Improvements and Comparison of Heuristics for Solving the Multisource Weber Problem

Jack BRIMBERG
Department of Business Administration,
Royal Military College of Canada, Kingston, Canada K7K 5L0

Pierre HANSEN
GERAD and Ecole des Hautes Etudes Commerciales
5255, avenue Decelles, Montreal, Canada H3T 1V6

Nenad MLADENOVIĆ
GERAD and Ecole des Hautes Etudes Commerciales
5255, avenue Decelles, Montreal, Canada H3T 1V6

Éric D. TAILLARD
IDSIA, Corso Elvezia 36, CH-6900 Lugano, Switzerland.

Technical report IDSIA-33-97.

June 13, 1997

Abstract

The multisource Weber problem is to locate simultaneously m facilities in the Euclidean plane in order to minimize the total transportation cost for satisfying the demand of n fixed users, each supplied from its closest facility. Many heuristics have been proposed for this problem, as well as a few exact algorithms. Heuristics are needed to solve quickly large problems and to provide good initial solutions for exact algorithms. We compare various heuristics, i.e., alternative location-allocation (Cooper, 1964), projection (Bongartz *et al.* 1994), Tabu search (Brimberg and Mladenović 1996), p-Median plus Weber (Hansen, Mladenović and Taillard, 1996), Genetic Search and several versions of Variable Neighbourhood Search. It is found that most traditional and some recent heuristics give poor results when the number of facilities to locate is large and that Variable Neighbourhood Search gives consistently best results on average, in moderate computing time.

1 Introduction

The location-allocation problem requires locating a set of facilities and simultaneously allocating to these facilities demands for service from a set of customers in order to optimize some performance criterion. This problem occurs in many practical settings where facilities provide a homogeneous service, such as the location of plants, warehouses, retail outlets and public facilities. In the continuous version of the location-allocation problem, referred to as the multisource Weber problem, the objective is to generate m new facility sites in $\mathbb{R}^2$ to serve the demands of n customers or fixed points, in such a manner as to minimize the total transportation (or service) cost. The uncapacitated version which we will consider may be formulated as follows (e.g., see Love, Morris and Wesolowsky (1988)):

$$\begin{aligned} \min_{W,X} \quad & \sum_{i=1}^{n} \sum_{j=1}^{m} w_{ij} \|x_j - a_i\| && (1) \\ s.t. \quad \sum_{j=1}^{m} w_{ij} &= w_i, \quad i = 1, \ldots, n, \\ w_{ij} &\geq 0, \quad \forall i, j, \end{aligned}$$

where

$a_i = (a_{i1}, a_{i2})$ is the known location of customer i, $i = 1, \ldots, n$;

$X = (x_1, \ldots, x_m)$ denotes the matrix of location decision variables, with $x_j = (x_{j1}, x_{j2})$ being the unknown location of facility j, $j = 1, \ldots, m$;

w_i is the given total demand or flow required by customer i, $i = 1, \ldots, n$;

$W = (w_{ij})$ denotes the vector of allocation decision variables, where w_{ij} gives the flow to customer i from facility j, $i = 1, \ldots, n,\ j = 1, \ldots, m$;

$\|x_j - a_i\| = [(x_{j1} - a_{i1})^2 + (x_{j2} - a_{i2})^2]^{1/2}$ is the Euclidean norm.

The objective function above gives the total transportation cost, while the constraint set ensures that all customer demands are satisfied. Since there are no capacity constraints on the facilities, an optimal solution will have the demand at each customer served by the facility that is closest to it (ties being broken arbitrarily).

The main difficulty in solving (1) arises from the fact that the objective function is nonconvex (Cooper, 1967), and, in general, contains a large number of local minima. Consider for example the well-known 50 customer problem in Eilon *et al.* (1971). Using

200 randomly-generated starting solutions, the authors obtained 61 local minima for $m = 5$, where the worst solution deviated from the best by 40.9%. It was later shown (Krau, 1997) that the best solution was indeed the global optimum. Thus, due to the complex shape of the objective function, the problem falls in the realm of global optimization. The problem may also be viewed as an enumeration of the Voronoi partitions of the customer set, and is known to be NP-hard (Megiddo and Supowit, 1984).

As a consequence, exact methods for solving the location-allocation problem have been restricted until very recently to relatively small instances. Kuenne and Soland (1972) derive branch-and-bound algorithms which allow the solution of problem sizes of the order of 30 customers and 2 facilities or 15 customers and up to 4 facilities. The set reduction method and p-Median algorithm of Love and Morris (1975) for rectangular distances is restricted to similarly-sized problems. An efficient solution procedure is developed for the special case of Euclidean distances and $m = 2$ (Ostresh 1973a, Drezner 1984). This method is based on an observation that the subsets of customer locations allocated to the two facilities are separated by a straight line (Ostresh, 1975). Computation times for up to 100 customers were reported at less than 14 seconds on an Amdahl 470/V8 computer (Drezner, 1984). However, since there are $O(n^2)$ single facility location problems to solve, computation time will increase rapidly with n. More recently, Chen *et al.* (1992) have used a $d.-c.$ programming method to solve the two-facility case more efficiently. The authors observe a near linear increase in the computation time as n increases, and conclude that problem sizes of up to 1000 customers can be solved exactly. However, for three or more new facilities, the memory requirements of the method quickly restrict the problem sizes which can be attempted. A branch-and-bound algorithm of Ostresh (1973b) solves problems with three facilities and 50 customers. Rosing (1992) extends the approach of Ostresh (1973a) in the following way: for a given number of facilities, potential subsets of customers served by the same facility (or, in other words, market areas) are determined by separating iteratively their set of locations by straight lines $n - 1$ times. Then a large partitioning problem is solved, where each customer must belong to exactly one of those subsets. Unfortunately, the number of subsets augments very rapidly and hence problems of the size of 30 customers and 5 facilities or 25 customers and 6 facilities only can be solved.

The use of new tools has drastically augmented the size of problems which can be solved exactly. Krau (1997) proposes a column generation approach, combined with global optimization and branch-and-bound, which leads to the exact solution of instances with 287 customers and 2 to 100 facilities. These solutions are used for comparison of heuristics later in the paper. Combining this approach with a bundle method in the ℓ_1-norm (du Merle *et al.* 1997) to stabilize solution of the dual, leads to a very effective algorithm (Hansen *et al.* 1997) which can solve problems of up to 1000 customers and 100 facilities. To work well, both the column generation and the ℓ_1-norm bundle method require an initial solution quite close to the optimum. Therefore, they use in an initial step the best heuristics developed in the present paper. Conversely, they provide exact solutions as a benchmark for large instances studied in this paper. Knowledge of exact optimal values is important, as some heuristics give solutions very far from the best values obtained by other ones (i.e., 100% or more for large m), and if optimal values are unknown, one might wonder if all heuristics give bad solutions for large instances or not. As will be shown below, this is not the case.

For initialization of exact algorithms as well as for solution of very large problem instances which may occur in practice in terms of both parameters n and m, heuristic solution procedures are required. Many such methods have been proposed in the literature beginning with the well-known iterative location-allocation algorithm of Cooper (1964). This heuristic uses the property that the location and allocation phases of the problem are very easy to solve in isolation. Thus, given the facility locations, each customer is simply allocated to its nearest one. Alternatively, knowing the allocation of the customers among the facilities, the problem reduces to the solution of m independent single facility minisum problems which, due to the convexity of the objective function for normed distances, are readily solved by descent methods such as the Weiszfeld procedure or variants thereof (e.g., see Kuhn (1973), Rosen and Xue (1991), Brimberg and Love (1993), Drezner (1992), Frenk *et al.* (1994), and Brimberg *et al.* (1996)). Starting with an initial partition of the customer set, the Cooper algorithm alternates between the location and allocation phases until no further improvement can be made. Each iteration produces a lower value of the objective function until the process becomes trapped at a local minimum. A multi-start version involves repeating the Cooper algorithm many times from randomly-generated initial solutions, and retaining the best local minimum from the trials as the final solution. Variants of the Cooper algorithm are discussed in Scott (1970) and Baxter (1981), while Sullivan and Peters (1980) propose a method to cluster customers into mutually exclusive subsets, in each of which a facility is then located.

Love and Juel (1982) devise a heuristic method with a defined neighbourhood structure. This neighbourhood consists of all the points around a current solution which are obtained by exchanging a specified number of assignments of customers from their current facilities to new ones. Five variants of the proposed method are investigated. The first three algorithms denoted as H1 to H3, use a single-exchange, while the last two, H4 and H5, allow up to two exchanges. Different strategies such as first improvement and best improvement are employed to make descent moves from the current solution to a neighbourhood point. Again, since the search is local, the H heuristics of Love and Juel are only guaranteed to obtain a local minimum. The motivation for the larger neighbourhood of H4 and H5 is to better enable the algorithm to jump out of a “local optimum trap”, but obviously, this comes at a large cost in computation time.

A completely different heuristic approach is given by Chen (1983). Using an approximation by Charalambous and Bandler (1976), the objective function is transformed by giving an exponent $(-N)$ to all distances between customers and facilities and an exponent $(-1/N)$ to the sum of so modified distances from all facilities to each user. For N sufficiently large, this last quantity approaches the distance between the customer and its closest facility. In this way, the allocation decision variables are eliminated. The resulting problem is then solved by the Broyden-Fletcher-Shanno quasi-Newton method (e.g., Avriel, 1976). Good results, but not always the best known, are obtained with N set at 100.

Murtagh and Niwattisyawong (1982) propose a heuristic which uses MINOS, a large-scale nonlinear programming package, to solve simultaneously for both the locations and allocations. As the iterations proceed, the algorithm fixes any allocation decision variables (w_{ij}) which reach either a value of zero or w_i, and then updates only the free variables. The update uses a quasi-Newton approximation of the Hessian matrix within the space of the free variables, and at nondifferentiable points, a subgradient suggested by Kuhn (1973).

Moreno *et al.* (1990) construct a “drop” heuristic which begins with an initial solution of N clusters where N is chosen between m and $2m$. Then surplus facilities are dropped in a greedy manner until exactly m are left. This method was tested on problem sizes of up to 900 customers and 10 facilities, and obtained results comparable to the Cooper algorithm.

More recently, Bongartz *et al.* (1994) develop a projection method for solving the multisource Weber problem. Instead of assuming Euclidean distances, as is typically the case, the authors consider the more general l_p norm. As in Murtagh and Niwattisyawong (1982), the new method solves simultaneously for location and allocation decision variables. Simple projection formulas on subspaces of the domain are derived (instead of solving the system of equations in general), and used to find descent directions. The algorithm is guaranteed to converge to a local minimum. The authors test a multi-start version of their algorithm, where the initial solutions may be generated randomly or by partitioning customers in successive sets along a traveling salesman tour. The solutions are compared with multi-start versions of Cooper's algorithm, Murtagh and Niwattisyawong (1982), and Chen (1983). The projection method generally outperforms the other heuristics, but in several of the reported test problems, Cooper's algorithm comes in a close second. Thus, for the purposes of our current study, both these methods will be considered as the state-of-the-art.

Other heuristics have appeared after the projection method by Bongartz *et al.* (1994). These will for reference purposes be termed *recent* heuristics. Mladenović and Brimberg (1995) test a hybrid algorithm which takes random points in a k-exchange neighbourhood of the type used in the H-heuristics of Love and Juel (1982), and then applies Cooper's algorithm at each of these points. In Brimberg and Mladenović (1996a), elementary tabu search rules are added to the H3 heuristic to allow ascent moves away from a local optimum. Hansen *et al.* (1996b) obtain an approximate solution by solving a related p-Median problem followed by the solution of single facility Weber problems. This idea was first suggested by Cooper (1963). A variable neighbourhood concept is introduced by Brimberg and Mladenović (1996b), which systematically increases the number of exchanges (k) of the H-type neighbourhood to expand the search radius about a local optimum. Variable neighbourhood search may be viewed as a new metaheuristic, with a wide range of possible applications in combinatorial optimization (see Mladenović (1995) and Mladenović and Hansen (1996)).

There is a clear need for a comparative study of the heuristics which have appeared after the projection method of Bongartz *et al.* (1994). At the moment, there are several disconnected pieces, but no unified framework defining the current state-of-the-art. Thus we begin the next section with a review of the recent heuristics. In addition, we present a new genetic algorithm and new facility relocation heuristics which we have developed. The defined relocation neighbourhood structures are used to conduct a simple local search, or alternatively, Tabu and variable neighbourhood searches, producing several new methods. The subsequent section reports on an extensive empirical study comparing old, recent and new heuristics. The last section summarizes our conclusions, and suggests future directions of research.

In overview, the main objectives of our study are:

1. to update the state-of-the-art by reviewing under one roof the several 'recent' heuristics appearing after the projection method (Bongartz *et al.* (1994));
2. to add to this list 'new' heuristics, and hybrid versions thereof, which we are currently studying;
3. to conduct an extensive empirical study comparing the new and recent heuristics together and with the old establishment (Bongartz *et al.* and Cooper). Standard test problems will be used, but we will also consider much larger problem instances than previously reported in the literature. This will permit us to evaluate trends in performances of the various heuristics as problem size increases.

2 Recent Heuristics

In this section, we review several heuristic approaches to solve the multisource Weber problem which have been recently developed by us.

2.1 Tabu Search (TS)

This method (Brimberg and Mladenović (1996a)) is an adaptation of the H3 heuristic of Love and Juel (1982) within the Tabu search framework (Glover 1989, Glover 1990, Hansen and Jaumard 1990). A neighbourhood is constructed around a given (current) solution by considering all points obtained by a *single* exchange of a customer allocation from its current facility to another one. However, unlike the H3 heuristic, the tabu search algorithm will allow ascent moves from a local minimum. The parameters required by the basic method are *ntabu* and *nbmax*, for the length of the tabu list and the number of moves allowed without an improvement in the objective function, respectively. It is understood below that the facilities are always optimally located with respect to the specified allocations of customers by solving up to m independent single facility minisum problems.

Step 1 { *initialization* }: Obtain an initial solution by randomly partitioning the customer set $\{1, \ldots, n\}$ into m mutually-exclusive subsets A_j, and allocating A_j to facility $j, \forall j = 1, \ldots, m$ (located by using e.g. the Weiszfeld procedure). Denote this solution by X_c, and let Z_c equal the value of the objective function at X_c. Set $k = 0$, $(X_{best}, Z_{best}) = (X_c, Z_c)$, and the tabu list $= \emptyset$.

Step 2 { *neighbourhood search* }: Consider all points $X_i, i = 1, \ldots, n(m-1)$, in the one-exchange neighbourhood of X_c, except those that are not permitted by the tabu list. Retain the best solution (X^*, Z^*) from among the neighbourhood points. If $Z^* < Z_{best}$, set $(X_{best}, Z_{best}) = (X^*, Z^*)$.

Step 3 { *move to adjacent point* }: Place the reverse exchange $(X^* \rightarrow X_c)$ at the bottom of the tabu list, and remove the top element in the list (using a FIFO rule) if the length exceeds *ntabu*. If $Z^* < Z_c$, (descent move), set $k = 0$; else, $k = k + 1$. If $k > nbmax$, STOP; else $(X_c, Z_c) = (X^*, Z^*)$ and return to step 2. ∎

2.2 p-Median Heuristic (PM)

The p-Median problem is a discrete version of the multisource Weber problem, where p facility locations ($p = m$) are to be chosen from n nodes on a network representing the customer set. (For a review of the p-Median problem, see Mirchandani and Francis (1990)). The proposed p-Median heuristic (Hansen *et al.* (1996b)) solves the discrete problem optimally, where the facility locations are now restricted to the set of fixed points $\{a_1, \ldots, a_n\}$. It should be noted that the optimal solution of the continuous problem often has facilities located at or near customer sites. The travel distances between nodes are calculated with the Euclidean norm. The resulting p-Median problem is solved using the efficient code of Hanjoul and Peeters (1985).

Step 1: Define and solve a p-Median problem ($p = m$) with the same customers and demands as in the continuous problem, and the set of fixed points $\{a_1, \ldots, a_n\}$ as the set of sites for locating facilities. Let A_j be the subset of customers allocated to facility j in the optimal solution, $j = 1, \ldots, m$. (Note that the A_j are non-empty, mutually-exclusive sets, and $\cup_{j=1}^{m} A_j = \{1, \ldots, n\}$.)

Step 2: Solve m independent continuous single facility minisum problems (e.g., using the Weiszfeld procedure), where facility j serves exclusively subset A_j, $j = 1, \ldots, m$. Let x_j^* denote the optimal facility site thus obtained, $j = 1, \ldots, m$.

Step 3: A heuristic solution for the multisource Weber problem is given by $\{(x_j^*, A_j);\ j = 1, \ldots, m\}$, with objective function value

$$Z_{PM} = \sum_{j=1}^{m} \sum_{i \in A_j} w_i \|x_j^* - a_i\|. \qquad \blacksquare$$

A useful feature of the PM heuristic is that no parameters need to be specified by the analyst.

2.3 Variable Neighbourhood Search (VNS)

The variable neighbourhood search combines the elements of random search with a systematic way of exploring different regions of the solution space (Mladenović 1995, Brimberg and Mladenović 1996b, Mladenović and Hansen 1996). If a given neighbourhood does not produce a better solution, we augment the neighbourhood in order to move further away from the current solution and resume the search. The neighbourhood structure used here is a generalization of the fixed neighbourhood used in the H-heuristics of Love and Juel (1982) and in the hybrid algorithm of Mladenović and Brimberg (1995). We define the k-neighbourhood of a given solution as the set of all possible surrounding points obtained by exactly k exchanges of customer allocations from current facilities to new ones. This may be viewed as exchanging k existing branches on a bipartite graph representation with k new ones. The total number of points in the k^{th} neighbourhood is bounded by

$$\binom{n}{k} (m-1)^k,$$

which increases exponentially with k. The procedure randomly chooses a specified number b of points in this neighbourhood from which to conduct a local search with Cooper's algorithm. A basic form of VNS is outlined below. Generalizations of the method are discussed in Brimberg and Mladenović (1996b).

Step 1 { *initialization*: } Specify an initial solution, and run Cooper's algorithm to obtain a local optimum (X_c). Set $k = 1$.

Step 2 { *neighbourhood search*: }
(1) Select b points at random in the k-neighbourhood of X_c;
(2) for each of these points run Cooper's algorithm to obtain local minima $X_i, i = 1, \ldots, b$. (Note that these solutions will not in general be all unique);
(3) retain the best solution $X^* \in \{X_i, i = 1, \ldots, b\}$;
(4) if X^* is a better solution than X_c, $X_c = X^*$, $k = 1$, and return to the beginning of step 2; otherwise proceed to the next step.

Step 3 { *augmenting the neighbourhood*: }
(1) $k = k + 1$;
(2) if $k \leq k_{max}$, return to step 2;
otherwise if the stopping criterion is not satisfied, set $k = 1$ and return to step 2; else STOP (final solution is X_c). ∎

The parameters to be specified in VNS are b and k_{max}.

In the heuristic by Mladenović and Brimberg (1995), the neighbourhood size is fixed at a specified parameter value (k). Random points are chosen in the neighbourhood and the local descent from each of these points is carried out using Cooper's algorithm as above. This procedure referred to as fixed neighbourhood search , as well as VNS, may be easily modified to allow ascent moves.

3 New Heuristics

In this section, we describe a new genetic algorithm and a framework for new facility relocation heuristics.

3.1 Genetic Algorithm (GA)

Unlike random search methods which do not use any previous information, the genetic algorithm attempts to construct improved solutions from predecessors in an evolutionary type process (Holland, 1975). In this respect the genetic algorithm may be thought of as a more intelligent stochastic search technique. A genetic algorithm has already been developed for the multisource Weber problem by Houck *et al.* (1996). We give below the general framework of the algorithm followed by details of our implementation.

Step 1: Generate N different initial solutions (the population of solutions).

Step 2: Sort the population in nonincreasing order of solution quality measured by the value of the objective function.

Step 3:

Repeat

(1) select two solutions from the population;
(2) mix these solutions with a cross-over operator to create a new solution;
(3) modify the new solution with a mutation operator;
(4) insert the modified solution in the population and sort;
(5) remove solutions from the population with a culling operator;

Until a stopping criterion is satisfied. ■

To determine an initial solution in step 1 of GA, the facilities are located at m randomly-chosen fixed points. Using these starting locations, the alternating algorithm of Cooper is applied until a local minimum is reached. The process is repeated until N local minima are obtained to make up the population.

The selection operator in step 3 (1) generates two instances y_1, y_2 of a random variable uniformly distributed in the interval (0,1). The first solution selected from the sequenced population is identified as $s_1 = \lfloor y_1^2 \cdot Q + 1 \rfloor$, where Q is the number of solutions currently in the population and $\lfloor y \rfloor$ denotes the largest integer value less than or equal to y. The second solution is given by

$$s_2 = \begin{cases} s_2', & \text{if } s_2' < s_1 \\ s_2' + 1, & \text{otherwise,} \end{cases}$$

where $s_2' = \lfloor y_2^2 \cdot (Q-1) + 1 \rfloor$. Note that the squaring of y_1 and y_2 in the above formulas tends to generate smaller integer values for s_1 and s_2; that is, the tendency is to select the best solutions from the population in line with a *survival-of-the-fittest strategy.*

The cross-over operation combines the features of the two existing solutions, s_1 and s_2 (the *parents*), to produce a new solution s_3 (the *child*). In our implementation, each facility j is added to s_3 at a site it occupies in s_1 or s_2. A minimal separation distance d_{min} between facilities is specified in order to spread them out among the customers and avoid duplication of good sites. We use a d_{min} equal to the smallest distance between two customers. Let x_{ij} be the site of facility j in solution i. The cross-over operation works as follows. First set $x_{31} = x_{11}$ or x_{21}, with equal probability. Then, for each $j = 2, \ldots, m$, calculate

$$d_1 = \min_{t<j} d(x_{3t}, x_{1j}), \quad d_2 = \min_{t<j} d(x_{3t}, x_{2j});$$

If $(d_1 < d_{min})$ and $(d_2 > d_{min})$, set $x_{3j} = x_{2j}$; *else if* $(d_2 < d_{min})$ and $(d_1 > d_{min})$, set $x_{3j} = x_{1j}$; *else* set $x_{3j} = x_{1j}$ or x_{2j} with equal probability.

The mutation operator in step 3 (3) is simply a local improvement on the new solution s_3 using the Cooper algorithm to obtain a local minimum. If the population size exceeds a specified limit Q_{max}, the culling operator removes the worst solution in step 3 (5). The process will continue producing new generations of solutions indefinitely if it is not stopped. The stopping criterion is typically a limit on the number of iterations or on the execution time or the convergence of the algorithm is detected (i.e. all solutions of the population are the same). Thus, in summary, we need to specify the parameters N, Q_{max}, d_{min}, and a stopping criterion to implement the procedure.

3.2 Relocation Heuristics

Until now, local searches have been conducted in a neighbourhood of the current solution defined by a fixed number of customer-to-facility reallocations. We propose here a new local search procedure which constructs its neighbourhood as the set of points obtained by a given number of facility relocations. The simplest construction considers the relocation of a single facility to any unoccupied customer location (i.e., a customer which does not have a facility coincident with it). Since there are m candidates to choose from, and as many as n customer sites to reposition them at, there are $O(mn)$ points in the resulting neighbourhood.

Local searches which visit all the points in the neighbourhood will be referred to as *interchange* (CH) heuristics. Various strategies may be employed in this context to trade-off the accuracy or depth of the search at a neighbourhood point with speed. For example, the facilities may always be optimally located in continuous space (for the specified allocations), or forced to remain only at customer sites. In the latter case, the heuristic is solving the related discrete m-Median problem. The algorithm would adjust the facility locations in continuous space at well-defined times. The net effect would be to allow more iterations (with less precision) in the same amount of CPU time.

Instead of visiting all points in the interchange neighbourhood, an alternative strategy referred to as *drop and add* (DA) could be used. This procedure has been applied with success in other settings such as the Traveling Salesman Problem (see for example Gendreau *et al.* (1992)); however, to the best of our knowledge, this is the first application in location-allocation problems. The DA method decides using some criterion which is the best facility to drop, and only then, by another criterion, where is the best site to reinsert it. It follows that $O(m + n)$ points are investigated in place of $O(mn)$ of the interchange procedure. However, visiting $O(mn)$ solutions in an interchange neighbourhood may not be more time consuming than visiting $O(m + n)$

solutions in a DA neighbourhood because of the following facts: (i) A good drop strategy may be time-consuming (if, for example, a local minimum is obtained each time for the remaining $m-1$ facilities); likewise, for the add move. (ii) As shown by Whitaker (1983), the points in an interchange neighbourhood may be updated efficiently in a p-Median problem. We found that these results could also be applied to the continuous relocation neighbourhood we constructed.

A basic form of the DA heuristic follows:

Step 1 { *initialization*}: Find an initial solution X_c, and let Z_c be the corresponding value of the objective function.

Step 2 { *drop* }: Delete a facility at site $(x_{j_{del}1}, x_{j_{del}2})$ using some criterion. (All other facilities remain in their current locations.)

Step 3 { *add* }: Reinsert the facility at an unoccupied customer location $(a_{i_{new}1}, a_{i_{new}2})$ according to some other criterion.

Step 4 { *local improvement* }: Use Cooper's algorithm and the modified set of facility locations to find a local minimum (X^*, Z^*). If $Z^* < Z_c$, save the new currently best solution, $(X_c, Z_c) = (X^*, Z^*)$, and return to step 2; otherwise STOP. ■

The key features in the DA heuristic are seen to be the criteria which are used in steps 2 and 3. Several deletion and insertion rules were investigated, but for brevity, we will only report on the more succesful ones. The three best criteria for dropping a facility were found to be:

1. *Drop least useful (DLU).* Here each facility j is deleted in turn, and a local minimum W_j is obtained by Cooper's algorithm for the remaining $(m-1)$ facilities. The facility to be dropped corresponds to the minimum-valued W_j, i.e., $W_{j_{del}} = \min\{W_j, j = 1, \ldots, m\}$.

2. *Drop by second closest criterion (DSC).* Find the second closest facility to each customer. Now temporarily remove a facility j. Let r_j be its contribution in the current objective function, and let s_j equal the weighted sum of distances between customers temporarily without a facility and their second closest facility. Repeat the preceding step for $j = 1, \ldots, m$. Then facility j_{del} corresponds to the minimum difference, $s_j - r_j$.

3. *Drop by minimum potential criterion (DMP).* Define the potential of each facility j as the product $r_j d_j$, where r_j is its contribution to the objective function and d_j the distance to its closest facility. If facility j coincides with a customer i, then set $r_j = r_j + w_i$. Drop the facility j_{del} with the minimum potential.

The first drop procedure is intuitively the most appealing, since it identifies a facility whose removal will cause the least increase in the objective function. However, Cooper's algorithm must be called m times in each iteration, and hence, this procedure becomes time consuming for larger problem instances. The second and third drop procedures, on the other hand, are much faster, since a local improvement of the solution for the remaining $(m-1)$ facilities is not carried out.

Analogous versions of the drop step may be constructed for the add step. We mention only two of these:

1. *Add most useful (AMU).* Insert the deleted facility at an unoccupied customer location. Find a local minimum using Cooper's algorithm and the new set of facility sites. Repeat for each unoccupied customer location and retain the best solution.

2. *Add by second closest criterion (ASC).* Insert the deleted facility at an unoccupied customer location i. Reallocate those customers who are now closer to the newly-inserted facility than their current facility (which becomes the second closest). Calculate the decrease in the objective function $(s_i - r_i)$ attributed to the given insertion point. Repeat for each unoccupied customer location, and retain the insertion point which maximizes $(s_i - r_i)$.

The interchange and drop/add neighbourhoods may be enlarged by increasing the number of facility relocations from the current solution. In addition, we may allow ascent moves in the defined neighbourhood. This gives rise to a host of new heuristics based on tabu search and variable neighbourhood search. These algorithms apply the same steps as before, except that the *reallocation* neighbourhoods are now replaced by the newly-defined *relocation* neighbourhoods.

4 Computational Results

An extensive empirical study was carried out to compare the various heuristics - old, recent, and new - in a unified setting. We considered the following four problem configurations: the well-known 50 customer problem in Eilon *et al.* (1971), the 287 customer ambulance problem from Bongartz *et al.* (1994), and a 654 and 1060 customer problem listed in the TSP library (Reinelt (1991)). In each case, the number of facilities to locate was varied over a wide range. This provided a large number of problem instances from comparatively small sizes to much larger instances than previously reported in the literature. Thus, we were able to investigate the performance of the heuristics over an extensive range of problem difficulty.

$n = 50$		$n = 287$		$n = 654$		$n = 1060$	
m	*Optimal value*	m	*Optimal value*	m	*Best known value*	m	*Best known value*
2	135.5222	2	14427.5930	2	815313.2961	5	1851879.9
3	105.2139	3	12095.4422	3	551062.8811	10	1249564.8
4	84.1536	4	10661.4766	4	288190.9860	15	980132.1
5	72.2369	5	9715.6275	5	209068.7935	20	828802.0
6	60.9713	6	8787.5568	6	180488.2126	25	722061.2
7	54.5020	7	8160.3203	7	163704.1681	30	638263.0
8	49.9393	8	7564.2949	8	147050.7904	35	577526.6
9	45.6884	9	7088.1283	9	130936.1241	40	529866.2
10	41.6851	10	6705.0356	10	115339.0328	45	489650.0
11	38.0205	11	6351.5910	11	100133.2007	50	453164.0
12	35.0551	12	6033.0474	12	94152.0550	55	422770.0
13	32.3067	13	5725.1853	13	89454.7613	60	397784.4
14	29.6559	14	5469.6478	14	84807.6690	65	376759.5
15	27.6282	15	5224.7028	15	80177.0422	70	357385.0
16	25.7427	16	4981.9608	20	63389.0238	75	340242.0
17	23.9900	17	4755.1890	25	52209.5106	80	326053.2
18	22.2851	18	4547.3651	30	44705.1921	85	313738.2
19	20.6399	19	4342.0648	35	39257.2685	90	302837.0
20	19.3560	20	4148.8443	40	35704.4076	95	292875.1
21	18.0826	25	3348.7101	45	32306.9721	100	283113.0
22	16.8220	30	2716.9071	50	29338.0106	105	274576.0
23	15.6136	35	2238.1839	55	26699.1699	110	265801.0
24	14.4431	40	1900.8361	60	24504.3952	115	257605.0
25	13.3016	45	1630.3115	65	22747.0996	120	249584.0
26	12.3016	50	1402.5836	70	21468.1543	125	242930.0
27	11.4193	55	1203.9849	75	20312.9668	130	236154.0
28	10.4759	60	1055.1389	80	19193.8848	135	230431.0
29	9.5936	65	924.5547	85	18316.5391	140	224504.0
30	8.7963	70	814.2238	90	17544.3516	145	218279.0
31	7.9666	75	730.0354	95	16786.3887	150	212926.0
32	7.1814	80	655.3788	100	16087.6846		
33	6.4567	85	588.3680	105	15436.4004		
34	5.7484	90	529.2126	110	14830.1602		
35	5.0483	95	480.8592	115	14381.0566		
36	4.5471	100	441.2417	120	13921.5498		

Table 1. Optimal and best known values for test problems.

The different methods were compared on the basis of equivalent CPU times. Each problem instance was solved initially by 100 runs of Cooper's alternating algorithm from randomly-generated starting points. (The multistart version of Cooper's algorithm will be referred to as MALT from this point on.) The resulting CPU time was then used as a stopping criterion for the other heuristics. That is, the algorithm would be terminated at the completion of a local search if the total elapsed CPU time exceeded the stopping criterion; otherwise the iterations were allowed to continue. In cases where the stopping criterion could not be applied, such as the p-Median algorithm, actual CPU times to complete the first solution are reported.

	CPU	MALT		TS		VNS-1		PROJ		PM-1	
m	*time*	*Av.*	*Best*	*Av.*	*Best*	*Av.*	*Best*	*Av.*	*Best*	CPU	*Best*
2	0.94	0.00	0.00	3.71	0.00	3.39	0.00	1.45	0.00	0.06	0.05
3	1.59	0.00	0.00	1.51	0.00	1.35	0.00	2.36	0.00	0.06	0.26
4	2.21	0.00	0.00	3.04	0.00	1.23	0.00	4.16	0.00	0.06	0.02
5	2.82	0.00	0.00	1.09	0.00	0.15	0.00	2.93	0.00	0.08	0.63
6	3.75	0.00	0.00	6.41	0.00	1.25	0.00	2.61	0.00	0.05	0.21
7	4.00	0.00	0.00	3.51	0.00	2.97	0.00	4.65	0.25	0.05	0.24
8	4.23	0.06	0.00	4.61	0.00	0.63	0.00	4.94	0.72	0.05	0.00
9	4.26	0.38	0.00	4.00	0.79	1.78	0.00	6.03	0.79	0.05	0.00
10	4.27	0.51	0.00	8.16	1.77	2.65	0.00	8.14	2.40	0.05	0.00
11	4.31	2.24	1.34	11.73	5.83	2.53	0.00	9.19	0.00	0.05	0.00
12	4.27	2.65	0.35	13.75	7.84	1.06	0.00	9.41	0.61	0.05	0.00
13	4.24	3.90	1.57	15.51	2.22	1.45	0.00	10.21	3.19	0.05	0.30
14	4.18	4.22	2.18	16.49	6.35	1.91	0.00	12.78	5.65	0.05	0.00
15	4.14	4.73	0.84	19.55	9.06	1.39	0.00	12.34	5.33	0.05	0.41
16	4.05	4.93	1.53	16.88	8.98	0.76	0.00	12.18	5.44	0.06	0.00
17	3.91	4.88	1.11	20.88	9.62	0.03	0.00	13.18	5.75	0.05	0.00
18	3.90	6.39	3.70	20.72	8.73	0.32	0.01	16.16	7.56	0.06	0.00
19	3.80	5.66	1.41	21.11	10.15	0.04	0.00	18.21	7.02	0.05	0.00
20	3.75	5.81	0.68	19.87	6.34	0.20	0.00	18.11	5.18	0.05	0.04
21	3.67	5.77	2.11	13.45	6.40	0.16	0.00	19.36	2.58	0.09	0.35
22	3.60	6.04	3.76	16.17	5.18	0.30	0.00	20.10	7.81	0.05	0.44
23	3.52	4.93	1.00	23.38	11.74	0.42	0.00	22.08	7.71	0.09	0.89
24	3.46	5.77	2.92	22.24	9.64	0.27	0.00	24.56	12.88	0.10	0.95
25	3.38	5.58	2.50	23.68	9.64	0.19	0.00	26.84	20.38	0.05	0.82
Av.	3.59	3.10	1.12	12.98	5.01	1.10	0.00	11.75	4.22	0.06	0.23

Table 2. Old/recent heuristics and the 50-customer problem.

All methods were programmed in FORTRAN 77 except GA which uses C++. The codes were compiled using an optimizing option (*g++ -O3* for C++, and *f77 -cg92 -O4* for FORTRAN) and run on a SUN SPARC station 10.

Old and Recent Heuristics

Results for the 'old' and 'recent' heuristics are reported in Tables 2 to 5. Where applicable, two sets of values are given - the average and the best value of the objective function found from ten separate runs of the algorithm. These results are expressed as a % deviation from the best known solutions (listed in Table 1). It should be noted that the best known solutions listed for the 50 and 287 customer problems are also known to be global optimal solutions (Krau (1997)). The CPU times (in seconds) listed in the tables are for ten runs of MALT (with 100 restarts of Cooper's algorithm in each run), thus giving the total time to obtain the best solution.

	CPU	MALT		FNS		VNS-1		PROJ		PM-1	
m	*time*	*Av.*	*Best*	*Av.*	*Best*	*Av.*	*Best*	*Av.*	*Best*	CPU	*Best*
2	10.34	0.04	0.00	0.00	0.00	0.00	0.00	6.20	0.00	2.28	0.42
3	18.63	0.00	0.00	0.00	0.00	0.00	0.00	6.14	0.00	2.04	0.04
4	20.04	0.01	0.01	4.12	0.01	3.67	0.01	8.63	0.00	2.14	0.01
5	23.24	0.89	0.21	2.93	0.20	4.29	0.20	10.34	0.00	2.26	0.01
6	25.94	3.39	3.21	2.89	0.03	3.31	3.21	12.63	3.35	1.60	0.03
7	27.48	3.81	0.79	1.35	0.39	3.06	0.03	10.85	0.08	2.22	0.06
8	29.53	4.22	2.36	1.50	0.01	2.07	0.01	12.37	3.07	2.15	0.56
9	30.60	3.10	0.54	1.15	0.00	2.02	0.00	13.06	4.41	2.09	0.58
10	32.21	3.05	0.96	0.39	0.30	0.89	0.32	14.01	2.33	2.63	0.72
11	33.49	2.08	0.97	0.65	0.12	0.60	0.00	14.86	7.03	2.27	0.59
12	34.87	2.60	0.95	0.31	0.00	0.41	0.00	17.67	6.49	2.43	0.44
13	35.63	2.77	2.01	0.39	0.00	0.26	0.00	20.80	12.77	2.05	0.69
14	36.99	2.47	1.34	0.34	0.01	0.15	0.01	22.26	11.91	2.38	0.42
15	37.17	2.09	1.57	0.24	0.01	0.58	0.01	24.09	14.42	2.39	0.16
16	38.32	2.43	1.34	0.39	0.01	0.49	0.01	27.44	16.20	1.88	0.26
17	38.67	2.51	1.08	0.49	0.01	1.06	0.01	28.64	18.24	1.92	0.15
18	39.57	2.18	0.82	0.50	0.06	0.91	0.06	31.78	13.27	2.01	0.22
19	39.72	3.07	1.66	0.99	0.01	0.95	0.40	33.46	13.86	1.81	0.23
20	39.88	3.66	2.27	1.41	0.14	1.12	0.28	38.18	26.27	2.48	0.24
25	42.83	5.81	3.40	2.32	0.62	2.85	0.87	50.62	33.00	2.31	0.02
30	48.39	6.77	4.13	3.11	0.36	3.41	2.29	68.81	46.86	2.45	0.06
35	53.64	8.05	4.80	3.63	0.01	2.73	0.05	92.33	58.59	2.39	0.02
40	58.65	8.83	4.70	5.79	3.99	2.61	1.22	108.17	79.16	2.77	0.09
45	63.28	10.58	7.44	5.63	4.29	3.21	1.43	130.54	100.40	2.20	0.14
50	67.32	12.07	5.93	7.76	5.49	4.71	2.12	151.89	115.70	2.60	0.01
55	70.06	15.77	12.58	8.09	3.68	6.14	2.77	173.92	134.41	2.53	0.02
60	74.69	18.49	8.88	10.76	5.25	4.76	1.83	212.07	161.78	2.94	0.03
65	77.99	21.63	17.14	11.63	6.39	6.95	4.21	253.15	195.96	4.25	0.17
70	81.45	25.13	13.91	13.31	9.84	9.32	2.36	260.41	221.79	2.65	0.10
75	83.79	27.28	21.81	12.26	5.08	7.61	2.69	273.87	229.68	2.75	0.18
80	86.07	28.30	22.23	14.62	12.25	9.91	4.18	313.87	261.72	2.68	0.18
85	88.24	33.95	27.12	15.68	11.12	10.50	5.59	355.31	299.68	20.04	0.13
90	89.97	32.89	21.65	17.01	14.01	8.86	4.66	358.42	317.10	2.07	0.09
95	91.66	37.43	28.31	16.52	12.77	9.48	5.01	398.46	353.90	2.26	0.10
100	92.99	39.41	30.12	19.00	15.26	8.86	4.47	421.15	306.57	2.17	0.00
Av.	50.38	10.76	7.32	5.35	3.19	3.65	1.44	113.61	87.71	2.86	0.20

Table 3. Old/recent heuristics and the 287-customer problem.

Some general observations are inferred from a comparison of the results in Tables 2 to 5. These are summarized below:

	CPU	MALT		FNS		VNS-1		PROJ		PM-1	
m	*time*	*Av.*	*Best*	*Av.*	*Best*	*Av.*	*Best*	*Av.*	*Best*	CPU	*Best*
2	13.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	33.19	0.00
3	26.29	0.00	0.00	0.16	0.00	0.14	0.00	0.00	0.00	37.67	0.20
4	49.82	0.00	0.00	12.84	0.00	12.84	0.00	0.00	0.00	25.12	0.00
5	49.99	0.00	0.00	0.00	0.00	0.00	0.00	16.04	0.00	33.39	0.00
6	55.43	3.54	0.00	0.57	0.00	0.00	0.00	2.98	0.00	36.08	0.00
7	60.96	0.60	0.08	0.52	0.00	0.52	0.00	2.54	0.00	31.06	0.00
8	61.92	0.65	0.00	4.60	0.72	4.41	0.72	0.93	0.00	39.62	0.00
9	61.25	0.40	0.40	9.33	0.40	9.40	0.40	0.78	0.40	35.69	0.00
10	61.26	5.18	0.00	19.23	11.56	19.08	9.21	2.74	0.00	37.74	0.04
11	60.30	14.87	9.03	15.76	0.00	24.60	20.92	7.95	0.00	32.86	0.07
12	61.13	13.56	10.87	15.03	11.29	21.63	1.44	13.10	13.07	48.28	0.00
13	61.73	15.30	14.19	9.82	0.57	19.13	0.57	8.83	1.60	129.70	0.01
14	63.41	17.78	5.54	5.97	0.60	20.73	3.16	14.00	1.68	138.97	0.00
15	64.86	19.77	6.68	6.18	0.58	16.76	3.33	12.32	1.75	98.07	0.01
20	89.47	27.18	23.24	8.03	3.94	21.28	11.27	23.09	7.81	678.97	0.06
25	116.75	38.04	35.37	4.76	2.53	23.84	10.92	31.43	12.32	710.11	0.13
30	137.65	52.09	45.74	4.56	0.08	23.06	6.67	38.63	11.83	2111.24	0.22
35	154.34	59.99	50.99	2.05	0.22	22.64	10.88	40.36	19.35	803.95	0.38
40	176.75	63.99	55.97	1.33	0.41	24.63	9.63	41.87	26.97	881.32	0.56
45	186.37	68.45	62.73	2.66	1.62	30.42	7.75	43.64	27.55	648.16	0.50
50	204.21	67.83	62.03	3.79	2.72	19.18	7.89	47.37	33.52	1243.34	0.43
55	213.58	70.60	49.49	6.40	4.03	19.83	9.57	50.11	34.31	–	–
60	233.03	70.17	47.02	5.05	3.26	17.15	7.50	57.03	43.92	–	–
65	242.39	70.56	52.88	5.16	2.44	18.46	8.94	61.83	48.41	–	–
70	251.48	62.01	52.01	3.15	1.40	13.64	9.13	69.11	54.59	–	–
75	260.00	63.21	55.52	3.98	1.98	14.69	9.71	70.23	53.24	–	–
80	267.09	56.91	52.13	5.00	3.23	16.79	9.64	73.52	60.12	–	–
85	277.22	50.72	35.71	4.95	4.07	12.71	8.03	76.93	46.19	–	–
90	279.82	50.38	46.79	4.70	3.28	13.23	7.71	79.55	62.40	–	–
95	281.12	47.78	38.69	5.15	3.42	15.19	8.99	75.83	51.85	–	–
100	285.47	47.03	34.40	4.75	2.90	14.32	8.43	81.93	64.39	–	–
Av.	142.20	34.15	27.34	5.66	2.17	15.17	6.21	33.68	21.84	373.07	0.13

Table 4. Old/recent heuristics and the 654-customer problem.

1. At the lower values of m in each table, the old heuristics (MALT and the projection method (PROJ) of Bongartz *et al.*), perform as well or better than the recent heuristics. This implies that random restarts are an effective solution strategy for 'smaller' problem instances. This relates to the existence of relatively few local minima.

2. The quality of the solutions obtained by MALT and PROJ deteriorates with increasing m. The worst case occurs with PROJ in Table 3, where deviations in excess of 100% are observed for the best solution and $m \geq 45$. The relatively poor performance of both methods may be attributed to the fact that the number of local minima increases with problem size at an exponential rate giving rise to a *central-limit catastrophe* (Boese *et al.*

(1994)). As a result, procedures which use random restarts lose their effectiveness. The breakdown of PROJ in the 287-customer problem may also be attributed to the weight structure in this problem. (Unit weights are used in other test problems.) Also note that PROJ is slower than MALT, and hence, fewer local descents are achieved in the allotted time. Furthermore, PROJ may terminate at a degenerate local minimum where one or more facilities are at locations which are serving no customers (Mladenović and Brimberg 1996)). This problem is eliminated in our implementation of MALT by the addition of an insertion procedure; however, PROJ uses the original code supplied by its authors. It is interesting to note on the other hand, that PROJ performs best overall in Table 5 for the 1060-customer problem. This may be due to the use of traveling salesman solutions in PROJ to generate starting points.

	CPU	MALT		FNS		VNS-1		PROJ	
m	*time*	*Av.*	*Best*	*Av.*	*Best*	*Av.*	*Best*	*Av.*	*Best*
5	121.74	0.00	0.00	0.22	0.00	0.21	0.00	0.13	0.00
10	242.04	0.03	0.01	0.10	0.00	0.10	0.00	1.09	0.04
15	320.98	0.09	0.04	0.27	0.01	0.41	0.00	2.40	0.14
20	377.04	1.17	1.09	2.17	1.72	2.10	1.71	3.66	0.36
25	406.02	3.65	2.98	4.94	2.92	5.10	2.81	4.74	0.07
30	507.45	8.20	7.28	7.73	5.89	8.51	5.44	5.11	1.17
35	607.56	10.88	8.77	9.92	6.34	11.03	8.77	5.57	0.80
40	679.96	14.92	13.19	12.77	10.58	15.68	11.18	7.46	1.53
45	718.97	19.70	18.37	13.20	8.94	18.82	10.86	8.85	2.44
50	740.57	23.75	22.36	13.73	10.09	23.37	15.81	10.20	2.14
55	802.29	28.09	25.67	11.88	8.08	26.28	21.65	10.81	3.69
60	855.83	32.38	30.19	13.20	10.15	26.31	20.59	11.79	2.83
65	905.99	35.02	32.83	11.18	9.23	28.29	20.74	12.18	4.87
70	956.30	38.59	37.17	10.69	7.38	32.75	23.22	12.59	3.82
75	1005.61	39.75	37.29	10.02	7.60	31.77	23.33	13.21	4.22
80	1020.64	41.77	38.00	9.46	3.77	31.13	27.05	13.05	4.49
85	1028.75	42.79	40.71	7.87	4.48	31.19	24.24	13.42	5.08
90	1044.53	42.75	39.10	7.16	3.83	32.67	23.60	13.44	5.51
95	1084.29	43.59	35.57	4.04	2.08	34.31	28.30	13.60	5.41
100	1092.74	43.59	37.26	3.37	2.02	33.33	23.34	13.20	5.02
Av.	725.96	23.57	21.43	7.73	5.29	19.70	14.67	8.83	2.72

Table 5. Old/recent heuristics and the 1060-customer problem.

3. The Tabu search method (TS-1) performs poorly in comparison to MALT for large problem sizes (see Table 2). This is attributed to the neighbourhood structure in TS which results in a very slow local descent or ascent. Thus, relatively few iterations are completed within the imposed time limit. For this reason, we report TS-1 only in Table 2. For small problem sizes, TS-1 is seen to be competitive with the other methods within the imposed time limit.

4. The variable neighbourhood search (VNS-1) reported in Tables 2 to 5 uses in all cases parameter values of $b = 1$ (number of points randomly selected in a given neighbourhood), and $k_{max} = n$ (maximum number of customer reallocations or largest neighbourhood). No attempt was made to find the best parameter values for individual problems, but rather, a 'parameterless' version was implemented to see how the algorithm in its simplest form could perform over all problem sizes.

Referring to the average error summary in the tables, we observe that VNS-1 significantly outperformed MALT within the same CPU time. For the 50-customer problem, the best solution obtained by VNS-1 was optimal in almost all cases! However, the results were not uniform for the other problem sets. It is also interesting to note that the fixed neighbourhood search (FNS) with number of reallocations $k = n/2$, obtained substantially better results than VNS-1 for the problem sets in Tables 4 and 5. This implies that the variable neighbourhood search is sensitive to the parameter settings. Thus, for best results, the parameters should be adjusted for individual problems (or, CPU time increased). Alternatively, b and k_{max} may vary according to an intensification/diversification strategy during the execution of the algorithm.

5. The solutions obtained by the p-Median algorithm (PM-1) are very good in comparison to the other methods reported in Tables 2 to 5. With the exception of the 50-customer problem, PM-1 outperforms VNS-1 by a considerable margin. However, the execution time for PM-1 to complete a solution far exceeded the time limit imposed on VNS-1, so that a direct comparison of the two methods cannot be made. Note that results for PM-1 are not listed for execution times exceeding a 3000 second limit. (This is also the reason PM-1 is not included in Table 5.)

New Descent Relocation Heuristics

We begin with a discussion of results for the *drop/add* (DA) algorithm. As noted in the description of this method in section 3, several criteria may be selected to determine which facility to remove and where to insert it back. This provides a large number of possible drop/add strategies. We will report on only a few of the more promising combinations.

Table 6 provides results on four DA strategies for the 287-customer problem. The *Av.* column gives the average result from ten random restarts, while the next column lists the best result. Computation times are totals for the ten descents. Referring to the best solutions, we first observe that the *drop least useful*, *add most useful* strategy (DLU, AMU) worked very well. The % deviation from optimal is low irrespective of m, and regularly, the optimal solution itself is obtained. Note however that the computational times are much higher than for the other DA heuristics. This is attributed to the excessive number of Cooper iterations carried out between adjacent moves in the solution space.

	(DLU,AMU)			(DLU,ASC)			(DMP,ASC)			(DSC,ASC)		
m	*Av.*	*Best*	*Time*	*Av.*	*Best*	*Time*	*Av.*	*Best*	*Time*	*Av.*	*Best*	*Time*
2	0.00	0.00	133.5	0.38	0.38	0.6	0.38	0.38	0.4	0.38	0.38	0.4
3	0.00	0.00	99.7	0.14	0.00	1.1	0.17	0.00	0.6	0.17	0.00	0.6
4	0.01	0.00	169.3	0.01	0.01	2.4	0.01	0.01	0.8	0.41	0.00	0.8
5	0.01	0.01	227.0	0.01	0.01	4.0	0.01	0.01	1.1	1.11	0.01	1.0
6	0.02	0.02	294.4	0.31	0.03	4.4	0.89	0.02	1.4	4.49	0.03	1.0
7	0.03	0.02	296.1	0.03	0.03	6.4	0.83	0.02	1.5	1.49	0.03	1.2
8	0.00	0.00	328.1	0.77	0.01	6.2	2.86	0.01	1.3	3.03	0.01	1.3
9	0.29	0.00	297.8	0.51	0.00	6.8	1.84	0.50	1.3	2.90	0.08	1.2
10	0.00	0.00	366.9	0.28	0.00	8.2	0.66	0.40	1.4	2.33	0.42	1.4
11	0.23	0.00	337.6	0.76	0.15	10.5	1.78	0.47	1.4	3.42	0.49	1.3
12	0.19	0.00	348.0	0.97	0.00	12.5	1.39	0.16	1.6	2.86	0.00	1.4
13	0.35	0.00	378.0	0.83	0.11	12.5	1.66	0.32	1.5	3.45	0.32	1.3
14	0.42	0.01	367.6	1.23	0.27	16.0	1.43	0.20	1.7	3.84	1.03	1.3
15	0.53	0.05	369.7	2.18	0.65	15.3	2.67	0.76	1.6	3.16	1.56	1.5
16	0.61	0.18	382.9	1.21	0.34	14.7	2.39	0.66	1.5	2.89	0.95	1.4
17	0.43	0.09	470.8	1.03	0.23	19.6	3.40	0.76	1.5	4.63	2.72	1.4
18	0.09	0.01	535.7	0.96	0.06	23.6	2.89	0.76	1.9	4.81	2.43	1.5
19	0.12	0.01	621.2	1.65	0.34	23.1	2.67	0.34	2.0	5.18	2.52	1.7
20	0.11	0.01	538.3	2.86	0.03	21.6	3.03	0.93	1.9	3.29	1.45	1.9
25	0.31	0.00	675.3	0.68	0.01	37.1	2.32	0.61	3.0	6.68	1.46	2.2
30	0.08	0.03	693.9	0.36	0.03	50.0	0.86	0.05	3.4	5.37	2.74	2.8
35	0.02	0.01	809.3	0.03	0.01	66.6	1.41	0.01	3.8	4.40	1.35	3.5
40	0.10	0.01	1081.1	3.44	0.01	93.1	3.94	1.16	4.0	5.50	1.50	4.2
45	0.15	0.01	1175.0	3.96	0.01	113.5	4.00	1.53	3.8	6.15	2.93	4.4
50	0.02	0.00	1462.4	3.23	0.07	142.3	2.97	2.11	4.9	5.99	1.23	4.9
55	0.03	0.00	1721.0	3.87	0.00	213.3	3.93	1.58	6.2	8.78	5.31	5.7
60	0.15	0.01	1905.9	0.23	0.01	275.5	2.43	1.19	7.1	8.47	1.85	6.2
65	0.09	0.01	1934.9	0.61	0.01	314.6	3.48	0.83	7.2	13.94	8.08	5.5
70	0.07	0.00	2003.5	0.60	0.06	363.9	2.70	0.76	7.9	9.01	3.95	6.7
75	0.08	0.04	2462.3	1.18	0.08	452.1	1.62	0.12	10.3	10.57	6.77	7.6
80	0.18	0.05	2444.6	0.51	0.12	519.1	3.14	1.59	9.4	8.22	4.40	8.3
85	0.14	0.04	2865.1	0.66	0.04	601.4	2.82	1.09	10.0	7.35	3.70	9.4
90	0.06	0.01	3190.0	3.47	0.01	656.9	3.07	0.11	11.6	9.60	5.14	10.1
95	0.11	0.02	3171.0	0.15	0.02	846.5	2.28	0.55	13.4	10.05	5.49	10.8
100	0.01	0.00	3605.0	1.55	0.00	925.8	3.58	1.54	12.2	8.54	3.95	11.5
Av.	0.14	0.02	1078.9	1.16	0.09	168.0	2.16	0.62	4.1	5.21	2.12	3.6

Table 6. Descent DA heuristics in 287-customer problem.

A compromise between quality and CPU time is obtained with (DLU, ASC), since the second-closest criterion is fast to implement. (Also AMU takes much longer than DLU, especially when m is small.) The fastest procedures are given by (DMP, ASC) and (DSC, ASC), since optimal relocation of facilities is not carried out during the drop and add phases. The listed computation times for (DMP, ASC) and (DSC, ASC) are substantially less than those for MALT. Yet, a comparison shows for example that (DMP, ASC) performs significantly better than MALT and the "parameterless" VNS-1.

Table 7 presents a summary of results for four local search heuristics using relocation neighbourhood structures. The values listed here are combined average percentage deviations and CPU times over the same sets of values of m reported in the previous tables. The columns have the same interpretation as in Table 6.

	DA-1 ≡(DMP,ASC)			DA-2 ≡(DSC,ASC)			CH			PM-2		
Pb.	*Av.*	*Best*	*Time*	*Av.*	*Best*	*Time*	*Av.*	*Best*	*Time*	*Av.*	*Best*	*Time*
50	7.06	2.07	0.07	0.98	0.21	0.15	1.18	0.11	0.17	0.58	0.02	0.04
287	2.16	0.62	4.13	5.21	2.12	3.65	0.26	0.04	3.35	0.28	0.07	1.33
654	6.64	4.02	18.66	0.58	0.36	32.42	0.95	0.53	29.93	0.29	0.01	8.11
1060	2.83	1.57	85.40	0.48	0.21	115.09	1.05	0.67	93.29	0.67	0.21	71.03
Av.	4.67	2.07	27.07	1.81	0.73	37.83	0.86	0.34	31.69	0.45	0.08	20.13

Table 7. Summary results for local descent heuristics with *Relocation* neighbourhoods.

The first two heuristics in Table 7 are the 'fast' versions of drop/add reported in Table 6. The next heuristic (referred to as CH for interchange) considers all possible location interchanges of a single facility from its current position to an unoccupied fixed point. One iteration of Cooper's algorithm (allocate-locate) is run only at the best neighbourhood point to save computation time. The fourth heuristic PM-2 is a discrete version, where the facilities remain at fixed points during the interchange process. When no further improvement can be made (the current solution is a local minimum in its neighbourhood), one iteration of Cooper's algorithm is performed as in CH.

Comparing the four relocation heuristics in Table 7, we see that no one method dominates the others. DA-2 obtains better results than DA-1 in three of the four problem sets, but not in the 287-customer set. Interestingly, the same pattern occurs between PM-2 and CH. As a group, CH and PM-2 obtain better solutions than the DA heuristics in all problem sets. However, DA-2 obtains the best average deviation in the 1060-customer set, and outperforms CH here and in the 654-customer set. More importantly, we observe that these new local search heuristics are very efficient compared with the earlier methods reported in Tables 2 to 5. With the exception of VNS-1 in Table 2 (50-customer set), large net improvements are obtained over the earlier methods in a small fraction of the CPU time.

New heuristics

Tables 8 to 11 report on our new heuristics. These consist of a genetic algorithm (GA), and the latest versions of Tabu search and variable neighbourhood search using a relocation neighbourhood structure in place of the previous reallocation structure. The parameter settings used in GA are $N = 15$ for the initial population, and $Q_{max} = 30$ for population size. The first Tabu search procedure (TS-2) uses a drop/add neighbourhood and DA-2 strategy, while the second (TS-3) uses the interchange neighbourhood of CH.

In both cases, a tabu list is maintained of the last 15 fixed point insertions. The same neighbourhood structures are utilized again in the variable neighbourhood searches VNS-2 and VNS-3, while VNS-4 borrows the discrete interchange structure of PM-2. In the three VNS versions, the maximum neighbourhood size, k_{max}, is set equal to m, and one point is randomly chosen in each neighbourhood. Only one iteration of Cooper's algorithm is performed at a selected neighbourhood point, in order to make efficient use of CPU time. Note that these CPU times are not recorded in the tables, since they were set to the execution times previously obtained for MALT.

Referring to Tables 8 to 11, the following observations are made:

1. GA performs very well over the lower range of m values in all four problem sets. However, the percentage deviation tends to increase with m. This may be attributed to an exponentially-increasing number of local minima, and the fact that GA has time to visit only a small number of them.

			DA-2				CH				P-MED	
	GA		TS-2		VNS-2		TS-3		VNS-3		VNS-4	
m	*Av.*	*Best*	*Av.*	*Best*	*Av.*	*Best*	*Av.*	*Best*	*Av.*	*Best*	*Av.*	*Best*
2	0.00	0.00	0.92	0.00	0.08	0.00	0.01	0.00	0.00	0.00	0.01	0.00
3	0.00	0.00	0.93	0.64	0.24	0.00	0.02	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.12	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	0.01	0.00	0.19	0.01	0.11	0.00	0.33	0.00	0.00	0.00	0.19	0.00
6	0.04	0.00	0.61	0.21	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
7	0.07	0.00	0.02	0.00	0.00	0.00	1.34	0.00	0.00	0.00	0.02	0.00
8	0.23	0.00	0.19	0.00	0.00	0.00	0.95	0.00	0.00	0.00	0.00	0.00
9	0.68	0.00	0.57	0.00	0.00	0.00	0.78	0.00	0.00	0.00	0.00	0.00
10	0.42	0.00	0.01	0.00	0.00	0.00	1.43	0.00	0.00	0.00	0.00	0.00
11	1.24	0.00	0.47	0.00	0.00	0.00	1.69	0.00	0.00	0.00	0.00	0.00
12	1.72	0.00	0.40	0.00	0.00	0.00	2.06	0.00	0.00	0.00	0.00	0.00
13	1.47	0.00	0.68	0.00	0.00	0.00	0.34	0.00	0.00	0.00	0.03	0.00
14	2.79	0.41	0.46	0.00	0.00	0.00	1.76	0.00	0.00	0.00	0.00	0.00
15	3.10	0.16	0.67	0.00	0.00	0.00	0.36	0.00	0.00	0.00	0.00	0.00
16	1.71	0.11	0.86	0.26	0.03	0.00	1.37	0.52	0.00	0.00	0.00	0.00
17	3.99	0.44	1.31	0.00	0.04	0.00	0.66	0.00	0.00	0.00	0.00	0.00
18	4.56	0.48	1.45	0.00	0.03	0.00	0.71	0.00	0.00	0.00	0.03	0.00
19	6.25	2.27	1.27	0.09	0.01	0.00	0.46	0.00	0.00	0.00	0.00	0.00
20	4.41	0.52	1.37	0.68	0.00	0.00	0.84	0.00	0.00	0.00	0.07	0.00
21	4.91	1.56	0.91	0.00	0.00	0.00	0.84	0.13	0.00	0.00	0.08	0.00
22	4.42	0.97	1.14	0.54	0.05	0.00	0.70	0.00	0.00	0.00	0.09	0.00
23	6.66	1.82	1.23	0.00	0.12	0.00	0.91	0.24	0.02	0.00	0.09	0.00
24	8.39	3.79	1.33	0.37	0.02	0.00	0.98	0.00	0.00	0.00	0.10	0.00
25	9.73	6.07	2.07	0.00	0.00	0.00	0.42	0.00	0.00	0.00	0.08	0.00
Av.	2.78	0.78	0.80	0.12	0.03	0.00	0.79	0.04	0.00	0.00	0.03	0.00

Table 8. New heuristics and the 50-customer problem.

2. TS-3 outperforms TS-2 in Tables 8 and 9, but the reverse is true in Tables 10 and 11. Based on this limited data, we might infer that the interchange neighbourhood (CH) is better suited for smaller problems, while the drop/add (DA-2) should be used for larger instances. This may be due to the longer CPU time required to make a move in the interchange neighbourhood, which results in fewer iterations. Also note that the best TS heuristic outperforms GA in each table.

			DA-2				CH				P-MED	
	GA		TS-2		VNS-2		TS-3		VNS-3		VNS-4	
m	*Av.*	*Best*	*Av.*	*Best*	*Av.*	*Best*	*Av.*	*Best*	*Av.*	*Best*	*Av.*	*Best*
2	0.04	0.00	0.07	0.00	0.58	0.24	0.02	0.00	0.00	0.00	0.00	0.00
3	0.00	0.00	0.05	0.00	0.03	0.00	0.00	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.02	0.01	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
5	0.04	0.00	0.09	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01	0.01
6	0.53	0.00	2.56	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02	0.02
7	0.21	0.00	3.46	0.03	0.04	0.03	0.03	0.03	0.02	0.02	0.03	0.03
8	0.35	0.00	3.28	0.09	0.01	0.00	0.01	0.01	0.01	0.00	0.28	0.01
9	0.38	0.00	4.23	0.01	0.04	0.00	0.01	0.00	0.00	0.00	0.33	0.01
10	0.57	0.00	3.71	0.00	0.17	0.01	0.23	0.00	0.00	0.00	0.23	0.00
11	0.58	0.00	1.21	0.00	0.16	0.00	0.03	0.00	0.00	0.00	0.23	0.00
12	0.82	0.18	2.41	0.14	0.39	0.14	0.09	0.00	0.06	0.00	0.24	0.00
13	0.61	0.00	3.06	0.31	0.53	0.34	0.10	0.00	0.03	0.00	0.36	0.00
14	0.81	0.39	1.53	0.22	0.43	0.22	0.21	0.01	0.01	0.01	0.31	0.20
15	0.93	0.17	2.58	0.51	0.32	0.01	0.11	0.01	0.01	0.01	0.13	0.10
16	1.41	0.03	2.91	0.14	0.64	0.11	0.26	0.01	0.01	0.01	0.14	0.10
17	1.23	0.79	3.39	0.85	0.96	0.63	0.56	0.01	0.09	0.01	0.17	0.01
18	1.24	0.42	2.78	0.38	0.76	0.17	0.29	0.06	0.06	0.01	0.08	0.01
19	1.96	0.84	3.88	2.23	1.03	0.16	0.53	0.07	0.06	0.01	0.09	0.01
20	1.59	0.07	4.09	0.42	1.16	0.11	0.41	0.13	0.05	0.01	0.06	0.01
25	2.33	0.01	3.75	1.50	1.13	0.16	0.36	0.01	0.01	0.00	0.01	0.00
30	3.91	0.53	6.66	1.44	1.92	0.31	0.57	0.03	0.03	0.03	0.03	0.03
35	4.09	0.74	5.77	1.56	2.59	0.61	0.12	0.01	0.01	0.01	0.03	0.01
40	5.61	4.19	5.45	1.90	3.11	0.92	0.54	0.01	0.19	0.01	0.09	0.01
45	5.25	2.60	4.81	0.67	3.34	0.68	0.18	0.01	0.01	0.01	0.04	0.01
50	7.11	4.53	4.78	1.35	2.95	1.31	0.17	0.00	0.00	0.00	0.00	0.00
55	8.69	5.76	5.89	3.70	4.74	1.65	0.20	0.00	0.12	0.00	0.02	0.00
60	8.90	5.82	5.22	0.29	4.53	0.65	0.31	0.08	0.14	0.00	0.32	0.01
65	10.04	4.82	6.53	2.72	4.75	1.65	0.29	0.01	0.03	0.01	0.12	0.01
70	11.91	6.65	9.01	5.62	6.86	4.05	0.75	0.03	0.07	0.03	0.08	0.03
75	11.57	7.68	7.38	3.00	4.59	2.45	0.25	0.04	0.09	0.04	0.16	0.05
80	12.00	4.96	5.58	2.75	3.59	2.34	0.29	0.04	0.08	0.04	0.22	0.06
85	12.39	8.35	6.50	3.94	4.66	2.49	0.34	0.04	0.04	0.04	0.13	0.05
90	15.38	10.51	8.87	4.65	5.53	3.09	0.19	0.02	0.02	0.01	0.09	0.02
95	14.38	10.44	6.94	2.57	4.17	2.43	0.06	0.02	0.02	0.02	0.18	0.02
100	14.29	10.30	6.03	3.42	3.89	1.52	0.06	0.01	0.02	0.01	0.10	0.01
Av.	4.60	2.59	4.13	1.33	1.99	0.81	0.22	0.02	0.04	0.01	0.12	0.02

Table 9. New heuristics and the 287-customer problem.

3. A similar relation is observed between VNS-2 and VNS-3. Once again, the drop/add neighbourhood appears to be a better choice for larger problems. We might infer that the DA strategy in VNS-2, although more time-consuming, has more success finding descent directions in large problems as compared with the random selection of neighbourhood points in VNS-3. Hence, VNS-2 would be able to make more descent moves in the allotted CPU time.

			DA-2				CH				P-MED	
	GA		TS-2		VNS-2		TS-3		VNS-3		VNS-4	
m	*Av.*	*Best*	*Av.*	*Best*	*Av.*	*Best*	*Av.*	*Best*	*Av.*	*Best*	*Av.*	*Best*
2	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
3	0.00	0.00	0.17	0.00	0.13	0.00	0.18	0.00	0.00	0.00	0.00	0.00
4	0.00	0.00	0.03	0.02	0.07	0.01	0.00	0.00	0.00	0.00	0.00	0.00
5	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
6	0.57	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
7	0.02	0.00	0.17	0.00	0.02	0.00	0.04	0.00	0.00	0.00	0.00	0.00
8	0.05	0.00	0.00	0.00	0.02	0.00	0.43	0.00	0.00	0.00	0.04	0.00
9	0.04	0.00	0.01	0.00	0.07	0.04	0.43	0.40	0.00	0.00	0.00	0.00
10	0.10	0.00	0.01	0.00	0.01	0.00	0.09	0.00	0.00	0.00	0.03	0.00
11	2.14	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
12	2.10	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.15	0.00
13	1.07	0.00	0.10	0.01	0.09	0.00	0.60	0.19	0.01	0.09	0.11	0.00
14	0.52	0.01	0.04	0.04	0.02	0.01	0.55	0.52	0.01	0.00	0.01	0.00
15	1.55	0.00	0.05	0.04	0.05	0.01	0.54	0.53	0.02	0.01	0.02	0.00
20	0.86	0.03	0.03	0.02	0.03	0.01	0.30	0.28	0.06	0.00	0.07	0.00
25	3.01	1.43	0.05	0.02	0.01	0.01	0.55	0.24	0.21	0.01	0.07	0.01
30	5.43	0.05	0.04	0.01	0.04	0.01	0.17	0.12	0.06	0.01	0.01	0.01
35	6.58	3.09	0.10	0.01	0.08	0.01	0.48	0.09	0.22	0.03	0.13	0.01
40	7.67	2.79	0.13	0.02	0.19	0.12	1.45	0.93	0.40	0.14	0.28	0.04
45	6.37	3.17	0.59	0.29	0.26	0.14	1.94	1.31	0.83	0.63	0.50	0.13
50	10.30	5.83	1.38	1.15	0.93	0.78	2.88	2.22	1.00	0.69	0.55	0.11
55	12.25	8.78	0.48	0.30	0.56	0.30	2.48	0.96	0.92	0.53	0.29	0.00
60	14.49	9.94	0.36	0.13	0.46	0.23	1.78	0.92	1.05	0.46	0.28	0.05
65	18.32	12.57	0.48	0.31	0.40	0.24	1.59	0.39	0.99	0.26	0.40	0.04
70	20.22	14.34	0.48	0.17	0.31	0.17	0.60	0.25	0.33	0.11	0.39	0.20
75	22.41	12.49	0.35	0.14	0.25	0.12	1.16	0.63	0.35	0.12	0.51	0.31
80	24.52	17.06	0.87	0.58	0.59	0.37	1.74	1.29	1.10	0.76	1.26	0.71
85	24.54	18.03	1.10	0.73	0.97	0.73	1.38	1.04	0.96	0.64	1.06	0.73
90	27.73	23.45	1.06	0.64	0.80	0.61	1.23	0.27	0.71	0.25	0.58	0.37
95	26.99	20.49	0.90	0.55	0.62	0.34	1.40	0.96	0.74	0.51	0.55	0.28
100	30.79	25.82	0.47	0.11	0.42	0.11	1.32	0.64	0.46	0.15	0.49	0.14
Av.	8.73	5.79	0.30	0.17	0.24	0.14	0.82	0.46	0.34	0.17	0.25	0.10

Table 10. New heuristics and the 654-customer problem.

4. Comparing TS-2 with VNS-2 and TS-3 with VNS-3, we observe that better results are obtained by the variable neighbourhood approach. However, TS-2 is competitive in Tables 10 and 11, where it outperforms VNS-2 and VNS-3 in a few cases.

5. VNS-3 is clearly the best overall method in Tables 8 and 9, but VNS-4 takes over in Tables 10 and 11. The advantage with VNS-4 pertains to its p-Median neighbourhood structure. Since the facilities are kept at fixed point locations, VNS-4 is able to evaluate neighbourhood points extremely quickly. Only when a descent move is made, does the algorithm locate the facilities in continuous space to obtain a candidate solution. The greater number of visits through the neighbourhoods with VNS-4 appears to be a critical factor in large problems.

	GA		DA-2				CH				P-MED	
			TS-2		VNS-2		TS-3		VNS-3		VNS-4	
m	*Av.*	*Best*	*Av.*	*Best*	*Av.*	*Best*	*Av.*	*Best*	*Av.*	*Best*	*Av.*	*Best*
5	0.00	0.00	0.98	0.87	0.62	0.26	0.00	0.00	0.02	0.00	0.00	0.00
10	0.03	0.00	0.55	0.45	0.60	0.33	0.04	0.00	0.01	0.00	0.01	0.00
15	0.06	0.01	0.21	0.18	0.31	0.17	0.20	0.01	0.03	0.00	0.04	0.01
20	0.25	0.08	0.17	0.05	0.31	0.14	0.78	0.53	0.11	0.01	0.06	0.00
25	0.26	0.09	0.08	0.04	0.07	0.03	0.45	0.02	0.09	0.01	0.17	0.02
30	0.65	0.04	0.37	0.31	0.30	0.18	0.79	0.22	0.15	0.03	0.06	0.00
35	1.04	0.35	0.19	0.02	0.15	0.02	1.89	1.15	0.46	0.11	0.09	0.00
40	1.36	0.77	0.12	0.01	0.16	0.06	2.11	1.30	0.67	0.38	0.10	0.00
45	1.56	0.69	0.13	0.02	0.17	0.00	1.52	0.94	0.45	0.18	0.19	0.01
50	2.48	1.52	0.22	0.05	0.16	0.02	1.77	1.29	0.66	0.45	0.21	0.08
55	1.97	0.69	0.13	0.02	0.10	0.07	0.91	0.21	0.16	0.04	0.21	0.00
60	2.42	0.83	0.15	0.02	0.10	0.03	0.92	0.51	0.21	0.06	0.07	0.00
65	2.19	0.96	0.29	0.09	0.13	0.06	0.96	0.51	0.15	0.01	0.17	0.00
70	3.49	2.05	0.24	0.05	0.15	0.06	0.95	0.57	0.29	0.11	0.27	0.02
75	3.72	1.89	0.19	0.00	0.13	0.03	0.84	0.61	0.29	0.01	0.20	0.00
80	4.35	3.21	0.37	0.06	0.13	0.03	0.71	0.21	0.22	0.01	0.28	0.00
85	4.18	1.86	0.30	0.13	0.13	0.00	0.65	0.25	0.19	0.10	0.24	0.07
90	4.24	3.14	0.30	0.15	0.12	0.04	0.87	0.58	0.41	0.16	0.31	0.01
95	4.49	3.49	0.23	0.06	0.13	0.00	0.77	0.55	0.33	0.13	0.23	0.01
100	4.71	3.83	0.42	0.11	0.30	0.10	0.92	0.47	0.47	0.28	0.33	0.07
Av.	2.17	1.27	0.28	0.13	0.21	0.08	0.90	0.49	0.26	0.10	0.16	0.02

Table 11. New heuristics and the 1060-customer problem.

5 Conclusions

An extensive empirical study is presented of heuristic methods for solving the multi-source Weber problem. Included are several new methods which have not been reported previously. An important aspect of the current work is that it considers much larger problem sizes than previously investigated in the literature. Thus, we are able to show that the state-of-the-art heuristics tend to deteriorate in performance with increasing problem size, sometimes in a disastrous fashion.

The new heuristics presented here obtained excellent results, which are far superior to the solutions found by the existing methods. Deviations from the best known solutions of less than 0.1% are consistently reported by the new methods over all problem sets. The fact is made more remarkable in view of the restricted CPU time. Thus, we may claim that the state-of-the art is advanced.

Some general conclusions are inferred from the results of this study:

1. Relocation-based methods (drop/add or interchange) are more efficient than their counterpart reallocation-based methods. That is, better solutions are obtained in general in the same CPU time, when local or variable neighbourhood searches are conducted with a relocation neighbourhood structure. One reason may be the fact that the neighbourhood points in the relocation structure correspond to Voronoi partitions of the customer set, but not so for reallocations.

2. The variable neighbourhood concept can be effectively used to obtain superior solutions. We may view the variable neighbourhood search (VNS) as a 'shaking' process, where movement to a successive neighbourhood corresponds to a harder shake. Unlike random restart, which moves from the current solution to any point (uncontrolled shaking) typically far away, VNS allows a controlled increase in the level of the shake.

3. Comparison of the new heuristics suggests that no one method is best in all cases. Issues to consider in the design of an algorithm include the type of neighbourhood (e.g., drop/add or interchange, discrete or continuous facility locations), the amount of shaking to permit, when to conduct local searches at neighbourhood points and by what method, and whether or not to permit ascent moves. The variation of strategies is limitless in terms of shaking and local search, and parameter settings. Future studies may establish general guidelines for choosing the 'best' algorithm as a function of problem size (m and n).

Acknowledgements: Research of the first author was supported by The Department of National Defence (Canada) Academic Research Program. Research of the second and third authors was supported by Office of Naval Research Grant N00014-92-J-1194, Natural Sciences and Engineering Research Council of Canada Grant GPO 105574 and Fonds pour la Formation des Chercheurs et l'Aide à la Recherche Grant 32EQ 1048. Research of the fourth author was supported by an International Post-doctoral Fellowship of Natural Sciences and Engineering Research Council of Canada, Grant OGPOO 39682 and by the Swiss National Science Foundation project number 21-45653.95. The authors thank Paul Calamai for making the program for his projection method written with Ingrid Bongartz and Andrew Conn, available to them, and Dominique Peeters for communicating his program for the p-Median problem, written

with Pierre Hanjoul.

References

[1] Avriel, M. 1976. *Nonlinear Programming: Analysis and Methods.* Prentice-Hall, Englewood Cliffs, New Jersey.

[2] Baxter, J. 1981. Local Optima Avoidance in Depot Location. *Journal of the Operational Research Society* **32**, 815-819.

[3] Boese, K.D., A.B. Kahng and S. Muddu. 1994. A New Adaptive Multi-Start Technique for Combinatorial Global Optimizations. *Operations Research Letters* **16**, 101-113.

[4] Bongartz, I., P.H. Calamai and A.R. Conn. 1994. A Projection Method for l_p Norm Location-Allocation Problems. *Mathematical Programming* **66**, 283-312.

[5] Brimberg, J., R. Chen and D. Chen. 1996. Accelerating Convergence in the Fermat-Weber Location Problem. (submitted for publication).

[6] Brimberg, J., and R.F. Love. 1993. Global Convergence of a Generalized Iterative Procedure for the Minisum Location Problem with l_p Distances. *Operations Research* **41**, 1153-1163.

[7] Brimberg, J., and N. Mladenović. 1996a. Solving the Continuous Location-Allocation Problem with Tabu Search. *Studies in Locational Analysis* **8**, 23-32.

[8] Brimberg, J., and N. Mladenović. 1996b. A Variable Neighbourhood Algorithm for Solving the Continuous Location-Allocation Problem. *Studies in Locational Analysis* **10**, 1-12.

[9] Brimberg, J., P. Hansen, N. Mladenović and É. Taillard. 1997. Improvements and Comparison of Heuristics for Solving the Multisource Weber Problem. *Les Cahiers du GERAD*, Montreal, Canada (forthcoming).

[10] Charalambous, C., and J.W. Bandler. 1976. Non-linear Optimization as a Sequence of Least p-th Optimization with Finite Values of p. *International Journal of System Science* **7**, 377-391.

[11] Chen, R. 1983. Solution of Minisum and Minimax Location-Allocation Problems with Euclidean Distances. *Naval Research Logistics Quarterly* **30**, 449-459.

[12] Chen, P.C., P. Hansen, B. Jaumard and H. Tuy. 1992. Solution of the Multisource Weber and Conditional Weber Problems by d.-c. Programming. *Les Cahiers du GERAD*, G-92-35, Montreal, Canada (to appear in *Operations Research*).

[13] Cooper, L. 1963. Location - Allocation Problems. *Operations Research* **11**, 331-343.

[14] Cooper, L. 1964. Heuristic Methods for Location - Allocation Problems. *SIAM Review* **6**, 37-53.

[15] Cooper, L. 1967. Solutions of Generalized Locational Equilibrium Models. *Journal of Regional Science* **7**, 1-18.

[16] Drezner, Z. 1984. The Planar Two-Center and Two-Median Problems. *Transportation Science* **18**, 351-361.

[17] Drezner, Z. 1992. A Note on the Weber Location Problem. *Annals of Operations Research* **40**, 153-161.

[18] Eilon, S., C.D.T. Watson - Gandy and N. Christofides. 1971. *Distribution Management.* Hafner, New York.

[19] Frenk, J.B.G., M.T. Melo and S. Zhang. 1994. A Weiszfeld Method for a Generalized l_p Distance Minisum Location Model in Continuous Space. *Location Science* **2**, 111-127.

[20] Gendreau, M., A. Hertz and G. Laporte. 1996. The Traveling Salesman Problem with Backhauls. *Computers and Operations Research* **23**, 501-508.

[21] Glover, F. 1989. Tabu Search. Part I. *ORSA J. Computing* **1**, 190-206.

[22] Glover, F. 1990. Tabu Search. Part II. *ORSA J. Computing* **2**, 4-32.

[23] Glover, F., and M. Laguna. 1993. Tabu Search, in C. Reeves ed., *Modern Heuristic Techniques for Combinatorial Problems* (Chapter 3), Oxford, Blackwell.

[24] 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.

[25] Hansen, P., and B. Jaumard. 1990. Algorithms for the Maximum Satisfiability Problem. *Computing* **44**, 279-303.

[26] Hansen, P., N. Mladenović and É. Taillard. 1996. Heuristic Solution of the Multisource Weber Problem as a p-Median Problem. *Les Cahiers du GERAD*, G-96-10, Montreal, Canada (to appear in *Operations Research Letters*).

[27] Hansen, P., B. Jaumard, S. Krau and O. du Merle. 1997. A Column Generation Algorithm for the Multisource Weber Problem. *Les Cahiers du GERAD* (1997) (forthcoming).

[28] Holland, J.H. 1975. *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor, Michigan.

[29] Houck, C.R., J.A. Joines and M.G. Kay. 1996. Comparison of Genetic Algorithms, Random Restart and Two-Opt Switching for Solving Large Location-Allocation Problems. *Computers and Operations Research* **23**, 587-596.

[30] Krau, S. 1997. *Extensions du problems de Weber*, Ph D. Thèse, École Polytechnique de Montréal (under direction of P. Hansen and B. Jaumard).

[31] Kuenne, R. E., and R. M. Soland. 1972. Exact and Approximate Solutions to the Multisource Weber Problem. *Mathematical Programming* **3**, 193-209.

[32] Kuhn, H.W. 1973. A Note on Fermat's Problem. *Mathematical Programming* **4**, 98-107.

[33] Love, R.F., and H. Juel. 1982. Properties and Solution Methods for Large Location - Allocation Problems. *Journal of the Operational Research Society* **33**, 443-452.

[34] Love, R.F., and J.G. Morris. 1975. A Computational Procedure for the Exact Solution of Location - Allocation Problems with Rectangular Distances. *Naval Research Logistics Quarterly* **22**, 441-453.

[35] Love, R.F., J.G. Morris and G.O. Wesolowsky. 1988. *Facilities Location: Models and Methods.* New York, North Holland.

[36] Megiddo, N., and K.J. Supowit. 1984. On the Complexity of Some Common Geometric Location Problems. *SIAM Journal on Computing* **13**, 182-196.

[37] du Merle, O., D. Villeneuve, J. Desrosiers and P. Hansen. 1996. Stabilization dans le cadre de la génération de colonnes, *Les Cahiers du GERAD*, Montreal (forthcomming).

[38] Mirchandani, P., and R. Francis (eds.). 1990. *Discrete Location Theory.* A Wiley-Interscience Publication.

[39] Mladenović, N. 1995. A Variable Neighbourhood Algorithm - a New Metaheuristic for Combinatorial Optimization. Presented at Optimization Days, Montreal.

[40] Mladenović, N., and J. Brimberg. 1995. A Descent-Ascent Technique for Solving the Multisource Weber Problem. *Yugoslav Journal of Operations Research* **5**, 211-219.

[41] Mladenović, N., and J. Brimberg. 1996. A Degeneracy Property in Continuous Location-Allocation Problems. *Les Cahiers du GERAD*, G-96-37, Montreal.

[42] Mladenović, N., and P. Hansen. 1996. Variable Neighbourhood Search. *Les Cahiers du GERAD*, G-96-49 (to appear in *Computers and Operations Research*).

[43] Moreno, J., C. Rodrigez and N. Jimenez. 1990. Heuristic Cluster Algorithm for Multiple Facility Location-Allocation Problem, RAIRO -*Operations Research* **25**, 97-107.

[44] 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.

[45] Ostresh, L.M.,Jr. 1973a. TWAIN – Exact Solutions to the Two Source Location-Allocation Problem. in: G. Rushton, M.F. Goodchild and L.M. Ostresh Jr. (eds.), *Computer Programs for Location-Allocation Problems*, Monograph Number 6, Department of Geography, University of Iowa, Iowa City, IA, 15-28.

[46] Ostresh, L.M.,Jr. 1973b. MULTI – Exact Solutions to the M-Center Location-Allocation Problem. in: G. Rushton, M.F. Goodchild and L.M.Jr. Ostresh (eds.), *Computer Programs for Location-Allocation Problems*, Monograph Number 6, Department of Geography, University of Iowa, Iowa City, 29-53.

[47] Ostresh, L.M.,Jr. 1975. An Efficient Algorithm for Solving the Two Center Location-Allocation Problem. *Journal of Regional Science* **15**, 209-216.

[48] Reinelt, G. 1991. TSLIB-A Traveling Salesman Library. *ORSA J. on Computing* **3**, 376-384.

[49] Rosen, J. B., and G.-L. Xue. 1991. Computational Comparison of Two Algorithms for the Euclidean Single Facility Location Problem. *ORSA Journal on Computing* **3**, 207-212.

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

[51] Scott, A.J. 1970. Location-Allocation Systems: A Review. *Geographical Analysis* **2**, 95-119.

[52] Sullivan, P.J., and N. Peters. 1980. A Flexible User Oriented Location-Allocation Algorithm. *Journal of Environmental Management* **10**, 181-193.

[53] Whitaker, R. 1983. A Fast Algorithm for the Greedy Interchange for Large-Scale Clustering and Median Location Problems. *INFOR* **21**, 95-108.