Journal of Heuristics, 9: 51–73, 2003
© 2003 Kluwer Academic Publishers. Manufactured in The Netherlands.

Heuristic Methods for Large Centroid Clustering Problems*

ÉRIC D. TAILLARD
EIVD, University of Applied Sciences of Western Switzerland, Route de Cheseaux 1, CasePostale, CH1401 Yverdon-les-Bains, Switzerland
email: Eric.Taillard@eivd.ch; http://www.ina.eivd.ch/taillard

Abstract

This article presents new heuristic methods for solving a class of hard centroid clustering problems including the p-median, the sum-of-squares clustering and the multi-source Weber problems. Centroid clustering is to partition a set of entities into a given number of subsets and to find the location of a centre for each subset in such a way that a dissimilarity measure between the entities and the centres is minimized. The first method proposed is a candidate list search that produces good solutions in a short amount of time if the number of centres in the problem is not too large. The second method is a general local optimization approach that finds very good solutions. The third method is designed for problems with a large number of centres; it decomposes the problem into subproblems that are solved independently. Numerical results show that these methods are efficient—dozens of best solutions known to problem instances of the literature have been improved—and fast, handling problem instances with more than 85,000 entities and 15,000 centres—much larger than those solved in the literature. The expected complexity of these new procedures is discussed and shown to be comparable to that of an existing method which is known to be very fast.

Key Words: clustering, location-allocation, p-median, sum-of-squares clustering, multi-source Weber problem

1. Introduction

Cluster analysis is to partition a set of entities into subsets, or clusters, such that the subsets are homogeneous and separated one another, considering measurements describing the entities. In this paper, we propose new efficient methods for centroid clustering problems. More precisely, we are going to apply our methods to problems of the following type: given n entities e_i with weights $w_i (i = 1, \ldots, n)$ it is searched p centres $c_j (j = 1, \ldots, p)$ minimizing $f(c_1, \ldots, c_p) = \sum_{i=1}^{n} \min_j w_i d(e_i, c_j)$, where $d(e_i, c_j)$ measures the dissimilarity between e_i and c_j. However, the methods are very general and may be applied to other problems or objective functions.

If the entities are described by their co-ordinates in $\mathbb{R}^m$, $d(e_i, c_j)$ is typically the distance or the square of the distance between e_i and c_j. In the last case, the problem is the well known sum-of-squares clustering (SSC) (see e.g. Ward (1963), Edwards and Cavalli-Sforza (1965), Jancey (1966), and MacQueen (1967)). There are many commercial softwares that implement approximation procedures for this hard problem. For instance, the popular

*A former version of the article was entitled "Heuristic methods for large multi-source Weber problems."

S-Plus statistical analysis software incorporates the k-means iterative relocation algorithm of Hartigan (1975) to try to improve the quality of given clusters. For exact algorithms for SSC, see e.g. Koontz, Narendra, and Fukunaga (1975) and Diehr (1985).

In case: (1) the space is $\mathbb{R}^2$, i.e. the Euclidean plane, (2) the centres can be placed everywhere in $\mathbb{R}^2$ and (3) the dissimilarity measure is the Euclidean distance, the problem is called the multi-source Weber problem (MWP). This problem occurs in many practical applications, such as the placement of warehouses, emitter antennas, public facilities, airports, emergency services, etc. See e.g. Saaty (1972), Dokmeci (1977), Fleischmann and Paraschis (1988), Bhaskaran (1992), and Lentrek, MacPerson, and Phillips (1993) that describe practical applications that need to solve MWPs with up to more than 1700 entities and 160 centres. For exact methods solving the MWP, see e.g. Rosing (1992) and Krau (1997). For a unified comparison of numerous approximation algorithms, see Brimberg et al. (2000).

In case dissimilarities between entities are given by an arbitrary $n \times n$ matrix and the centres can be placed on the entities only, the problem is called the p-median problem (PMP). The last is a well-known NP-hard problem, see e.g. Hakimi (1965), ReVelle and Swain (1970), Mirchandani and Francis (1990), and Daskin (1995). For exact methods solving the PMP, see e.g. Erlenkotter (1978), Rosing, ReVelle, and Rosing-Vogelaar (1979), Beasley (1985), and Hanjoul and Peeters (1985). For an introduction to location theory and clustering, see also Gordon (1981), Späth (1985), and Wesolowsky (1993).

The new methods presented in this paper, candidate list search (CLS), local optimization (LOPT) and decomposition/recombination (DEC), have been successfully applied to SSC, MWP and PMP, but they can be extended to solve other problems. For example, the CLS and LOPT methods can be applied to any location-allocation problems as soon as two appropriate procedures are available: the first one for allocating entities to centres and the second one for optimally locating a centre, given the entities allocated to it. For SSC, MWP or PMP, the allocation procedure simply consists in finding the nearest centre to each entity. For other problems, this procedure must be more elaborated (e.g. if there is a constraint limiting the sum of the weights of the entities allocated to a centre).

The LOPT method proceeds by local optimization of subproblems. This is a general optimization method that can be applied to problems not directly related to clustering (see e.g. Taillard and Voß (2002), where LOPT is presented under the name of *POPMUSIC*).

In order to remain relatively concise, we are going to present applications of our methods for PMP, SSC and MWP only, but with a special attention to the under studied MWP. Indeed, while the MWP by itself does not embrace all of the problem features found in some practical applications, this model can be very useful, especially for real applications dealing with many thousands of entities. In figure 1, we show the decomposition into 23 clusters of a very irregular problem built on real data, involving 2863 cities of Switzerland. The large black disks are the centres while the small disks are the cities (or entities). Cities allocated to the same centres have the same colour. In this figure, we have also added the federal frontiers and the lakes. Politically, Switzerland is composed of 23 states (Cantons); physically, it is composed of extremely thickly populated regions (Plateau) and regions without cities (Alps, lakes). We see in figure 1 that the positions of the centres are sensible (no centres are located outside Switzerland or on a mountain or in a lake) and that the decomposition

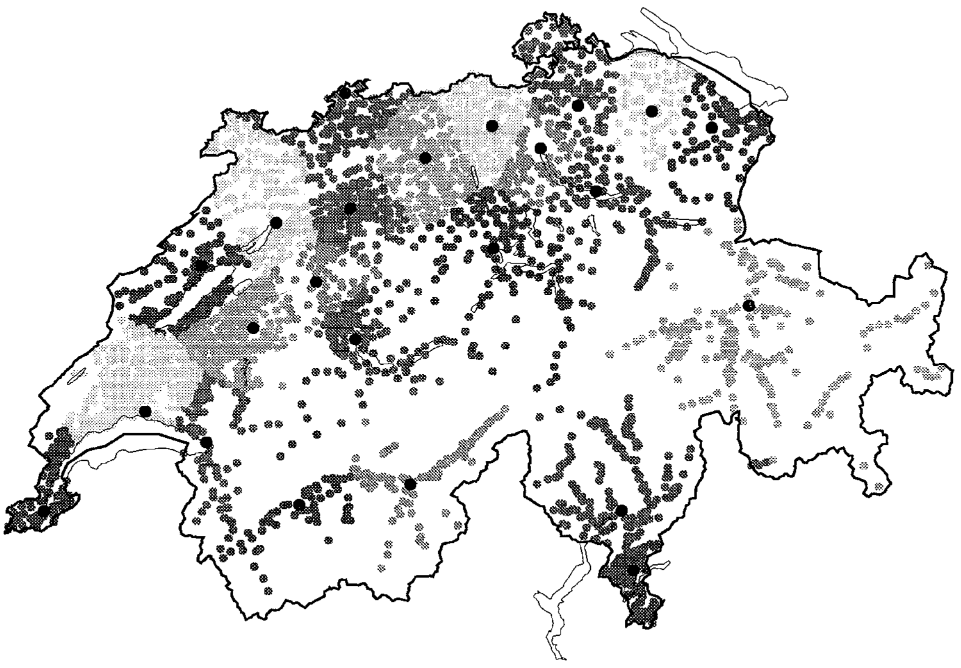

Figure 1. Decomposition of Switzerland into 23 clusters by solving a multi-source Weber problem.

generally respects the natural barriers (spaces without cities); there are very few entities that are separated from their centre by a chain of mountains.[1] Moreover, if the solution of figure 1 is compared to the solution obtained by solving a PMP with dissimilarity measure being the true shortest paths (the road network having more than 30000 connections), it can be shown that the PMP solution is very similar to the MWP one (21 centres are placed almost at the same position; the main difference is that there are less entities allocated to a centre located on the other border of a lake). However, solving this PMP is time consuming: the computation of the shortest paths matrix took 100 times longer than finding a very good MWP solution. Therefore, solving an MWP in a first phase before attacking the true problem (as exemplified by a PMP or a multi-depot vehicle routing problem) can be pertinent, even with an irregular, real problem.

Since the clustering problems treated in this paper are difficult, they can be solved exactly for instances of moderate size only. For solving larger instances, as often arise in practice (see the 6800 entities, 380000 network nodes instance of Hikada and Okano (1997)), it is appropriate to use heuristic methods. However, most of the methods of the literature present the same disadvantage of a large increase of the computing time as the number of centres increases and, simultaneously, a decrease in the quality of the solutions produced. The aim of this paper is to show that it is possible to partition a problem with a large number of centres into subproblems that are much smaller, in order to benefit from the advantages of the existing methods for small problems while rapidly producing solutions of good quality to the original problem.

The article is structured as follows: in Section 2, we present in detail the alternate location-allocation (ALT) procedure used as a subprocedure of our candidate list search (CLS), showing how it can be implemented efficiently. ALT was first proposed by Cooper (1963) for the MWP. However, it can be generalised for any location-allocation problem as soon as a location procedure and an allocation procedure are available. In this section, we also present CLS, our basic procedure for solving the subproblems generated by partition methods. In Section 3, we present two partition methods for large problems. The first one, LOPT, can be viewed either as a generalization of the ALT procedure or as a restricted CLS for the post-optimization of a given solution. The second decomposition method, DEC, splits a large problem into independent subproblems and the solutions of these subproblems are optimally mixed together to create a solution to the original problem. Section 4 analyses the computational performances of the methods proposed.

2. Basic procedures *ALT* and *CLS*

The procedures ALT and CLS are used as subprocedures in the decomposition methods we propose. Referring to the paper of Cooper (1963) is not sufficient to understand the procedure ALT well, since certain details of this algorithm are not discussed in the original paper and the choices made for implementing the procedure can have a profound impact on its effectiveness. Moreover, we have adapted this procedure to accelerate its execution.

2.1. Generalized ALT procedure

The iterative location-allocation procedure of Cooper (1963) may be sketched as shown in Algorithm 1. Cooper has designed this algorithm for the MWP. In this case, the location procedure can be implemented using a procedure like those of Weiszfeld (1937). For the SSC, the centre of gravity of the entities is the optimum location of the centre. For the PMP, the optimum location of a centre can be obtained by enumerating all possible location for the centre. The allocation procedure is very simple for SSC, PMP and MWP: each entity is allocated to its nearest centre. For other problems, this procedure can be more difficult to implement.

Two steps of this algorithm have to be discussed: the choice of the initial solution at step 1, and the repositioning of centres that are not used at step 2a. For the choice of an initial solution, many variants have been tested:

```
0) Input: Set of entities with weight and dissimilarity measure,
   problem specific allocation and location procedures.
1) Choose an initial position for each centre.
2) Repeat the following steps while the location of the centres varies:
   2a) Allocate the entities given the centre locations.
   2b) Given the allocation made at step 2a, locate each centre optimally.
```

Algorithm 1. Locate-allocate procedure of Cooper (1963) (ALT).

(1) Position the centres on p randomly elected entities; the probability of choosing an entity being proportional to its weight.
(2) Choose the position of the centres one by one, by trying to position them on an entity and by electing the position that minimizes the objective function.

The first variant takes into account the structure of the problem, i.e. the geographical and weighting spread of the entities. It produces relatively good initial solutions, especially for problems with non uniform weights.

The second variant induces the ALT procedure to produce the best solutions on the average but its computing time is high: for each of the p centres, $O(n)$ positions have to be tried, and for each of these positions, one has to verify whether each entity is serviced by the new position. This implies a procedure that operates in $O(n^2 \cdot p)$ time, while the other variant can be done much faster. To reduce the complexity of this variant and to make it non deterministic,[2] we adopt the following $O(n \cdot p)$ greedy procedure (Algorithm 2) in the spirit of those of Dyer and Frieze (1985):

After having repositioned the centres at step 2b of the ALT procedure, it may happen that the allocation of the next iteration, at step 2a, does not use all the centres. The unused centres can be relocated to improve the current solution. We have adopted the following policy:

Determine the centre that contributes most to the objective function and place an unused centre on its most distant entity; re-allocate the entities and repeat this as long as unused centres exist.

Starting with a very bad initial solution ($O(p)$ centres that are not used), this re-location policy could lead to a $O(p^2 \cdot n)$ procedure. However, our initial solution generator (as well as our CLS procedure presented below) furnish solutions to the ALT procedure that contain an unused centre only exceptionally (for the MWP, we have observed somewhat less than one occurrence in 1000, even for a large number of centres). So, the re-location policy has almost no influence on the solution quality, if one starts with a "good" initial solution as we do. Mladenovic and Brimberg (1996) have shown that the re-location policy can have a substantial effect on MWP solution quality if one starts with "bad" initial solutions.

```
0) Input: Set of entities with weight and dissimilarity measure.
1) Choose an entity at random and place a centre on this entity.
2) Allocate all entities to this centre and compute their weighted
   dissimilarities.
3) For $k$ = 2 to $p$ do:
   3a) Find the entity that is the farthest from a centre (weighted
       dissimilarities) and place the $k$th centre at that entity's
       location.
   3b) For $i$ = 1 to $n$ do, if entity $i$ is allocated to a centre farther
       than centre $k$:
       Allocate entity $i$ to centre $k$ and update its weighted
       dissimilarity.
```

Algorithm 2. Initial solution generator.

2.1.1. Complexity of ALT for PMP, SSC and MWP. First, let us introduce a new complexity notation: In the remaining of the paper, let $\hat{O}(.)$ denote an empirically estimated complexity, while $O(.)$ denotes the standard worst case complexity. For example, both *quick sort* and *bubble sort* algorithms operate in $O(n^2)$ time. In practice however, it is observed that *quick sort* has an $\hat{O}(n \cdot \log(n))$ behaviour while *bubble sort* has an $\hat{O}(n^2)$ behaviour (Rapin, 1983).[3] There are also algorithms for which the theoretical worst case complexity is not established. However, observing the average computing times by executing an algorithm on many instances can provide a good idea of its complexity in practice. The advantage of this notation is to make a distinction between practice and theory. Indeed, it is common to read that the complexity of *quick sort* is $O(n \cdot \log(n))$, which is not true, formally. Moreover, the "ˆ" notation is often used by statisticians for estimated values.

The complexity of the ALT procedure can be estimated as follows. The complexity of Step 2a (allocation of the entities to a centre) is $O(p \cdot n)$. Indeed, for the problems under consideration one has to allocate each entity to its nearest centre. For large values of p, this step can be substantially accelerated by observing that only the centres that have moved from one iteration to the next can modify the allocation previously made. (Compares the computing times of old and new ALT implementations in Table 3.)

Step 2b can be performed in $O(n)$ for the SSC. Indeed, each entity contributes only once in the computation of the position of each centre (independently from the number of centres). For the MWP, the optimum location can be found with a Weiszfeld-like procedure (1937) that repeats an unknown number of gradient steps. We have arbitrarily limited this number to 30. So, in our implementation, Step 2b has a complexity of $O(n)$. For small values of p, the computing time of this step dominates. For the PMP, let us suppose that $O(n/p)$ entities are allocated to each centre (this is reasonable if the problem is relatively regular).[4] For each centre, one has to scan $O(n/p)$ possible locations and the evaluation of one position can be performed in $O(n/p)$. So, the total complexity of Step 2b is $O(p \cdot n/p \cdot n/p) = O(n^2/p)$ for locating the p centres.

Since p is bounded by n, the global complexity of steps 2a and 2b is bounded by $O(n^2)$ for SSC, MWP and PMP. Now, we have to estimate the number of repetitions of Loop 2 which is unknown. However, in practice, we have observed that the number of iterations seems to be polynomial in n and p. Therefore, we will use an $\hat{O}(p^\alpha \cdot n^\beta)$ estimation of the overall complexity of our implementation of the ALT procedure. In this study, we are mostly interested in instances with large values of p, so, we have considered instances with $n/5 \leq p \leq n/3$ for evaluating the α and β values for the various clustering problems. For the SSC and MWP, we have considered about 7000 instances uniformly generated with up to 9400 entities. For the PMP, we have considered about 38000 runs of the ALT procedure. The PMP instances were based on the 40 different distance matrices proposed by Beasley (1985). The number of entities for these instances ranges from 100 to 900.

For the SSC, we have estimated $\alpha \cong 0.83$ and $\beta \cong 1.19$; for the PMP the estimation is $\alpha \cong 0.70$ and $\beta \cong 1.23$ and for the MWP $\alpha \cong 0.85$ and $\beta \cong 1.34$. So, if p grows linearly with n, the estimated complexity of the ALT procedure is not far from $\hat{O}(n^2)$ for all these problem types. The memory requirement is $O(n)$ for the SSC and MWP and $O(n^2)$ for the PMP i.e. equivalent to the data size.

Table 1. Quality of the greedy procedure for Beasley's PMP instances (% above optimum).

n	100	200	300	400	500	600	700	800	900
p									
5	0	0	0	0.03	0	0	0.00	0.06	0.12
10	0.04	0.08	0	0.14	0.25	0	0.02	0.13	0
$n/10$	0.04	0.00	0.01	0.10	0.10	0.05	0.01	0.07	0.06
$n/5$	0.04	0.15	0.04	0.05	0.05	0.04	0.06	–	–
$n/3$	0.04	0.05	0.06	0.09	0.14	0.14	–	–	–

2.2. *Candidate list search (CLS)*

CLS is based on a greedy procedure that randomly perturbs a solution that is locally optimal according to the ALT procedure. Then, ALT is applied to the perturbed solution and the resulting solution is accepted only if it is better than the initial one, otherwise one returns to the initial solution. The perturbation of a solution consists in eliminating a centre and in adding another one, located on an entity. The process can be repeated until all pairs entities/centres have been scanned. This greedy procedure finds very good solutions: In Table 1, we report the quality of the solutions found when applied to the 40 PMP instances of Beasley (1985). These instances have been solved exactly and the quality of a solution is given in percent above the optimum value. The greedy procedure was executed 20 times for each instance. For 8 instances each run found the global optimum and all instances but one were optimally solved at least once.

For MWP instances with 50 (respectively 287) entities we observed that the greedy procedure finds a global optimum in more than 60% (respectively 40%) of the cases. For the SSC, we succeeded in improving all the best solutions known to 16 instances with 1060 entities and 10 to 160 centres (See Table 4).

For the p-median problem, this type of perturbation has been used for a long time (c.f. Goodchild and Noronha, 1983; Whitaker, 1983; Glover, 1990; Voß, 1996; Rolland, Schilling, and Current, 1997); in this case Glover proposes an efficient way to evaluate the cost of eliminating a centre: during the allocation phase, the second closest centre is memorized—this can be done without increasing the complexity. However, evaluating the decrease of the cost due to the opening of a centre on an entity takes a time proportional to n. Therefore, finding the best possible perturbation has a complexity of $O(n^2 \cdot p)$, without considering the application of the ALT procedure.

This complexity is too high for large instances thus we make use of a candidate list strategy scheme (Glover, 1990) for implementing a probabilistic perturbation mechanism. The idea is to identity the centre to close by a non deterministic but systematic approach. The entity associated with an open centre is also randomly chosen, but its weighted distance from its previously allocated centre must be higher than the average. The process is repeated

```
1) Input: initial solution $\boldsymbol{s_1}$ (location of the $p$ centres), parameter $q$.
2) Generate $\pi$, a random permutation of the elements {1,...,$p$} and $\mu$,
   a random permutation of the elements {1,...,$n$}. Set $i=0$, $j=0$.
3) For $k=1$ to $q$ repeat:
   3a) $i=(i$ modulo $n)+1$; $j=(j$ modulo $p)+1$.
   3b) Compute $d_{max}$, the distance of the most (weighted) distant entity.
   3c) While the weighted distance from entity $\mu_i$ to the nearest centre
       is lower than $(d_{max} - f(\boldsymbol{s_k})/n)/k + f(\boldsymbol{s_k})/n$ do:
       $i=(i=$ modulo $n)+1$.
   3d) Close centre $\pi_j$ and open a new one located at entity $\mu_i$ to obtain
       a perturbed solution $\boldsymbol{s'_k}$
   3e) Improve $\boldsymbol{s'_k}$ with ALT to obtain $\boldsymbol{s''_k}$
   3f) If $f(\boldsymbol{s''_k}) < f(\boldsymbol{s_k})$ then $\boldsymbol{s_{k+1}} = \boldsymbol{s''_k}$, else $\boldsymbol{s_{k+1}} = \boldsymbol{s_k}$.
   3g) If $j=p$, generate a new random permutation $\pi$.
```

Algorithm 3. Candidate list search (CLS).

for a number q of iterations, specified by the user. Algorithm 3 presents CLS into details. CLS starts from an initial solution s_1 that is tentatively improved. To examine the neighbourhood in a non-deterministic way, two permutations π and μ are first generated (step 2). These permutations determine the order in which the perturbations are tried at step 3d. Then, q iterations are repeated, where q is the only parameter of CLS. An iteration consists of moving a centre on an entity whose weighted distance from its previously allocated centre is higher than the average. To belong to the candidate list of entities where a centre can be moved, the distance criterion decreases as the number of iterations grows (step 3c). Finally, the perturbed solution is improved by means of ALT (step 3e) and it is retained only if it improves the solution (step 3f).

The most time consuming part of this algorithm is step 3e, i.e. the application of the ALT procedure to the perturbed solution. As seen above, we can estimate the complexity of this step as $\hat{O}(p^\alpha \cdot n^\beta)$. Therefore, the complexity of CLS is $\hat{O}(q \cdot p^\alpha \cdot n^\beta)$. From now on, we write CLS(q) the improvement of a given solution with q iteration of the CLS procedure.

3. Decomposition methods

In this section, we propose two decomposition methods for solving problems with a large number p of centres. The complexity of these methods is not higher than the ALT procedure while producing solutions of much higher quality. The first decomposition technique, LOPT, starts with any solution with p centres and improves it by considering a series of subproblems involving $r < p$ centres and the entities allocated to them. The subproblems are solved by our CLS algorithm. This method can be viewed as a local search defined on a very large neighbourhood involving up to r centres re-locations at a time. Another point of view is to consider this procedure as a generalization of ALT. Indeed, a solution produced by ALT is locally optimal if we consider any subset of entities allocated to a single centre: every entity is serviced by the nearest centre, and the centres are optimally positioned for the subset of entities they are servicing. Our procedure produces a solution that is sub-optimal (since the subproblems are solved in a heuristic way and since we do not consider all subsets of

r centres) for subsets of entities allocated to r centres. A third point of view is to consider LOPT as a CLS procedure with a much smaller list of candidate moves regarding to the CLS presented above.

The second decomposition method, DEC, partitions the problem into t smaller subproblems. These subproblems are then solved with our CLS for various numbers of centres. A solution to the initial problem is then found by combining solutions of the subproblems. To decompose the initial problem, we solve an intermediate problem with t centres with our CLS procedure. Each set of entities allocated to a centre of the intermediate problem is considered as an independent subproblem.

3.1. Local optimization (LOPT)

The basic idea of LOPT is to select a centre, a few of its closest centres and the set of entities allocated to them to create a subproblem. We try to improve the solution of this subproblem with CLS. If an improved solution is found, then all the selected centres are inserted in a candidate list C, otherwise the first centre used for creating the subproblem is removed from C. Initially, all the centres are in C and the process stops when C is empty. LOPT has two parameters: r, the number of centres of the subproblems and s, the number of iterations of each call to CLS. Algorithm 4 presents more formally the LOPT method.

3.1.1. Complexity of LOPT. To estimate the complexity of LOPT we make two assumptions. First we assume that $O(n/p)$ entities are allocated to each centre (this hypothesis is reasonable if the problem instance is relatively uniform) and second that loop 3 is repeated $\hat{O}(p^\gamma \cdot n^\lambda)$ times. Empirically, we have observed that γ is less than 1 and λ is close to 0 (see Table 8); we estimate that the value of γ is about 0.9 and λ is about 0.2 for the LOPT parameters we have chosen and for the MWP). Then, the complexity of LOPT can be established as follows:

Steps 3a and 3d have a complexity of $O(p)$; step 3b has a complexity of $O(r \cdot p)$; step 3c solves a problem with r centres and $O(r \cdot n/p)$ entities, this leads to a complexity of $\hat{O}(s \cdot r^\alpha \cdot (r \cdot n/p)^\beta)$. This leads to a total complexity of $\hat{O}(r \cdot p^{\gamma+1} \cdot n^\lambda + s \cdot r^{\alpha+\beta} \cdot p^{\gamma-\beta} \cdot n^{\lambda+\beta})$. If r and s are fixed and if p grows linearly with n, the complexity of the LOPT procedure is therefore $\hat{O}(n^{\lambda+\gamma+1})$. This complexity seems to be similar to that of the ALT

```
1) Input: initial position of the $p$ centres, parameters $r$ and $s$.
2) Set $C = \{1,...,p\}$
3) While $C \neq \emptyset$, repeat the following steps:
   3a) Randomly select a centre $i \in C$.
   3b) Let $R$ be the subset of the $r$ closest centres to $i$ ($i \in R$).
   3c) Consider the subproblem constructed with the entities allocated to
       the centres of $R$ and optimize this subproblem with $r$ centres with
       CLS ($s$).
   3d) If no improved solution has been found at step 3c, set $C = C\backslash\{i\}$,
       else set $C = C \cup R$.
```

Algorithm 4. Local optimization procedure (LOPT).

procedure. In practice, step 3c of the LOPT procedure takes most of the computing time, even if steps 3b has a higher expected complexity for extremely large p. Indeed, for fixed n, we have always observed that the computing time diminishes as p increases, even for p larger than 10000 (see Tables 4 to 8). From now on, we denote by LOPT(r, s) the version of the LOPT procedure using parameters r and s. The memory requirement of the LOPT procedure is $O(n)$.

3.2. Decomposition algorithm (DEC)

LOPT optimizes the position of a given number of centres dynamically, but it is also possible to proceed to a static decomposition of the entities, and solve these subproblems with a variable number of centres. A solution to the complete problem may be found by choosing the right number of centres for each subproblem. Naturally, the total number of centres must be limited to p. This re-composition may be performed efficiently and optimally with dynamic programming.

The crucial phase of the algorithm is the first decomposition: if the subproblems created do not have the right structure, it is impossible to obtain a good solution at the end. The more irregular the problem is (i.e. where the entities are not uniformly distributed, or their weights differ widely), the more delicate its decomposition is. For partitioning the problem, we use our CLS procedure applied to the same set of entities but with a number $t < p$ of centres.

The subproblems created may have very different sizes: a subproblem may consist of just a few entities with very high weights or it may comprise a large number of close entities. Thus it could be difficult to evaluate the number of centres to be assigned to a subproblem. Let n_i $(i = 1, \ldots, t)$ be the number of entities of subproblem i. Suppose that subproblem i is solved with $j \in J_i = \{1, \ldots, n_i\}$ centres and let f_{ij} be the value of the objective function when solving subproblem i with j centres. To build a solution to the initial problem, we have to find $j_1^*, \ldots, j_t^*$ minimizing:

$$\begin{aligned} &\underset{j_i \in J_i, i=1,\ldots,t}{\text{minimize}} \sum_{i=1} f_{ij_i} \\ &\text{such that} \quad \sum_{i=1}^{t} j_i \leq p \end{aligned}$$

This problem is a kind of knapsack and may be reformulated as:

$$\underset{j_1 \in J_1}{\text{minimize}} \left\{ f_{1j_1} + \left(\begin{array}{l} \underset{j_i \in J_i, i=2,\ldots,t}{\text{minimize}} \displaystyle\sum_{i=2}^{t} f_{ij_i} \\ \text{such that} \quad \displaystyle\sum_{i=2}^{t} j_i \leq p - j_1 \end{array} \right) \right\}$$

Thus, the problem can be decomposed and solved recursively by dynamic programming in $O(t \cdot p)$ time. This procedure can also produce all the solutions with $t, t+1, \ldots, n$ centres in $O(t \cdot n)$ time. Such a feature can be very useful when we want to solve a problem for which the number of centres is unknown and must be determined, as for example when there is an opening cost for each centre (the opening cost has just to be added in the f_{ij} values).

However, solving each subproblem with $1, \dots, n_i$ centres is time consuming. If the problem is relatively uniform, one can expect that the optimum number of centres found by dynamic programming is not far from p/t for all subproblems. So, we propose to first solve the subproblems for only three different numbers of centres: $\lfloor p/t - 1 \rfloor$, $\lfloor p/t \rfloor$ and $\lfloor p/t + 1 \rfloor$. These are solved with one less (respectively one more) centre when the optimum number of centres determined by dynamic programming is exactly the lower (respectively higher) number for which a solution was computed. Algorithm 5 presents our DEC procedure in details.

3.2.1. Complexity of DEC. For analysing the complexity of DEC, we make the following assumptions: First, each subproblem has $O(n/t)$ entities, second, each subproblem is assigned $O(p/t)$ centres and third, the number of repetitions of loop 5 is a constant (i.e. the total number of subproblems solved with CLS in steps 3 and 5a is in $O(t)$). These assumptions are empirically verified if the problem instances are relatively uniform (see Table 8). With these assumptions, the complexity of DEC can be established as follows: Step 1 is in $\hat{O}(u \cdot t^{\alpha} \cdot n^{\beta})$; steps 3 and 5a can be performed in $\hat{O}(v \cdot t^{1-\alpha-\beta} \cdot p^{\alpha} \cdot n^{\beta})$; finally, the complexity of dynamic programming in steps 4 and 5b is $O(t \cdot p)$. The overall complexity of DEC strongly depends on the parameter t. As shown in the next section, the quality of solutions produced by CLS slightly diminishes as the number of centres increases. We therefore seek to reduce the number of centres in the auxiliary problem and in subproblems as much as possible. For this purpose, we have chosen $t = \lfloor \sqrt{p} \rfloor$. The overall complexity of our implementation of DEC is $\hat{O}(u \cdot p^{\alpha/2} \cdot n^{\beta} + v \cdot p^{(1+\alpha-\beta)/2} \cdot n^{\beta} + p^{3/2})$. If u and v are constant and p grows linearly with n, the complexity is $\hat{O}(n^{\beta+\alpha/2})$, assuming $\beta \geq 1$ and $\beta + \alpha/2 \geq 3/2$, i.e. lower than the ALT procedure. The memory requirement is $O(n^{3/2})$. DEC requires more memory than CLS and LOPT, but the increase is not too high and we

```
0) Input: Set of entities with dissimilarity measure.
1) Solve an auxiliary problem with $t$ centres with CLS($u$).
2) The subsets of entities allocated to the same centre form $t$
   independent subproblems. Set $f_{ij} = \infty$, $1 \leq i \leq t$, $1 \leq j \leq n_i$.
3) For each subproblem do: Solve subproblem $i$ with CLS($v$) with
   max (1, $\lfloor p/t \rfloor$ - 1) $\leq j \leq$ min ($n_i$, $\lfloor p/t \rfloor$ + 1) centres and update the
   $f_{ij}$ values associated.
4) Find a collection $j_1^*, \dots, j_t^*$ of optimum number of centres to
   attribute to each sub-problem with dynamic programming.
5) While $K = \{(i, j_i) \mid f_{ij_i} = \infty, j_i = \min(n_i, j_i^* + 1)$ or
   $j_i = \max(1, j_i^* - 1), 1 \leq i \leq t\} \neq \emptyset$, repeat:
   5a) For all $(i, j_i) \in K$, solve subproblem $i$ with $j_i$ centres, using
       CLS($v$) and update the $f_{ij_i}$ associated.
   5b) Find a new collection $j_1^*, \dots, j_t^*$ of optimum number of centres to
       attribute to each subproblem with dynamic programming.
```

Algorithm 5. The decomposition procedure (DEC).

have succeeded in implementing all the algorithms on a personal workstation. From now on, we note DEC(u, v) the use of the DEC procedure with $t = \lfloor\sqrt{p}\rfloor$, and parameters u and v.

4. Numerical results

4.1. Test problems

For the numerical results presented in this section, we consider six sets composed of 654, 1060, 2863, 3038, 14051 and 85900 entities respectively. The 2863 entities set is built on real data: the entities are the cities of Switzerland and the weight of each city is the number of inhabitants. This set is denoted CH2863.

The other sets correspond to the travelling salesman problems that can be found under the names of P654, U1060, Pcb3038 Brd14051 and P1a85900 in the TSPLIB compiled by Reinelt (1995). For these sets, all entities are weighted to one and the dissimilarity between two entities is the Euclidean distance (for PMP and MWP) or the square of the Euclidean distance (for the SSC). From these six sets of entities, we have constructed a large collection of instances by varying p. In Table 2, we give the values of p we have considered for each set, and the best MWP solution known associated with each p.

All the best solutions known have been found during the elaboration of methods presented in this paper; some have been reported earlier in Hansen, Mladenovic, and Taillard (1998) or in Brimberg et al. (2000) for P654 and U1060. For P654, we were able to find the same best solution values reported by Brimberg et al. for $p \leq 60$, and to find better values for $p > 60$; for U1060, we succeeded in improving all the best solution values with the exception of $p = 10$ where we got the same value. The best solutions published in Brimberg et al. were obtained by considering more than 20 different methods, and running each of them 10 times. This last reference also reports the optimum solution values of smaller problem instances with 50 and 287 entities. We were able to find all these optimum solution values with our CLS method. So, we conjecture that many of the solution values given in Table 2 for the smallest set of entities are optimal. For the larger sets, we think that small improvements can be obtained.

The aim of Table 2 is to provide new MWP instances and to assert the absolute quality of our methods: Indeed we think that providing the relative quality (measured in per cent over the best solution value of Table 2) allows comparisons to be made more easily than providing absolute solution values. Sometimes, the best solutions known have been found by using sets of parameters for which results are not reported in this paper and it would be difficult to estimate the effort needed to obtain each best solution known. Consequently, we do not provide computing times in this table.

Our algorithms are implemented in C++ and run on a Silicon Graphics (SG) 195 MHz workstation with R10000 processor. In order to make fair comparisons with algorithms implemented by other authors and executed on a different machine, we have sometimes used another computer, clearly indicated in the tables that follow. It was not possible to report exhaustive numerical results due to the large number of problem instances (160), problem types (PMP, MWP or SSC) and methods (CLS, DEC and LOPT). We try to report representative results in a condensed form. However, let us mention that the conclusions

Table 2. Number of centres and best solution values of the MWP instances.

P654		CH2863		Pcb3038		Brd14051		Pla85900	
p	Best known	*p*	Best known	*p*	Best known	*p*	Best known	*p*	Best known
2	815313.30	23	662591164.9	100	351171.14	100	2504969.0	500	980624500
3	551062.88	100	249683701.9	110	333361.33	110	2379966.4	1000	641279543
4	288190.99	110	233112123.2	120	317493.30	120	2273050.9	1500	504307979
5	209068.79	120	219389699.5	130	303337.42	130	2181810.9	2000	430487424
6	180488.21	130	207402173.2	140	291019.41	140	2098583.0	2500	379898116
7	163704.17	140	196628487.0	150	279724.73	150	2021251.0	3000	342898337
8	147050.79	150	187152713.6	160	269670.44	160	1952950.7	4000	290679280
9	130936.12	160	178764596.3	170	260281.77	170	1890823.0	5000	254125361
10	115339.03	170	171147676.8	180	251595.67	180	1835182.6	6000	228045203
11	100133.20	180	164263718.8	190	243642.56	190	1784816.3	7000	207638816
12	94152.055	190	157733395.3	200	236294.30	200	1736960.1	8000	191874305
13	89454.761	200	151650126.1	250	206527.62	250	1547174.7	9000	177990886
14	84807.669	250	126652250.1	300	184832.94	300	1404028.4	10000	166535699
15	80177.042	300	107783856.7	350	168324.66	350	1293453.6	15000	130395710
20	63389.024	350	92871653.41	400	154657.34	400	1203580.1		
25	52209.511	400	81459860.80	450	143330.07	450	1129413.2		
30	44705.192	450	72088804.48	500	133590.94	500	1066429.9		
35	39257.268	500	64277122.05	600	117716.57	600	966474.12		
40	35704.408	600	51713134.53	700	104627.14	700	887518.23		
50	29338.011	700	41923352.49	800	94301.618	800	824127.49		
60	24504.395	800	34171537.06	900	85704.652	900	771207.89		
70	21465.436	900	28112316.53	1000	78458.720	1000	725300.72		
80	19193.861	1000	23063201.81			1500	570402.36		
90	17514.423					2000	475580.22		
100	16083.535					2500	409677.92		
110	14826.578					3000	359978.52		
120	13887.739					5000	237511.94		
130	13127.544					10000	67991.511		
140	12396.740								
150	11668.528								
160	11011.508								
170	10379.992								
180	9781.5080								
190	9314.0959								
200	8842.2075								
250	6909.0608								
300	5341.6934								
350	4322.2073								
400	3594.3474								
450	2887.0762								
500	2181.8495								

U1060			
p	Best known	*p*	Best known
5	1851877.266	80	325971.2435
10	1249564.785	85	313446.5796
15	980131.6889	90	302479.0412
20	828685.6547	95	292282.6205
25	721988.1555	100	282536.4434
30	638212.3349	105	273463.3113
35	577496.6286	110	264959.9572
40	529660.1236	115	256763.0089
45	489483.7564	120	249050.4758
50	453109.5682	125	241880.3791
55	422638.6801	130	235203.3867
60	397674.5281	135	228999.8046
65	376630.2949	140	223062.6283
70	357335.1348	145	217462.7950
75	340123.4976	150	212236.2574

we draw for a given method for a problem type are generally valid for another problem type.

4.2. *ALT and CLS*

First, we want to show the efficiency of our CLS algorithm by comparing it to the results produced by one of the best methods at the present time for the MWP: MWPM, an algorithm that first solves exactly a p-median before re-locating optimally the centres in the continuous plane. This method is due to Cooper (1963) but has been forgotten for a long time before Hansen, Mladenovic, and Taillard (1998) show that in fact, it is one of the most robust for small and medium size MWPs (see also Brimberg et al., 2000). We do not consider methods such as those of Bongartz, Calamai, and Conn (1994) which are too slow and produce too poor solutions or those of Chen (1983) or Murtagh and Niwattisyawong (1982) which are not competitive according to Bongartz et al. Also, we do not compare our results with the HACA algorithm of Moreno, Rodrígez, and Jiménez (1990) for two reasons: First the complexity of HACA is $O(p^2n)$ and requires an $O(p^2)$ memory, i.e. $O(n^3)$ in time and $O(n^2)$ in memory if $p = O(n)$, which are clearly higher than those of our methods. Second, HACA produces solutions that are not as good as MWPM. Indeed, HACA first builds a heuristic solution to the p-median instance associated to the MWP and then applies the ALT procedure to the p-median solution. The reader is referred to Brimberg et al. (2000) for a unified comparison of a large range of heuristic methods for the MWP.

To show the effects of the improvements of the ALT procedure proposed in this paper, we provide the best solutions obtained over 100 repetitions of an old version of ALT that starts with different initial solutions; this method is denoted MALT(100). The results for MALT(100) and MWPM originate from Hansen, Mladenovic, and Taillard (1998). In Table 3, we give the solution quality (measured in per cent above the solution value given in Table 2) of MWPM, MALT(100), CLS(100) and CLS(1000) and their respective computing times (seconds on Sun Sparc 10 workstation) for P654. The computing time of CLS(1000) is roughly 10 times that of CLS(100). We have averaged all these results for 10 independent runs of the algorithm. Where the 10 runs of CLS(100) find solutions values identical to those given in Table 2, we provide in brackets the number of iterations required by the worst run of CLS out of 10 to find the best solution known. From this table, we can conclude that:

- The new ALT procedure runs 6 to 9 times faster than the old one (both MALT(100) and CLS(100) call 100 times an ALT procedure, the old one for MALT, the new one for CLS).
- CLS(100) provides much better solutions than MALT(100).
- CLS(1000) provides better solutions than MWPM, in a much shorter computing time.
- As p grows, the solution quality of all the algorithm diminishes.

4.3. *Decomposition methods DEC and LOPT*

As LOPT requires an initial solution in input, we indicate the performances of LOPT when applied to the solution produced by the DEC procedure. In the following tables, the

Table 3. Comparisons of CLS(100) and CLS(1000) with MWPM and MALT(100) for MWP instances P654.

	Quality (% above best known)				Computing time (s. Sun Sparc 10)		
p	MWPM	MALT(100)	CLS(100)	CLS(1000)	MWPM	MALT(100)	CLS(100)
2	0	0	0	[1]	66	69	9.7
3	0.20	0	0	[30]	75	59	8.7
4	0	0	0	[20]	50	66	9.6
5	0	0	0	[1]	67	66	10.4
6	0	0	0	[20]	72	64	7.6
7	0	0	0	[40]	62	75	6.7
8	0	0	0	[50]	79	57	6.1
9	0	0.15	0	[70]	71	55	6.3
10	0.040	1.1	0	[100]	75	54	6.2
11	0.069	6.1	0	[40]	66	52	5.8
12	0	3.6	0	[70]	97	52	5.8
13	0.013	1.4	0.017	0.0028	259	51	6.0
14	0.014	1.9	0.034	0.017	277	50	6.2
15	0.014	2.2	0.11	0.0099	196	49	6.2
20	0.55	5.3	0.16	0.011	1358	49	6.2
25	0.13	9.5	0.75	0.0014	1420	53	6.1
30	0.22	13.6	0.66	0.017	4222	56	6.3
35	0.38	15.3	0.82	0.04	1608	58	6.6
40	0.56	18.0	1.1	0.042	1762	61	7.0
45	0.50	19.9	1.4	0.17	1296	64	7.3
50	0.43	25.6	1.7	0.30	2487	64	7.6

computing times for LOPT, do not take into consideration the computing time of DEC to obtain the initial solution.

Table 4 compares CLS(1000), DEC(20, 50), LOPT(10, 50) and 3 VNS variants (due to Hansen and Mladenovic (1999) for SSC instances built on entities set U1060. This table gives: The best solution value known (found with our methods), the solution quality of the methods (per cent over best known; VNS results originate from Hansen and Mladenovic), and their respective computing times (seconds on Sun Sparc 10 workstation). The computing time of VNS1 and VNS2 is 150 seconds for all instances. VNS3 corresponds to the best over ten executions of VNS2; therefore, its computing time is 1500 seconds. It is shown in Hansen and Mladenovic that all VNS variants are more efficient than other methods of the literature, such as the k-means algorithm of Hartigan (1975). From this table, we can conclude:

– For small values of p, CLS provides better solutions than DEC, DEC + LOPT and VNSs.

Table 4. Comparison of CLS(1000), DEC(20, 50). LOPT(10, 50) and various VNSs for SSC instances U1060.

		Quality (% above best known)						Computing time (s. Sparc 10)		
p	Best solution value known	CLS	DEC	LOPT	VNS1	VNS2	VNS3	CLS	DEC	LOPT
10	1754840214	0.00	16.32	0.00	0.14	0.14	0.04	156.5	9.0	114.2
20	791794596.2	0.00	6.54	0.01	3.52	0.76	0.03	128.1	11.6	67.2
30	481251642.9	0.02	10.28	0.37	10.88	1.08	0.22	120.5	11.1	79.1
40	341342885.9	0.13	4.60	0.36	16.46	1.25	0.44	115.8	12.3	53.5
50	255509536.2	0.33	7.54	0.45	30.65	1.97	0.54	114.8	10.8	44.1
60	197273037.6	0.53	7.99	0.67	36.48	1.55	0.95	117.2	11.9	35.8
70	158450591.9	0.33	6.98	0.17	45.06	1.63	0.84	120.2	12.5	27.6
80	128890171.4	0.74	7.40	0.38	52.43	1.65	0.89	118.6	12.1	25.8
90	110456793.7	1.04	7.45	0.49	46.08	1.52	0.78	122.4	12.0	25.2
100	96330296.40	1.14	7.29	0.44	44.51	2.23	1.06	125.2	12.3	24.7
110	84849661.98	1.26	6.78	0.49	46.41	3.06	1.69	127.4	12.0	22.6
120	75545061.47	1.55	7.10	0.58	40.11	2.14	1.11	131.1	12.1	21.5
130	67561764.35	2.15	7.96	0.69	40.21	1.95	1.27	135.0	12.5	19.5
140	61128895.04	2.28	7.81	0.57	32.13	2.42	0.98	138.0	12.4	20.0
150	55918930.43	2.10	7.01	0.52	27.12	2.72	1.36	142.2	11.9	19.6
160	51310503.02	1.84	7.18	0.48	28.04	2.47	1.75	145.1	11.6	19.8

- For the largest values of p, DEC produces fairly good solutions and their quality seems not to decrease as p increases.
- The solution quality of LOPT is always very good and seems to be somewhat correlated with the initial solution quality (obtained here with DEC).
- Unexpectedly, the computing time of LOPT and DEC diminishes as p increases; this is undoubtedly due to the small number of entities of U1060.
- CLS (for $p \leq 80$) and DEC + LOPT produces better solutions than VNSs in a much lower computation time (DEC + LOPT is up to 50 times faster).

Table 5 show the effect of the parameters of DEC and LOPT by confronting DEC(20, 50), DEC(20, 200), LOPT(10, 50) (starting with the solution obtained by DEC(20, 50)) and LOPT(10, 200) (starting with the solution obtained by DEC(20, 200)). This table provides the solution quality and the computing times (seconds on SG) for the MWP instances CH2863; all the results are averaged over 10 runs. From this table, we can conclude:

- For small values of p, the quality of DEC(20, 200) is slightly better than DEC(20, 50) but the computing times are much higher.
- Starting with solutions of similar quality, LOPT(10, 50) and LOPT(10, 200) produces solutions of similar quality but the computing time of LOPT(10, 200) is much higher.
- For larger values of p, the quality of DEC(20, 50) slightly decreases but the quality of DEC(20, 200) remains almost constant.

Table 5. Quality and computing time of DEC and LOPT for different parameter settings for MWP instance CH2863.

	Quality (%)				Computation time (s. on SG)			
	DEC		LOPT		DEC		LOPT	
p	20, 50	20, 200	10, 50	10, 200	20, 50	20, 200	10, 50	10, 200
100	3.4	3.2	0.28	0.20	30	162	53	197
110	3.3	3.0	0.17	0.09	29	154	49	174
120	3.5	3.0	0.27	0.17	28	160	56	183
130	3.4	3.0	0.15	0.12	30	157	46	160
140	3.4	3.0	0.15	0.17	31	163	46	163
150	3.8	3.5	0.23	0.14	27	145	45	161
160	3.9	3.4	0.17	0.12	26	146	47	166
170	4.4	4.1	0.24	0.14	22	126	43	163
180	4.5	4.1	0.18	0.18	24	134	47	160
190	4.8	4.3	0.34	0.20	24	132	45	157
200	4.9	4.4	0.21	0.25	21	121	46	152
250	5.0	4.4	0.26	0.14	20	117	43	149
300	5.3	4.3	0.47	0.20	18	111	41	136
350	4.9	4.2	0.50	0.30	17	100	38	123
400	5.1	4.0	0.63	0.38	17	103	36	108
450	5.2	3.7	0.94	0.44	17	106	35	110
500	4.8	3.4	1.02	0.39	18	108	35	105
600	5.7	3.4	1.42	0.39	16	104	35	89
700	5.8	3.7	1.48	0.52	19	106	32	82
800	5.7	3.8	1.48	0.44	19	107	33	79
900	6.6	4.0	1.59	0.71	18	108	32	70
1000	7.1	4.7	2.18	1.10	20	108	33	69

– The computing times of DEC and LOPT diminishes as p increases.
– LOPT greatly improves the solution quality obtained by DEC.
– The methods seems to be very robust since they provide good results for instances with a very irregular distribution of dissimilarities.

In Tables 6 and 7, we compare DEC + LOPT to a fast variant of VNS, called RVNS, for SSC and PMP instances built on entities set Pcb3038. RVNS results originate from Hansen and Mladenovic (1999). We have adapted the LOPT parameters in order to get comparable computation times. For all SSC, PMP and MWP instances, we succeeded in improving the best solutions published in this last reference. In Table 6, we can see that DEC + LOPT is able to find better solutions than RVNS in shorter computation times. For the PMP, RVNS seems to be faster than DEC and LOPT for the smallest number of centres. However, let us mention that our implementation derives directly from the MWP one and is not optimized

Table 6. Comparison of DEC(20, 50), LOPT(6, 40) and RVNS for SSC instance Pcb3038.

		Quality (%)			Time (s. Sparc 10)		
p	Best known	RVNS	DEC	LOPT	RVNS	DEC	LOPT
100	47721950.8	2.27	5.24	1.04	152.8	50	82
150	30524769.8	3.13	4.63	1.09	153.0	48	55
200	21885997.1	2.44	5.55	0.90	159.9	48	44
250	16621446.5	2.56	6.98	1.59	182.1	46	39
300	13290304.8	2.50	7.22	1.44	229.3	49	33
350	11027516.8	2.53	7.24	1.42	230.9	44	30
400	9362179.2	3.35	7.56	1.70	165.0	43	27
450	8101618.7	3.47	7.18	1.66	242.6	42	26
500	7102678.4	2.85	7.47	1.73	204.4	43	25

Table 7. Comparison of DEC(20, 50), LOPT(7, 50) and RVNS for PMP instance Pcb3038.

		Quality (%)			Time (s. Sparc 10)		
p	Best known	RVNS	DEC	LOPT	RVNS	DEC	LOPT
100	352704.86	1.12	4.04	0.65	132.4	197	485
150	281193.96	0.65	4.41	0.74	128.5	141	277
200	238432.02	1.23	4.12	0.74	107.6	106	187
250	209241.25	0.71	4.17	0.59	150.3	85	150
300	187723.46	0.52	4.08	0.64	130.6	84	125
350	170973.34	0.83	3.99	0.67	153.1	72	110
400	157030.46	1.13	4.04	0.83	158.7	64	97
450	145422.94	1.13	4.14	0.76	179.5	65	87
500	135467.85	0.89	4.01	0.71	209.7	59	81

for the PMP. For example we do not compute the distances only once at the beginning of the execution and store them in a (very large) matrix. For large number of centres, DEC + LOPT is again faster and better than RVNS.

In Table 8, we provide computational results for our methods DEC(20, 50), DEC(20, 200) and LOPT(7, 50) (applied to the solution obtained with DEC(20, 200)) for MWP instances Brd14051 and Pla85900. We give the following data in this table: The number n of entities, the number p of centres, the solution quality obtained by DEC and LOPT (percent over best known), the respective computing times (seconds on SG), the proportion of sub-problems solved by DEC and LOPT. For DEC, this proportion corresponds to the number of subproblems solved divided by $t = \sqrt{p}$. For LOPT, this proportion corresponds to the number of subproblems solved divided by p. The results are averaged for 5 runs for Brd14051 and the methods were executed only once for Pla85900. From Table 8, we can conclude that:

Table 8. Computational results for DEC and LOPT for MWP instances Brd14051 and Pla85900.

		Quality (%)			Time (s. SG)			Proportion		
n	p	DEC (20, 50)	DEC (20, 200)	LOPT (7, 50)	DEC (20, 50)	DEC (20, 200)	LOPT (7, 50)	DEC (20, 50)	DEC (20, 200)	LOPT (7, 50)
14051	100	2.4	2.38	0.39	458	1931	3109	4.4	4.4	4.4
	200	1.9	2.03	0.30	336	1379	1632	4.3	4.6	3.3
	300	2.2	2.08	0.30	252	1073	1055	5.0	5.0	2.6
	400	2.2	2.04	0.26	214	915	885	4.9	5.2	2.4
	500	2.5	2.12	0.27	195	909	838	4.9	5.8	2.5
	600	2.4	1.99	0.26	184	799	707	5.1	5.8	2.3
	700	2.5	1.93	0.23	177	829	632	5.7	6.8	2.2
	800	2.5	1.97	0.24	149	760	555	4.8	6.8	2.1
	900	2.6	2.00	0.26	146	736	505	6.2	7.3	2.1
	1000	3.0	2.17	0.31	129	667	437	4.6	6.6	2.0
	1500	3.0	2.41	0.43	97	491	288	4.0	5.8	1.9
	2000	4.0	2.81	0.59	88	379	227	4.7	4.8	1.8
	2500	4.5	3.28	0.91	88	345	193	4.4	4.9	1.8
	3000	4.7	3.65	1.15	82	347	173	4.6	4.9	1.8
	5000	4.4	3.59	1.28	73	309	123	4.3	4.5	1.5
85900	1000	1.78	1.53	0.09	3557	9415	7634	4.2	5.2	2.9
	1500	1.97	1.70	0.17	3149	7885	5343	4.3	5.7	2.8
	2000	1.81	1.46	0.12	2819	6923	4750	4.1	5.1	2.5
	2500	1.84	1.48	0.05	3100	7031	4640	4.8	6.1	2.6
	3000	1.74	1.30	0.10	2405	5959	4532	4.1	5.2	2.6
	4000	1.72	1.30	0.04	2440	5503	4098	3.9	4.5	2.2
	5000	1.87	1.41	0.00	2597	5214	3423	4.1	4.3	2.0
	6000	1.67	1.37	0.05	2328	5392	2872	4.0	4.2	1.9
	7000	2.03	1.50	0.13	2276	4770	2526	4.3	4.3	1.9
	8000	2.07	1.53	0.03	2681	4685	2344	4.1	4.6	2.0
	9000	2.55	1.67	0.16	2796	4658	1992	4.1	4.3	1.8
	10000	2.78	1.80	0.16	2629	4863	1813	5.1	4.5	1.8
	15000	3.71	2.61	0.58	3144	5242	1552	5.3	6.0	1.8

– The solution quality provided by DEC slightly decreases as p increases; this is due mainly to the decrease in the solution quality provided by the CLS procedure when solving the subproblems.
– The computing times of DEC and LOPT diminisses as p increases. However, we can observe an increase in DEC computation times—as predicted by the complexity analysis—only for very large values of p. For LOPT we cannot observe such an increase, meaning

that solving the subproblems takes more time than finding close centres for generating the subproblems.

- The solution quality of LOPT is very good, generally well below 1% over the value of the best solution known.
- The proportion of subproblems solved by LOPT diminishes as p increases, showing that γ is smaller than 1.
- The proportion of subproblems solved by DEC seems to be constant as assumed in the complexity analysis of Section 3.2.

5. Conclusions

In this article we have proposed three new methods for heuristically and rapidly solving centroid clustering problems. First, we propose CLS, a candidate list search that rapidly produces good solutions to problems with a moderate number p of centres. Second, we propose LOPT, a procedure that locally optimizes the quality of a given solution. This method notably reduces the gap between the initial solution and the best solution known. The third method proposed, DEC, is based on decomposing the initial problem into subproblems. DEC and LOPT are well adapted to solve very large problems since their computing time increases more slowly with the number of entities than that of other methods in the literature. These methods can solve problems whose size is many order of magnitude larger than the problems treated up to now. Despite its speed, they produce solutions of good quality. The expected complexity of these procedures are given and experimentally verified on very large problem instances.

In fact, LOPT is a general optimization method that can be considered as a new meta-heuristic. Indeed it can be adapted for solving any large optimization problem that can be decomposed into independent sub-problems. LOPT has been shown to be very efficient for centroid clustering problems and vehicle routing problems. Future works should consider to apply LOPT to other combinatorial optimization problems.

The success of the methods presented in this paper could be explained as follows: solving problems with a very limited number of centres (e.g. below 15) is generally and easy task. Thanks to the use of an adequate neighbourhood, the CLS method allows problems up to 50–70 centres to be treated in a satisfactory way. DEC treats the problem at a high level and is able to determine the general structure of good solutions involving a large number of centres. Starting with a solution that has a good structure, LOPT is able to find very good solutions using a very simple improving approach. Therefore, it is interesting to remark that a very simple improving scheme can lead to a very efficient method if an initial solution with a good structure can be identified and an efficient neighbourhood is used. Indeed, the quality of the solutions obtained by the DEC + LOPT method rival what one would expect from a more elaborate meta-heuristic such as a genetic algorithm, taboo search or simulated annealing. The use of inadequate neighbourhood structures can explain the poor performances of previous implementation of such meta-heuristics. In summary, we can say that our methods open new horizons in the solution of large and hard clustering problems.

Acknowledgments

The author would like to thank Fred Glover for suggesting numerous improvement, Ken Rosing and Nicki Schraudolf for their constructive comments. This research was supported by the Swiss National Science Foundation, project number 21-45653.95.

Notes

1. The expert can even identify a number of Swiss Cantons in this figure. There are however differences that could be appropriate for solving political problems, such as the union of the South part of Jura to the Canton of Jura, the separation of the German-speaking part of Valais or the union of the small primitive Cantons.
2. In the context we use ALT, it is more interesting to have a non deterministic procedure. First, it may happen that ALT is called many times for solving the same (sub-) problem. With a non deterministic procedure, it is avoided to repeat exactly the same work. Then, let us mention that only non deterministic procedure can solve NP-hard problems in polynomial time if P $\neq$ NP. Therefore, our personal view is to consider non deterministic procedure potentially more interesting than deterministic ones, even if there is no theory supporting this for the moment.
3. More precisely, such a behaviour can be mathematically proven. In that case, we propose to follow the usual notation in statistics and to write $\bar{O}(.)$ for an expected running time derived from a mathematical analysis. Therefore it can be written that the complexity of *quick sort* is $\bar{O}(n \log n)$.
4. Without this assumption, the complexity is higher; with stronger assumptions (e.g. Euclidean distances), a lower complexity can be derived.

References

Beasley, J. (1985). "A Note on Solving Large p-Median Problems." *European Journal of Operational Research* 21, 270–273.

Bhaskaran, S. (1992). "Identification of Transshipment Center Locations." *European Journal of Operational Research* 63, 141–150.

Bongartz, I., P.H. Calamai, and A.R. Conn. (1994). "A Projection Method for l_p Norm Location-Allocation Problems." *Mathematical Programming* 66, 283–313.

Brimberg, J., P. Hansen, N. Mladenovic, and É.D. Taillard. (2000). "Improvements and Comparison of Heuristics for Solving the Multisource Weber Problem." *Operations Research* 48(1), 129–135.

Chen, R. (1983). "Solution of Minisum and Minimax Location-Allocation Problems with Euclidean Distances." *Naval Research Logistics Quarterly* 30, 449–459.

Cooper, L. (1963). "Location-Allocation Problems." *Operations Research* 11, 331–343.

Daskin, M.S. (1995). *Network and Discrete Location*. New-York: J. Wiley & Sons.

Diehr, G. (1985). "Evaluation of a Branch and Bound Algorithm for Clustering." *SIAM Journal on Scientific and Statistical Computing* 6, 268–284.

Dokmeci, V.F. (1977). "A Quantitative Model to Plan Regional Health Facility Systems." *Management Science* 24, 411–419.

Dyer, M.E. and A.M. Frieze. (1985). "A Simple Heuristic for the p-Centre Problem." *Operations Research Letters* 3, 285–288.

Edwards, A.W.F. and L.L. Cavalli-Sforza. (1965). "A Method for Cluster Analysis." *Biometrics* 21, 362–375.

Erlenkotter, D. (1978). "A Dual-Based Procedure for Uncapacitated Facility Location." *Operation Research* 26, 1590–1602.

Fleischmann, B. and J.N. Paraschis. (1988). "Solving a Large Scale Districting Problem: A Case Report." *Computers & Operations Research* 15, 521–533.

Glover, F. (1990). "Tabu Search for the p-Median Problem." Technical Report.

Goodchild, M.F. and V. Noronha. (1983). "Location-Allocation for Small Computers." University of Iowa, Monograph No. 8.

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

Hakimi, S.L. (1965). "Optimum Distribution of Switching Centers in a Communication Network and Some Related Graph Theoretic Problems." *Operations Research* 13, 462–475.

Hanjoul, P. and D. Peeters. (1985). "A Comparison of Two Dual-Based Procedures for Solving the p-Median Problem." *European Journal of Operational Research* 20, 387–396.

Hansen, P. and N. Mladenovic. (1999). "An Introduction to Variable Neighborhood Search." In S. Voss, S. Martello, I.H. Osman, and C. Roucairol (eds.), *Meta-Heuristic: Advances and Trends in Local Search Paradigms for Optimization*. Dordrecht: Kluwer. pp. 433–458.

Hansen, P., N. Mladenovic and É.D. Taillard. (1998). "Heuristic Solution of the Multisource Weber Problem as a p-Median Problem." *OR Letters* 22, 55–62.

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

Hidaka, K. and H. Okano. (1997). "An Approach for Solving a Real-World Facility Location Problem Using Digital Map." Presented at the EURO XV—INFORMS XXXIV Meeting, Barcelona, Spain.

Jancey, R.C. (1966). "Multidimensional Group Analysis." *Australian Journal of Botany* 14, 127–130.

Koontz, W.L.G., P.M. Narendra, and K. Fukunaga. (1975). "A Branch and Bound Clustering Algorithm." *IEEE Transactions on Computers* 24, 908–915.

Krau, S. (1997). "Extensions du problème de Weber." Thèse de doctorat de l'École Polytechnique de Montréal, Canada.

Lentnek, B., A. MacPerson, and D. Phillips. (1993). "Optimum Producer Services Location." *Environment and Planning A* 24, 467–479.

MacQueen, J.B. (1967). "Some Methods for Classification and Analysis of Multivariate Observations." In *Proceedings of the 5th Berkeley Symposium on Mathematical Statistics and Probability 1*. pp. 281–297.

Mirchandani, P. and R. Francis (eds.). (1990). *Discrete Location Theory*. Wiley-Interscience.

Mladenovic, N. and J. Brimberg. (1996). "A Degeneracy Property in Continuous Location-Allocation Problems." *Publication of the GERAD G-96-37*, University of Montreal, Canada.

Moreno, J., C. Rodrígez, and N. Jiménez. (1991). "Heuristic Cluster Algorithm for Multiple Facility Location-Allocation Problem." *Recherche Opérationnelle/Operations Research* 25, 97–107.

Murtagh, B.A. and S.R. Niwattisyawong. (1982). "An Efficient Method for the Multi-Depot Location-Allocation Problem." *Journal of the Operational Research Society* 33, 629–634.

Rapin, C. (1983). "Cours d'informatique générale." École Polytechnique Fédérale de Lausanne.

Reinelt, G. (1995). "TSPLIB95." Publication of the Institut für Angewandte Mathematik, Universität Heidelberg. Also available at http://www.iwr.uni-heidelberg.de/groups/comopt/software/TSPLIB95.

ReVelle, L. and R. Swain. (1970). "Central Facilities Location." *Geographical Analysis* 2, 30–42.

Rolland, E., D.A. Schilling, and J.R. Current. (1997). "An Efficient Tabu Search Procedure for the p-Median Problem." *European Journal of Operational Research* 96, 329–342.

Rosing, K.E. (1992). "An Optimal Method for Solving the (Generalized) Multi-Weber Problem." *European Journal of Operational Research* 58, 414–426.

Rosing, K.E., C.S. ReVelle, and H. Rosing-Vogelaar. (1979). "The p-Median and its Linear Programming Relaxation: An Approach to Large Problems." *Journal of the Operational Research Society* 30, 815–823.

Saaty, T.L. (1972). "Optimum Positions for Airports." *Naval Research Logistics Quarterly* 19, 101–109.

Späth, H. (1985). *Cluster Dissection and Analysis (Theory, Fortran Programs Examples)*. Chichester: Ellis Hérword.

Taillard, É.D. (1993). "Parallel Iterative Search Methods for Vehicle Routing Problems." *Networks* 23, 661–676.

Taillard, É.D. and S. Voß. (2002). "POPMUSIC: Partial Optimization Metaheuristic Under Special Intensification Conditions." In C.C. Ribeiro and P. Hansen (eds.), *Essay and Survey in Metaheuristics*. Kluwer, pp. 613–629.

Voß, S. (1996). "A Reverse Elimination Approach for the p-Median Problem." *Studies in Locational Analysis* 8, 49–58.

Ward, J.H. (1963). "Hierarchical Grouping to Optimize an Objective Function." *Journal of the American Statistical Association* 58, 236–244.

Weiszfeld, E. (1937). "Sur le point pour lequel la somme des distances de *n* points donnés est minimum." *Tôhoku Mathematical Journal* 43, 355–386.

Wesolowsky, G. (1993). "The Weber Problem: History and Perspectives." *Location Science* 1, 5–23.

Whitaker, R. (1983). "A Fast Algorithm for the Greedy Interchange for Large-Scale Clustering and Median Location Problems." *INFOR* 21, 95–108.