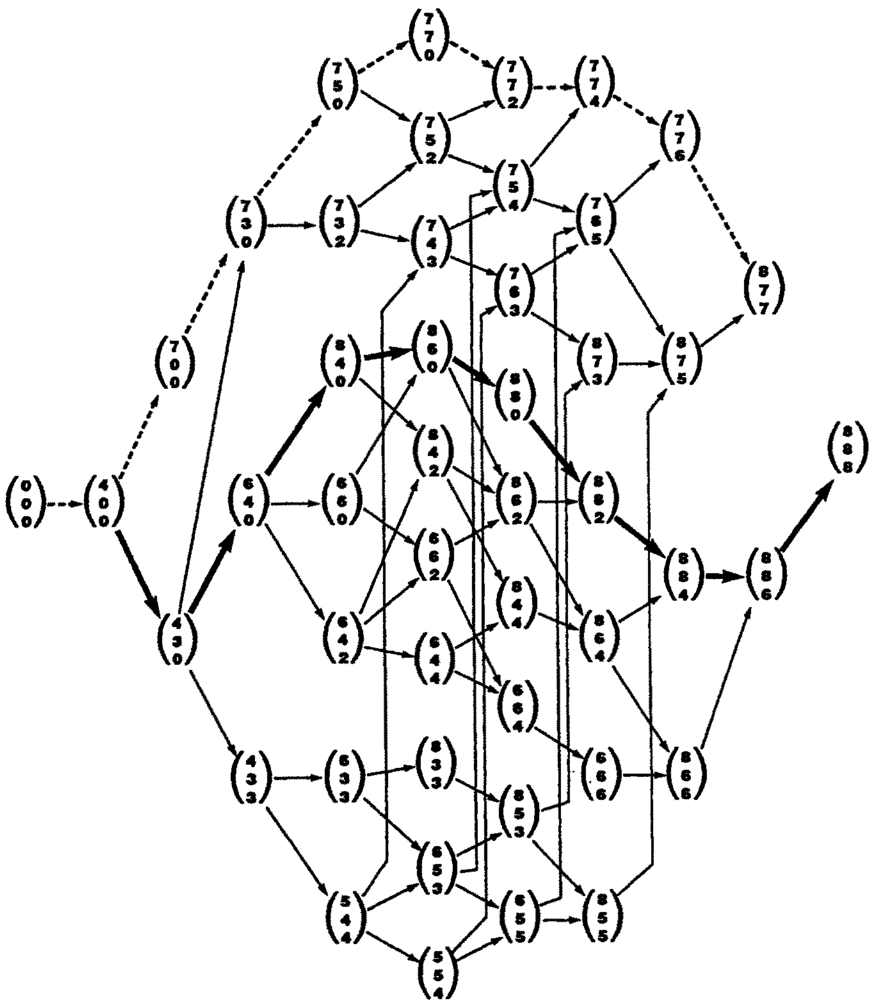
Figure 4. Enumeration of feasible profiles.

Table 1

$\overline{w}$	Bound on M	M^*	$s(P^*)$	cpu time	Fixed M	$s(P^*)$	cpu time
3000	$M \leq 7$	–	–	0.30	7	–	0.28
	$M \leq 8$	8	15.46	0.24	8	15.46	0.20
	$M \leq 9$	8	15.46	0.20	9	15.46	0.22
4000	$M \leq 5$	–	–	0.30	5	–	0.30
	$M \leq 6$	6	21.73	0.28	6	21.73	0.30
	$M \leq 7$	6	21.73	0.26	7	21.73	0.24
8000	$M \leq 2$	–	–	0.32	2	–	0.32
	$M \leq 3$	3	39.20	0.28	3	39.20	0.22
	$M \leq 4$	3	39.20	0.26	4	39.20	0.24
	$M \leq 5$	3	39.20	0.24	5	39.20	0.26
	$M \leq 15$	3	39.20	0.28	15	33.88	0.26

Table 2

$\overline{w}$	Bound on M	M^*	$s(P^*)$	cpu time
10000	$M \leq 25$	–	–	5.66
	$M \leq 26$	26	3.40	4.98
20000	$M \leq 12$	–	–	5.58
	$M \leq 15$	13	4.17	5.16
30000	$M \leq 7$	–	–	5.76
	$M \leq 8$	–	–	5.64
	$M \leq 9$	9	4.57	5.20
	$M \leq 10$	9	4.57	5.10
40000	$M \leq 7$	7	4.57	5.10
	$M \leq 8$	7	4.57	5.14
50000	$M \leq 5$	–	–	5.58
	$M \leq 6$	6	5.46	4.78

are summarized in Table 1. It appears that: (i) computation times are about 1/3 of a second of CPU time for values of $\bar{w}$ and M leading to a feasible problem, and slightly less than twice that time for values leading to an infeasible problem; (ii) when an upper bound on M is given, the best value for the split is obtained for the smallest number of clusters for which the problem is feasible; (iii) when M is fixed and the problem is feasible the split is equal to that obtained for the smallest number of clusters leading to a feasible solution, unless M is much larger than that number.

Finally, we generated a problem with 500 entities, drawing dissimilarities and weights randomly from uniform distributions, the latter one on [1,1000]. Results are summarized in Table 2 and lead to conclusions similar to those resulting from the analysis of Späth's (1980) data. Note in particular that computation times remain very moderate.

8. Conclusions

Constraints on the weight or the cardinality of the clusters appear to be natural ones in several applications of cluster analysis. We have studied how to maximize the split of a partition of the given set of entities into exactly M or at most M clusters, as done by the single-linkage algorithm in the unconstrained case, subject to such constraints. We have shown that these problems are NP-hard, except for fixed M with cardinality constraints or for $M = N$ with weight constraints. This worst case analysis implies that in some, possibly rare, cases computational times could be large. However, the exact branch-and-bound algorithms which we propose, appear to be very efficient. Large problems with 500 entities have been solved in a few seconds on a small computer, i.e., in time comparable to that of the single-linkage algorithm. This excellent empirical behaviour shows that many, and probably most, instances of weight or cardinality constrained maximum split clustering problems are amenable to exact solution.

References

AHO, A. V., HOPCROFT, J. E., and ULLMAN, J. D. (1974), *The Design and Analysis of Computer Algorithms*, Reading: Addison-Wesley.

BELL, D. A. (1984), "Physical Record Clustering in Databases," *Kybernetes, 13*, 31-37.

BELL, D. A., MCERLEAN, F. J., STEWART, P. M., and ARBUCKLE, W. (1988), "Clustering Related Tuples in Databases," *The Computer Journal, 31(3)*, 253-257.

BENZECRI, J. P. (1982), "Construction d'une classification ascendante hiérarchique par la recherche en chaîne des voisins réciproques," *Les Cahiers de l'Analyse des Données, (VII)2*, 209-218.

COFFMAN, E. G. Jr, GAREY, M. R., and JOHNSON, D. S. (1984), "Approximation Algorithms for Bin-packing — An Updated Survey," in *Algorithm Design for Computer System Design*, Eds., G. Ausiello, M. Lucertini and P. Serafini, Heidelberg: Springer, 49-106.

DAY, H. E., and EDELSBRUNNER, H. (1984), "Efficient Algorithms for Agglomerative Hierarchical Clustering Methods," *Journal of Classification, 1*, 7-24.

DELATTRE, M., and HANSEN, P. (1980), "Bicriterion Cluster Analysis," *IEEE Transactions on Pattern Analysis and Machine Intelligence, PAMI-2(4)*, 277-291.

DIDAY, E. et al. (1979), *Optimisation en classification automatique*, Le Chesnay: INRIA.

FLORY, A., GUNTHER, J., and KOULOUMDJIAN, J. (1978), "Database Reorganization by Clustering Methods," *Information Systems, 3*, 59-62.

GAREY, M. R., and JOHNSON, D. S. (1979), *Computers and Intractability: A Guide to the Theory of NP-Completeness*, San Francisco: Freeman.

GORDON, A. D. (1981), *Classification: Methods for the Exploratory Analysis of Multivariate Data*, New York: Chapman and Hall.

GOWER, J. C., and ROSS, G. J. S. (1969), "Minimum Spanning Trees and Single-linkage Cluster Analysis," *Applied Statistics, 18*, 54-64.

HANSEN, P., JAUMARD, B., and FRANK, O. (1989), "Maximum Sum-of-Splits Clustering," *Journal of Classification, 6*, 177-193.

HARTIGAN, J. A. (1975), *Clustering Algorithms*, New York: Wiley.

HSU, W-L., and NEMHAUSER, G. L. (1979), "Easy and Hard Bottleneck Location Problems," *Discrete Applied Mathematics, 1*, 209-215.

HUBERT, L. (1977), "Data Analysis Implications of Some Concepts Related to the Cuts of a Graph", *Journal of Mathematical Psychology, 15*, 199-208.

JOHNSON, D. S. (1974), "Fast Algorithms for Bin Packing," *Journal of Computers and Systems Sciences, 8*, 272-274.

KING, J. R., and NAKORNCHAI, V. (1982), "Machine-component Group Formation in Group Technology," *International Journal of Production Research, 20(2)*, 117-133.

KNUTH, D. E. (1973), *The Art of Computer Programming, Volume 3: Sorting and Searching*, Reading, Massachusetts: Addison-Wesley.

KUMAR, R. K., KUSIAK A., and VANNELLI, A. (1986), "Grouping of Parts and Components in Flexible Manufacturing Systems," *European Journal of Operational Research, 24*, 387-397.

KUSIAK, A., VANNELLI, A., and KUMAR, R. K. (1986), "Clustering Analysis: Models and Algorithms," *Control and Cybernetics, 15(2)*, 139-153.

LECLERC, B. (1977), "An Application of Combinatorial Theory to Hierarchical Classification", in *Recent Developments in Statistics*, Eds., J. R. Barra, F. Brodeau, G. Romier, and B. van Cutsem, Amsterdam: North Holland, 783-786.

MARTELLO S., and TOTH P. (1990), "Lower Bounds and Reduction Procedures for the Bin-packing Problem," *Discrete Applied Mathematics* (forthcoming).

MARTELLO S., and TOTH P. (1989), *Knapsack Problem Algorithms and Computer Implementation*, Wiley: New York (to appear).

MATIASEVITCH, Y. Y. (1973) "Enumerable Sets Are Diophantine," *Soviet Mathematics Doklady, 11*, 354-357.

MURTAGH, F. (1985), "A Survey of Algorithms for Contiguity-Constrained Clustering and Related Problems," *The Computer Journal, 28*, 82-88.

PRIM, R. C. (1957), "Shortest Connection Networks and Some Generalizations," *Bell System Technical Journal, 36*, 1389-1401.

RAO, M. R. (1971), "Cluster Analysis and Mathematical Programming," *Journal of the American Statistical Association, 66*, 622-626.

ROSENSTIEHL, P. (1967), "L'arbre minimum d'un graphe," in *Théorie des Graphes, Rome, I.C.C.*, Ed., P. Rosenstiehl, Paris: Dunod, 357-368.

SPÄTH, H. (1980), *Cluster Analysis Algorithms for Data Reduction and Classification of Objects*, Chichester: Horwood.

VINOD, H. (1969), "Integer Programming and the Theory of Groups," *Journal of the American Statistical Association, 64*, 506-519.

WAGHODEKAR, P. H. and SAHU S. (1983), "Group Technology: A Research Bibliography," *OPSEARCH, 20(4)*, 225-249.

ZAHN, C. T. (1971), "Graph-theoretical Methods for Detecting and Describing Gestalt Clusters," *IEEE Transactions on Computers, C-20*, 68-86.