Ann Oper Res (2008) 157: 73–89
DOI 10.1007/s10479-007-0193-1

Scatter search for the single source capacitated facility location problem

Iván A. Contreras · Juan A. Díaz

Published online: 1 August 2007

Abstract This paper considers the Single Source Capacitated Facility Location Problem (SSCFLP). We propose a Scatter Search approach to provide upper bounds for the optimal solution of the problem. The proposed approach uses GRASP to initialize the Reference Set. Solutions of the Reference Set are combined using a procedure that consists of two phases: (1) the initialization phase and (2) the improvement phase. During the initialization phase each client is assigned to an open facility to obtain a solution that is then improved with the improvement phase. Also, a tabu search algorithm is applied. In order to evaluate the proposed approach we use different sets of test problems. According to the results obtained we observe that the method provides good quality solutions with reasonable computational effort.

Keywords Discrete location · Scatter search · Tabu search · GRASP

1 Introduction

Discrete location problems consider a wide range of situations where the decision to be taken is to locate some facilities in an optimal way with respect to some objective (see for example Mirchandani and Francis 1990). The facilities are selected from a discrete set of possible locations, in relation to some clients to be served by the facilities. A family of problems within discrete location problems is the so-called Facility Location Problems (FLP's). The FLP's differ from each other according to different assumptions on the way clients are served or on the existence of limited capacity on the facilities. When capacities on the facilities are considered, it is most common to allow splitting the client's demands. However, we can find other applications where the demand of each client is restricted to be

I.A. Contreras · J.A. Díaz (✉)
Departamento de Ingeniería Industrial y Mecánica, Universidad de las Américas, Puebla, Sta. Catarina Mártir, Cholula 72820, Puebla, Mexico
e-mail: juana.diaz@udlap.mx

I.A. Contreras
e-mail: ivan.contreras@upc.edu

supplied from a single facility. This gives rise to the so-called Single Source Capacitated Facility Location Problem (SSCFLP).

The Single Source Capacitated Facility Location Problem is known to be an NP-*hard* optimization problem. Different approaches to obtain upper and lower bounds for SSCFLP are proposed in the literature. Most of them are Lagrangean relaxations that try to improve the lower bounds obtained with the linear programming relaxation. Different kinds of primal heuristics are often incorporated to obtain upper bounds (see Barceló and Casanovas 1984; Klincewics and Luss 1986; Beasley 1993; Pirkul 1987; Barceló et al. 1990, 1991; Sridharan 1993; Hindi and Pieńkosz 1999; Holmberg et al. 1999; Darby-Dowman and Lewis 1988, Cortinhal and Captivo 2003). Also, metaheuristic methods have been proposed to provide upper bounds. Different heuristic methods are proposed in Delmaire et al. (1999a): evolutive algorithms, tabu search, simulated annealing and GRASP. Delmaire et al. (1999a) improve the methods proposed in Delmaire et al. (1999a) and consider reactive hybrid methods that combine GRASP and tabu search. Rönnqvist et al. (1999) propose a heuristic algorithm based on repeated matching. Cortinhal and Captivo (2003) obtain lower bounds using a Lagrangean relaxation approach where the assignment constraints are dualized. Upper bounds are obtained with a primal heuristic that uses the solution of the relaxed problem to provide a feasible solution that is then improved using tabu search. Recently, Ahuja et al. (2004) propose a very large scale neighborhood search algorithm where the neighborhood structures are induced by customer multi-exchanges and by facility moves. The multi-exchanges are detected using improvement graphs that are dynamically built using a greedy scheme.

SSCFLP has also been tackled by exact methods. Neebe and Rao (1983) reformulate the problem as a Set Partitioning Problem with additional constraints where integrality constraints are relaxed to produce a linear program. At each iteration of the procedure a new column is brought into the basis in a column generation fashion. This column generation algorithm is embedded within a Branch & Bound algorithm. Holmberg et al. (1999) consider a Lagrangean relaxation where the assignment constraints are dualized. Then, this Lagrangean relaxation is embedded within a Branch & Bound algorithm. They use a depth first search strategy by branching on y-variables. When all y-variables are fixed, branching is performed on x-variables. Díaz and Fernández (2002) propose a Branch & Price approach. The procedure uses column generation for finding upper and lower bounds. The bounding procedure exploits the structure of the problem using an iterative approach. At each iteration of the proposed method a two-level optimization problem is considered. The first level deals with the selection of the facilities to be opened whereas the second level deals with the allocation of clients within the set of open facilities.

In this work we consider the SSCFLP and propose a Scatter Search algorithm. Scatter Search has been successfully implemented for a wide range of combinatorial optimization problems (see for example Martí 2006), but, to the best of our knowledge it has not been considered so far for SSCFLP. The implementation of the algorithm is based on the template proposed by Laguna and Martí (2003). An initial set of trial solutions is generated using GRASP. Then, a *Reference Set* of solutions is constructed by selecting a collection of both high quality solutions and diverse solutions from the set of trial solutions. Solutions of the *Reference Set* are combined to provide new solutions. Also, a tabu search heuristic is used to improve the combined solutions.

The most important feature of the proposed method is the *Solution Combination Method*. This procedure incorporates an effective balance between intensification and diversification strategies. Diversification is achieved by considering different sizes of the set of open facilities while intensification is achieved by identifying promising areas of the solution space.

Computational experiments have been performed on different sets of test problems to evaluate the performance of the proposed method. According to the obtained results the proposed method provide good quality solutions with reasonable computer effort.

This work is structured as follows: Sect. 2 introduces some notation and terminology and presents a formulation of the problem. Section 3 describes the proposed algorithm. Computational experiments are described in Sect. 4. Finally, Sect. 5 presents some conclusions and remarks.

2 Problem formulation

Let $I = \{1, \ldots, m\}$ be the set of potential sites to locate the facilities and $J = \{1, \ldots, n\}$ the set of clients. Also, let f_i, be the fixed cost for opening a facility in site $i \in I$, b_i the capacity of the facility located at $i \in I$ expressed in demand units, c_{ij} the cost of supplying the demand of client $j \in J$ from a facility established at site $i \in I$ and d_j the demand of client $j \in J$. The objective is to locate a number of facilities and to fully assign each client to one of the chosen facilities at minimum cost. Let

$$y_i = \begin{cases} 1, & \text{if site } i \in I \text{ is selected to locate a facility,} \\ 0, & \text{otherwise,} \end{cases}$$

and

$$x_{ij} = \begin{cases} 1, & \text{if client } j \in J \text{ is assigned to a facility located at site } i \in I, \\ 0, & \text{otherwise,} \end{cases}$$

denote the selection and assignment decision variables, respectively. The SSCFLP can be formulated as follows:

$$\text{(SSCFLP)} \quad \min \quad \sum_{i\in I}\sum_{j\in J} c_{ij}x_{ij} + \sum_{i\in I} f_i y_i \tag{1}$$

$$\text{s.t.} \quad \sum_{i\in I} x_{ij} = 1, \quad \forall j \in J, \tag{2}$$

$$\sum_{j\in J} d_j x_{ij} \le b_i y_i, \quad \forall i \in I, \tag{3}$$

$$y_i \in \{0,1\}, \quad \forall i \in I, \tag{4}$$

$$x_{ij} \in \{0,1\}, \quad \forall i \in I,\ \forall j \in J. \tag{5}$$

Constraints (2) together with integrality constraints (5) ensure that each client is assigned to a single facility. Constraints (3) ensure that no client is assigned to a site where no facility is opened, and also enforce the overall demand of clients assigned to a facility not to exceed its capacity.

As in Delmaire et al. (1999b) we consider the following representation of the objective function

$$\min \quad K + \sum_{i\in I}\sum_{j\in J} \Delta_{ij}x_{ij} + \sum_{i\in I} f_i y_i, \tag{6}$$

where $c_{ij} = \Delta_{ij} + c\min_j, \forall i \in I, j \in J$, $c\min_j = \min\{c_{ij} : i \in I\}$ and $K = \sum_{j\in J} c\min_j$. Here, $c\min_j$ denotes the best assignment for client j, K is a fixed assignment cost that has

to be incurred and Δ_{ij} represents the increment from the best assignment for serving client j from a facility located at site i. The Δ_{ij}'s allow fair comparisons when selecting the set of clients to be assigned to an open facility, particularly when the assignment costs ranges vary among clients.

In what follows, solutions are represented by pairs of the form $s = (S, a)$ where $S \subseteq I$ denotes the set of selected sites to locate the facilities and $a : J \to I$ is the assigning mapping, i.e. $a(j) = p$ if client $j \in J$ is assigned to facility $p \in S$. For any feasible assignment, h_i denotes the available capacity of facility i (i.e. $h_i = b_i - \sum_{j:a(j)=i} d_j$).

3 Scatter search

Scatter search was initially introduced by Glover (1977). It is an evolutionary method that operates on a set of solutions, called the *Reference Set*, by combining these solutions to create new ones. A detailed explanation of this methodology can be found in Glover (1997, 1999) and in Laguna and Martí (2003). The scatter search for the SSCFLP is implemented using the template proposed in Laguna and Martí (2003). It consists of five methods: (1) *Diversification Generation Method*, (2) *Improvement Method*, (3) *Reference Set Update Method*, (4) *Subset Generation Method* and (5) *Solution Combination Method.*

3.1 Diversification generation method

The goal of the diversification generation method is to provide a collection of diverse trial solutions. In our proposal this method is implemented using GRASP. The GRASP methodology was first introduced by Feo and Resende (1995). It is an iterative method that consists of two phases. The *Construction Phase* builds a solution from scratch by partially randomizing a greedy procedure while the *Local Search Phase* improves the solution obtained in the first stage using a local search procedure. Now we describe the GRASP algorithm to construct the initial set P of trial solutions.

3.1.1 Construction phase

At each step of the *Construction Phase* one potential site is selected and several clients are assigned to it. Let $s = (S, a)$ be a partial solution where some clients are not yet allocated (i.e. $a(j) = null$). We consider only feasible assignments with respect to the capacity constraints. Clients are ordered by increasing values of the Δ_{ij}'s. For each unselected site $i \in I \setminus S$ we compute the same greedy function proposed in Delmaire et al. (1999b),

$$\varphi_i = \frac{f_i + \sum_{i \in Clients_i} \Delta_{ij}}{|Clients_i|},$$

where $Clients_i$ is the list of all unassigned clients that fit into the facility located at i, assuming that clients are assigned to i according to the ordering described above. The φ_i value can be interpreted as the overall cost per client assigned to facility i.

In order to achieve a tradeoff between quality and diversity, a partially randomized greedy procedure is considered. The choice of the next site to open a facility is determined by randomly selecting one element from a *Restricted Candidate List* (*RCL*). The *RCL* is updated at each iteration of the construction phase and contains the best candidates with respect to a threshold value. This threshold value is computed in the following way. Let $\varphi_{min} =$

$\min\{\varphi_i : i \in I \setminus S\}$ and $\varphi_{max} = \max\{\varphi_i : i \in I \setminus S\}$. Then, $RCL = \{i : \varphi_i \leq \varphi_{min} + \alpha(\varphi_{max} - \varphi_{min})\}$. The construction phase terminates when all clients are assigned to an open facility.

Algorithm 1 Construction Phase of GRASP

```
Let A be the the set of assigned clients
S ← ∅
a(j) ← null, ∀j ∈ J
A ← ∅
while (A ≠ J) do
  for all (i ∈ I \ S) do
    φ_i = (f_i + Σ_{j∈Clients_i} Δ_ij) / |Clients_i|
  end for
  φ_min = min{φ_i : i ∈ I \ S}
  φ_max = max{φ_i : i ∈ I \ S}
  RCL = {i : φ_i ≤ φ_min + α(φ_max − φ_min)}
  Select i_s randomly from RCL
  S ← S ∪ {i_s}
  Update a(k), ∀k ∈ Clients_{i_s} and the set A
end while
```

3.1.2 Local search phase

The *Local Search Phase* procedure explores two types of neighborhood structures. The neighborhood structures of the first type are related to the assignment of clients within the set of selected facilities whereas the neighborhoods of the second type are related to solutions where the set of open facilities changes. In all cases, we consider solutions within the feasible domain.

The neighborhoods of the first type are the classical *shift* and *swap* neighborhoods. The shift neighborhood considers all solutions that can be reached from the current one by changing the assignment of exactly one client whereas the swap neighborhood contains all solutions that differ from the current one in the assignment of two clients. Let $s = (S, a)$ be the current solution, then

$$N_{shift}(s) = \{s' = (S, a') : \exists! j \in J, a'(j) \neq a(j)\}$$

and

$$N_{swap}(s) = \{s' = (S, a') : \exists j_1, j_2, a'(j_1) = a(j_2), a'(j_2) = a(j_1), a'(j) = a(j), \forall j \neq j_1, j_2\}.$$

To explore N_{shift} we consider all pairs of the form (i, j) where $a(j) \neq i$ and $h_i \geq d_j$. Also, to explore N_{swap} we consider all pairs of the form (j_1, j_2) where $a(j_1) \neq a(j_2)$, $h_{a(j_1)} + d_{j_1} \geq d_{j_2}$ and $h_{a(j_2)} + d_{j_2} \geq d_{j_1}$.

We explore three additional neighborhood structures of the second type. They affect the current set of open facilities. Two of them consider solutions that are obtained from the current one by interchanging an open facility by a closed one whereas the third one considers a set of solutions that can be obtained by selecting a new site to open a facility. Let $s = (S, a)$ denote the current solution and let $i \in I \setminus S$ be the facility selected to replace the facility located at site $k \in S$, then

$$N_{inter}(s) = \left\{ s' = (S', a') : S' = S \setminus \{k\} \cup \{i\}; \forall j, a'(j) = r \in S' \text{ s.t. } \sum_{j:a'(j)=r} d_j \leq b_r, \forall r \in S' \right\}.$$

However, $N_{inter}(s)$ is too large to be explored completely. Instead, two different subset of $N_{inter}(s)$ are considered. In the first subset, $N^1_{inter}(s) \subset N_{inter}(s)$, all clients assigned to facility k will be reassigned to facility i. When exploring N^1_{inter} we consider all pairs of the form $(k,i), k \in S, i \in I \setminus S$ where $b_i \geq \sum_{j:a(k)=j} d_j$.

In the second subset, $N^2_{inter}(s) \subset N_{inter}(s)$, the clients assigned to facility k can be reassigned within the set $S \setminus \{k\} \cup \{i\}$. We use the following procedure to explore $N^2_{inter}(s)$. All pairs of the form $(k,i), k \in S, i \in I \setminus S$ are considered. Let $s = (S,a)$ denote the current solution and let $a'(i)$ be the new assignment. Initially, $a'(p) = null$ for all $p \in \{j : a(j) = k\}$, $a'(p) = a(p)$ for all $p \in \{j : a(j) \neq k\}$ and $\hat{h}_r = h_r$ for all $r \in S \setminus \{k\} \cup \{i\}$. For each potential site to locate a facility (i.e. $i \in I \setminus S$), we consider clients in turn by increasing values of the Δ_{ij}'s. Client j is reassigned to facility i if $\Delta_{ij} \leq \Delta_{a(j),j}$ and $\hat{h}_i \geq d_j$. If we reassign client j to facility i (i.e. $a'(j)$ is set to i), we also update the available capacity of facilities i and $a(j)$. When this procedure terminates, some clients will remain unassigned. Then, we consider such clients by decreasing values of their demand, and assign them to the open facility with minimum assignment cost among the ones with enough available capacity. That is, if $a'(j) = null$, then $a'(j) \in \arg\min\{\Delta_{rj} : r \in S \setminus \{k\} \cup \{i\},\ \text{s.t. } \hat{h}_r \geq d_j, \}$. Again, after assigning any client the available capacity of its facility is also updated. If one or more unassigned clients cannot be assigned within $S \setminus \{k\} \cup \{i\}$ (in a feasible way with respect to the capacity constraints), then the pair $(k,i), k \in S, i \in I \setminus S$ is not considered as a candidate neighbor.

Finally, the third neighborhood structure of the second type, considers a subset of feasible solutions that are obtained from the current one by opening a new facility and by assigning some clients to it. Then,

$$N_{open}(s) \subset \left\{ s' = (S', a') : S' = S \cup \{i\}; \forall j, a'(j) = r \in S' \text{ s.t. } \sum_{j:a'(j)=r} d_j \leq b_r, \forall r \in S' \right\}$$

To explore $N_{open}(s)$ all facilities $i \in I \setminus S$ are considered. Again, let $s = (S,a)$ denote the current solution, a' the new assignment and $\hat{h}_r$ the available capacity of facility r. Initially, $a'(p) = a(p)$ for all $p \in J$ and $\hat{h}_r = h_r$ for all $r \in S$. For each potential site (i.e. $i \in I \setminus S$), we consider clients by increasing order of their Δ_{ij} values. As in neighborhood $N^2_{inter}(s)$, client j is reassigned to facility i if $\Delta_{ij} \leq \Delta_{a(j),j}$ and $\hat{h}_i \geq d_j$. If client j is reassigned to facility i we update its assignment ($a'(j)$ is set to i) and the available capacity of facilities i and $a(j)$.

The *Local Search Phase* of the GRASP algorithm is summarized in Algorithm 2.

Algorithm 2 Local Search Phase of GRASP

```
Terminate ← false
while (not Terminate) do
  repeat
    Explore N_shift
  until (no improvement move is found)
  Explore N_swap
  if (no improvement move is found) then
    Explore N^1_inter
    if (no improvement move is found) then
      Explore N^2_inter
      if (no improvement move is found) then
        Explore N_open
      else
        Terminate ← true
      end if
    end if
  end if
end while
```

3.2 Improvement method

An improvement method can be applied at different stages of scatter search in order to enhance trial solutions. In this work, we apply the same local search algorithm used in the *Local Search Phase* of the GRASP algorithm detailed in Sect. 3.1.2 to each combined solution. Additionally, a tabu search algorithm is applied to the best solution found by the scatter search algorithm. This procedure deals with the assignment of clients to facilities. At each stage of the tabu search procedure the best admissible move (with respect to the objective function value (6)) in $N_{shift}(s) \cup N_{swap}(s)$ is performed. We use tabu restrictions to prevent solutions visited in the recent past to be revisited. To keep track of forbidden attributes we consider ordered pairs of the form (i, j). When one of such pairs is labeled *tabu-active*, the assignment of client j to facility i is forbidden. That is, if client j assigned to facility $i_1 \in S$ is reassigned to facility $i_2 \in S$ during iteration l, the attribute (i_1, j) will be forbidden during the next t iterations, where t is the *tabu-tenure* parameter. *Tabu-active* attributes for a move are overridden whenever they satisfy an aspiration criterion. At any stage of the search process, the tabu status for a move is ignored if it leads the search to the best feasible solution found so far.

3.3 Reference set update method

The *Reference Set* (*RefSet*) is a collection of both high quality solutions and diverse solutions that are used to generate new solutions by applying the *Solution Combination Method*. The purpose of the *Reference Set Update Method* is to build and maintain a reference set of solutions. The size of the reference set is typically small (no more than 20 solutions). We initialize the *RefSet* with the $|RefSet|$ best solutions found with the *Diversification Generation Method* associated with different sets of open facilities. The *RefSet* is updated each time the *Solution Combination Method* terminates. When the *Solution Combination Method* is applied, new solutions are generated and added to a *Pool* of solutions. We use the so-called static update. That is, when the *Solution Combination Method* terminates, again the $|RefSet|$ best solutions of $RefSet \cup Pool$ associated with different sets of open facilities are selected. When the *Reference Set* remains unchanged after applying the *Reference Set Update Method*, the scatter search procedure terminates.

3.4 Subset generation method

The purpose of the *Subset Generation Method* is to generate subsets of reference solutions to be combined by the *Solution Combination Method*. In our approach we consider all subsets of size 2 and 3.

3.5 Solution combination method

Our scatter search algorithm is an iterative procedure. At each iteration a different reference set of solutions is considered. Let $S = \{s_1 = (S_1, a_1), \ldots, s_k = (S_k, a_k)\}$ be the current subset of solutions to be combined and let s^c be the combined solution. The *Solution Combination Method* consists of two phases: (1) the *Initialization Phase* and (2) the *Improvement Phase*. For each subset of solutions three different types of combinations are applied. They differ from each other in the *Initialization Phase*. During this phase, each client $j \in J$ is assigned to one open facility $i \in \bigcup_{r=1}^{k} S_r$. Let $i_1(j)$ be the initial assignment of client j. We use three different criteria to decide the initial assignment of each client. They are:

1. $i_1(j) = \arg\min\{\Delta_{ij} : i \in \bigcup_{r=1}^{k} S_r\}$,
2. $i_1(j) = \arg\min\{\gamma_{ij} = \frac{f_i}{b_i} \times d_j + \Delta_{ij} : i \in \bigcup_{r=1}^{k} S_r\}$, and
3. Is a randomized version of the second criterion. Client j is randomly assigned to one of the three best (according to γ_{ij} value) open facilities.

For each subset of solutions to be combined, we perform three different combinations, one for each criterion.

The improvement phase is an iterative approach. At each iteration two procedures are performed: (1) the *Reassignment Procedure* and (2) the *Elimination Procedure*. Since the *Initialization Phase* does not take into account the capacity constraints, the initial combined solution s^c will probably be infeasible. To recover feasibility we apply the *Reassignment Procedure* in order to obtain (if possible) a solution that does not violate the capacity constraints. For fulfilling the capacity constraints we consider in turn all clients assigned to a violated facility. Let $S_V^c = \{r \in S^c : h_r < 0\}$ denote the set of violated facilities where, as usual, h_r denotes the available capacity of facility r. Clients are ordered by increasing values of their minimum reassignment cost increment within S^c. That is, we order clients $j \in J$ s.t. $a^c(j) \in S_V^c$ by increasing values of $\min\{c_{ij} - c_{a(j),j} : i \in S^c\}$. Then, we select in turn each client $\hat{j}$ and evaluate the following candidate moves: (1) all reassignments of $\hat{j}$ within S^c that reduce the overall violated capacity and (2) all interchanges of $\hat{j}$ and another client assigned to a different facility that reduce the overall capacity violation. We randomly select a candidate move among the ones with smaller increase in cost of the resulting solution. When a feasible solution is found, we try to improve its cost by performing a local search in $N_{shift} \cup N_{swap}$.

If a feasible solution is found at termination of the *Reassignment Procedure*, the *Elimination Procedure* tries to close one open facility. Here we consider the set of facilities that can be closed without violating the aggregated demand constraint (i.e. $C = \{r \in S^c : \sum_{j \in J} d_j \leq \sum_{i \in S^c} b_i - b_r\}$). The facility to be closed is randomly selected from a restricted candidate list that is constructed in the following way. Let $h_{max} = \max\{h_r : r \in S^c\}$ and $h_{\min} = \min\{h_r : r \in S^c\}$. Then, the restricted candidate list is constructed in the following way: $RC = \{r : h_r \geq h_{\min} + (1 - \beta) \times (h_{\max} - h_{\min})\}$, where $0 \leq \beta \leq 1$. The *Improvement Procedure* terminates when $C = \emptyset$. After closing an open facility, each client previously assigned to it is reassigned to the best facility (with respect to Δ_{ij} values) within the remaining set of open facilities. This may cause to violate again some capacity constraints. Feasibility is recovered by reapplying the *Reassignment Procedure*.

4 Computational experience

The scatter search algorithm has been coded in C language and run on a PC with a Athlon/2.1 Ghz processor and 256 Mb RAM.

An important feature of SSCFLP instances is the relationship among fixed opening costs and assignment costs. Intuitively, when fixed opening costs dominate assignment costs one may expect optimal solutions to have a small set of open facilities and small slacks for the capacity constraints. On the contrary, when assignment costs are dominating, higher number of open facilities as well as larger capacity slacks can be expected. To illustrate the behavior of our approach in both cases, we considered two sets of instances in the computational experiences. In the first set assignment costs dominate fixed opening costs for most of the instances, whereas in the second set we have the opposite situation. The first set of benchmark instances contains the 71 problems used in Holmberg et al. (1999) and in Ahuja et al.

(2004). The instances of this set are divided in 4 classes, G_1–G_4, according to the dimension of the problems. The second set of problems contains the 57 benchmark instances considered in Delmaire et al. (1999b). The instances of this set are divided in 7 classes, G_5–G_{11}, according to the dimension of the problems.

The values of the different parameters used in the proposed approach are the following: the GRASP algorithm performs 150 iterations starting with $\alpha = 0.1$. In order to generate diverse solutions, every 5 iterations α is incremented in 0.03. The maximum value for the size of the *Reference Set*, $|RefSet|$, is 10. As in Cortinhal and Captivo (2003), the $Tabu\_tenure$ parameter for the tabu search procedure is set to $7 + n \times number/all\_number$, where *number* is the number of times the assignment was made and all_number is the maximum value for $number$, for all assignments already made. Also, the value of the β parameter used in the *Solution Combination Method* is set to 0.35.

In order to evaluate the proposed method we use three different measures: (1) the value of the best solution found, (2) its robustness, measured in terms of the percent average gap with respect to the optimal solution and, (3) its efficiency, measured in terms of the required computational effort.

The results obtained are shown in Tables 1–11. Each instance has been run 20 times. The contents of the columns are the following: Column *Opt* gives the values of the optimal solutions for each instance. The optimal solutions are reported in Ahuja et al. (2004) (for the problems of the first set) and in Díaz and Fernández (2002) (for the problems of the second set). Column *Best* shows the best solution values found for each instance. Columns under the heading *Avrg* depict the averages of the best values obtained in each of the 20 runs with the GRASP algorithm used in the *Diversification Generation Method* and with the scatter search approach (SS), respectively. The next three columns under the heading *AvrgGap* show the percent average gaps with respect to the optimal solutions of: (1) the best solution found, (2) the average solution value obtained with GRASP and, (3) the average solution value obtained with SS. Finally, the last column shows the averages of the execution times.

It can be observed, that for the first set of benchmark instances (Tables 1–4), the SS algorithm gives high-quality results. In particular, the algorithm obtains the optimal solution in 68 out of the 71 problems. Also, the algorithm is robust, since the average deviation from optimal solutions ranges from 0.00% to 0.05%, for the different classes of problems of this set. It is also interesting to note that the value of the best solutions found with the *Diversification Generation Method* (The GRASP algorithm) are significantly improved by SS for classes G_2, G_3 and G_4. Finally, the computational effort seems to be small considering the dimension of the problems. The average cpu time for classes G_1, G_2, G_3 and G_4 respectively, are 0.95, 28.99, 4.14 and 54.76 seconds.

Results for the second set of benchmark instances are shown in Tables 5–11. The SS algorithm seems to perform worse on this set of problems. However, it provides near-optimal solutions with small computational times. In 23 out of the 57 problems the optimal solution is found. Also, the average deviation from optimal solutions ranges from 0.22% to 0.74% for the different classes of problems of this set. It is somewhat surprising that the average deviation from the optimal solution for the problems of this set be greater than that for the problems of the first set, since the dimensions of the instances of the second set are small compared to the dimensions of the problems of the first set. This can only be interpreted in terms of the difficulty of the generated problems. Again, there is a significant improvement with respect to the best solution found by the GRASP algorithm used in the *Diversification Generation Method*. With respect to the efficiency measure, the average times needed are extremely small. The average cpu times are 0.32, 0.99, 2.35, 3.47, 6.57, 11.07 and 14.46 for the G_5, G_6, G_7, G_8, G_9, G_{10} and G_{11} classes, respectively.

Table 1 Results for class G_1

Problem	m	n	*Opt*	*Best*	*Avrg*		*Gap*			CPU
					GRASP	*SS*	*Best*	*GRASP*	*SS*	(seconds)
p1	10	50	8848	8848	8848.0	8848.0	0.00	0.00	0.00	0.71
p2	10	50	7913	7913	7913.0	7913.0	0.00	0.00	0.00	0.49
p3	10	50	9314	9314	9314.0	9314.0	0.00	0.00	0.00	0.80
p4	10	50	10714	10714	10714.0	10714.0	0.00	0.00	0.00	0.72
p5	10	50	8838	8838	8838.0	8838.0	0.00	0.00	0.00	0.54
p6	10	50	7777	7777	7777.0	7777.0	0.00	0.00	0.00	0.20
p7	10	50	9488	9488	9488.0	9488.0	0.00	0.00	0.00	0.55
p8	10	50	11088	11088	11088.0	11088.0	0.00	0.00	0.00	0.52
p9	10	50	8462	8462	8464.0	8462.0	0.00	0.02	0.00	1.04
p10	10	50	7617	7617	7617.0	7617.0	0.00	0.00	0.00	0.37
p11	10	50	8932	8932	8933.2	8932.0	0.00	0.01	0.00	1.11
p12	10	50	10132	10132	10134.7	10132.0	0.00	0.03	0.00	1.00
p13	20	50	8252	8252	8252.0	8252.0	0.00	0.00	0.00	1.32
p14	20	50	7137	7137	7137.0	7137.0	0.00	0.00	0.00	1.41
p15	20	50	8808	8808	8816.1	8808.0	0.00	0.09	0.00	1.49
p16	20	50	10408	10408	10422.8	10408.0	0.00	0.14	0.00	1.27
p17	20	50	8227	8227	8227.7	8227.0	0.00	0.01	0.00	0.98
p18	20	50	7125	7125	7125.0	7125.0	0.00	0.00	0.00	1.17
p19	20	50	8886	8886	8886.1	8886.0	0.00	0.00	0.00	1.03
p20	20	50	10486	10486	10486.0	10486.0	0.00	0.00	0.00	0.89
p21	20	50	8068	8068	8068.0	8068.0	0.00	0.00	0.00	1.26
p22	20	50	7092	7092	7092.0	7092.0	0.00	0.00	0.00	1.51
p23	20	50	8746	8746	8746.0	8746.0	0.00	0.00	0.00	1.27
p24	20	50	10273	10273	10273.0	10273.0	0.00	0.00	0.00	1.17
Average							0.00	0.01	0.00	0.95

As it was mentioned before, there is a main difference between the two different sets of instances used to test our approach. In the optimal solution of the problems of the first set, total fixed costs are dominated by total assignment costs (for most of the instances of this set) whereas in the optimal solution of the problems of the second set we have the opposite situation. A priori, this relationship may imply unequal performance of different search strategies. The proposed *Solution Combination Method* is effective in both situations since it always starts with a large set of open facilities that is reduced along the procedure. This permits to explore solutions for different sizes of the set of open facilities to obtain good solutions for instances with different structure.

In order to evaluate the influence of the *Solution Combination Method* on the overall search procedure we perform other 20 runs for each benchmark instance without performing the *Solution Combination Method*. That is, we apply the tabu search procedure to the best solution obtained by GRASP. We also compare the obtained results with other heuristic approaches. In particular, for the first set of instances, we compare our approach to the VLNS algorithm proposed in Ahuja et al. (2004), and for the second set of instances, to the Hybrid algorithm proposed by Delmaire et al. (1999b). The obtained results are shown in

Table 2 Results for class G_2

Problem	*m*	*n*	*Opt*	*Best*	*Avrg*		*Gap*			CPU
					GRASP	*SS*	*Best*	*GRASP*	*SS*	(seconds)
p25	30	150	10630	10630	11754.7	11631.3	0.00	1.07	0.01	31.77
p26	30	150	10771	10771	10778.2	10771.5	0.00	0.07	0.00	29.19
p27	30	150	12322	12322	12387.4	12322.2	0.00	0.53	0.00	34.69
p28	30	150	13722	13722	13941.7	13722.3	0.00	1.60	0.00	35.20
p29	30	150	12371	12377	12481.3	12384.3	0.05	0.89	0.11	48.81
p30	30	150	11331	11331	11396.4	11342.9	0.00	0.58	0.10	44.97
p31	30	150	13331	13331	13449.9	13343.2	0.00	0.89	0.09	44.11
p32	30	150	15331	15337	15483.7	15348.1	0.04	1.00	0.11	49.02
p33	30	150	11629	11629	11651.6	11629.0	0.00	0.19	0.00	23.88
p34	30	150	10632	10632	10634.7	10632.0	0.00	0.02	0.00	20.94
p35	30	150	12232	12232	12235.1	12232.0	0.00	0.03	0.00	20.48
p36	30	150	13832	13832	13838.2	13832.0	0.00	0.04	0.00	20.56
p37	30	150	11258	11258	11258.0	11258.0	0.00	0.00	0.00	15.18
p38	30	150	10551	10551	10551.0	10551.0	0.00	0.00	0.00	11.88
p39	30	150	11824	11824	11824.0	11824.0	0.00	0.00	0.00	15.65
p40	30	150	13024	13024	13024.0	13024.0	0.00	0.00	0.00	17.53
Average							0.01	0.43	0.03	28.99

Table 3 Results for class G_3

Problem	*m*	*n*	*Opt*	*Best*	*Avrg*		*Gap*			CPU
					GRASP	*SS*	*Best*	*GRASP*	*SS*	(seconds)
p41	10	90	6589	6589	6591.0	6589.5	0.00	0.03	0.01	2.92
p42	20	80	5663	5663	5664.1	5663.0	0.00	0.02	0.00	2.80
p43	30	70	5214	5214	5214.0	5214.0	0.00	0.00	0.00	2.03
p44	10	90	7028	7028	7030.3	7028.0	0.00	0.03	0.00	1.24
p45	20	80	6251	6251	6266.1	6251.0	0.00	0.24	0.00	5.19
p46	30	70	5651	5651	5651.0	5651.0	0.00	0.00	0.00	3.36
p47	10	90	6228	6228	6255.6	6255.6	0.00	0.44	0.44	0.65
p48	20	80	5596	5596	5596.0	5596.0	0.00	0.00	0.00	3.99
p49	30	70	5302	5302	5302.0	5302.0	0.00	0.00	0.00	3.63
p50	10	100	8741	8741	8776.7	8750.0	0.00	0.41	0.10	6.83
p51	20	100	7414	7414	7468.8	7415.1	0.00	0.74	0.01	9.15
p52	10	100	9178	9178	9208.9	9189.9	0.00	0.34	0.13	1.15
p53	20	100	8531	8531	8545.1	8531.1	0.00	0.17	0.00	5.83
p54	10	100	8777	8777	8855.2	8782.5	0.00	0.89	0.06	5.86
p55	20	100	7654	7654	7700.7	7656.4	0.00	0.61	0.03	7.50
Average							0.00	0.26	0.05	4.14

Table 4 Results for class G_4

Problem	m	n	*Opt*	*Best*	*Avrg*		*Gap*			CPU
					GRASP	*SS*	*Best*	*GRASP*	*SS*	(seconds)
p56	30	200	21103	21118	21253.9	21123.4	0.07	0.72	0.10	82.96
p57	30	200	26039	26039	26347.2	26055.2	0.00	1.18	0.06	79.43
p58	30	200	37239	37239	37558.2	37254.4	0.00	0.86	0.04	68.36
p59	30	200	27282	27282	27593.2	27302.2	0.00	1.14	0.07	59.74
p60	30	200	20534	20534	20534.0	20534.0	0.00	0.00	0.00	32.01
p61	30	200	24454	24454	24509.0	24454.0	0.00	0.22	0.00	41.34
p62	30	200	32643	32643	32917.4	32650.2	0.00	0.84	0.02	69.74
p63	30	200	25105	25105	25115.3	25105.2	0.00	0.04	0.00	42.38
p64	30	200	20530	20530	20530.0	20530.0	0.00	0.00	0.00	38.46
p65	30	200	24445	24445	24465.1	24445.0	0.00	0.08	0.00	36.89
p66	30	200	31415	31415	31556.8	31423.5	0.00	0.45	0.03	48.49
p67	30	200	24848	24848	24870.2	24850.2	0.00	0.09	0.01	40.26
p68	30	200	20538	20538	20538.0	20538.0	0.00	0.00	0.00	37.51
p69	30	200	24532	24532	24550.4	24532.0	0.00	0.07	0.00	44.92
p70	30	200	32321	32321	32723.0	32338.2	0.00	1.24	0.05	80.91
p71	30	200	25540	25540	25684.0	25566.6	0.00	0.56	0.10	72.80
Average							0.00	0.46	0.03	54.76

Table 5 Results for class G_5

Problem	m	n	*Opt*	*Best*	*Avrg*		*Gap*			CPU
					GRASP	*SS*	*Best*	*GRASP*	*SS*	(seconds)
p72	10	20	2014	2014	2028.6	2014.0	0.00	0.72	0.00	0.29
p73	10	20	4251	4251	4410.2	4264.2	0.00	3.75	0.31	0.35
p74	10	20	6051	6051	6089.3	6055.1	0.00	0.63	0.07	0.40
p75	10	20	7168	7176	7354.3	7219.9	0.11	2.60	0.72	0.32
p76	10	20	4551	4551	4605.3	4559.7	0.00	1.19	0.19	0.32
p77	10	20	2269	2269	2278.4	2269.0	0.00	0.41	0.00	0.27
Average							0.02	1.55	0.22	0.32

Tables 12 and 13. Columns under the heading "Average deviation from optimal" depict the percent average deviation from the optimal solution, for the different classes of instances, with respect to the average of the solution values obtained: (1) by applying the tabu search procedure to the best solution obtained by the GRASP algorithm used as a *Diversification Generation Method* ("GRASP-Tabu"), (2) by the proposed algorithm ("SS"), (3) by the VLNS procedure proposed in Ahuja et al. (2004) ("VLNS") for the first set of instances and by the Hybrid algorithm proposed by Delmaire et al. (1999b) ("HB") for the second set of instances.

As can be observed in Tables 12 and 13, for all different classes of benchmark instances, the percent average deviation is improved by the *Solution Combination Method*. This is es-

Table 6 Results for class G_6

Problem	*m*	*n*	*Opt*	*Best*	*Avrg*		*Gap*			CPU
					GRASP	*SS*	*Best*	*GRASP*	*SS*	(seconds)
p78	15	30	4366	4372	4462.2	4375.3	0.14	2.20	0.21	0.87
p79	15	30	7920	7973	8262.4	8057.3	0.67	4.32	1.73	1.17
p80	15	30	2480	2480	2640.6	2496.6	0.00	6.48	0.67	1.17
p81	15	30	23112	23112	24386.0	23154.3	0.00	5.51	0.18	1.44
p82	15	30	3447	3478	3559.5	3486.6	0.90	3.26	1.15	1.15
p83	15	30	3711	3711	3740.6	3724.2	0.00	0.80	0.36	0.51
p84	15	30	3760	3773	3895.4	3773.6	0.35	3.60	0.36	0.86
p85	15	30	5965	5971	6168.0	6000.5	0.13	3.40	0.60	0.85
p86	15	30	7816	7827	8332.3	7898.0	0.14	6.61	1.05	1.21
p87	15	30	11543	11554	12341.5	11625.8	0.10	6.92	0.72	1.09
p88	15	30	9884	9884	9955.9	9929.1	0.00	0.73	0.46	0.57
Average							0.22	3.98	0.68	0.99

Table 7 Results for class G_7

Problem	*m*	*n*	*Opt*	*Best*	*Avrg*		*Gap*			CPU
					GRASP	*SS*	*Best*	*GRASP*	*SS*	(seconds)
p89	20	30	15607	15615	16448.1	15738.2	0.05	5.39	0.84	3.12
p90	20	30	18683	18683	18901.2	18696.0	0.00	1.17	0.07	2.05
p91	20	30	26561	26600	27880.5	26665.6	0.15	4.97	0.39	3.09
p92	20	30	7295	7318	7682.0	7372.9	0.32	5.30	1.07	3.41
p93	20	30	3271	3271	3418.9	3281.7	0.00	4.52	0.33	1.58
p94	20	30	6036	6036	6318.2	6054.0	0.00	4.68	0.30	1.91
p95	20	30	6327	6327	6421.6	6340.8	0.00	1.50	0.22	1.27
p96	20	30	8947	8947	9027.7	8947.0	0.00	0.90	0.00	0.89
Average							0.06	3.55	0.40	2.35

pecially true for the instances of the second set where more significant improvements are achieved. This shows that the proposed combination method is the most distinctive feature of the proposed algorithm. The combination method permits to explore the infeasible space to obtain more flexibility along the search process. The *Initialization Phase*, together with the *Reassignment Procedure* of the *Improvement Phase*, incorporate flexible rules for identifying promising regions of the solution space. The *Initialization Phase* makes emphasis on quality while the *Reassignment Procedure* makes emphasis on feasibility and local optimality. The purpose of the *Reassignment Procedure* is twofold: (1) it guides the search process to the feasible space and (2) it tries to improve the solution using local search. On the other hand, the *Elimination Procedure* of the *Improvement Phase* is a diversification strategy. These combined strategies permit to achieve an effective interplay between intensification and diversification.

As can be observed in Table 12, with respect to the percentage average deviation from the optimal solution, the proposed algorithm is comparable to the VLNS procedure. How-

Table 8 Results for class G_8

Problem	*m*	*n*	*Opt*	*Best*	*Avrg*		*Gap*			CPU
					GRASP	*SS*	*Best*	*GRASP*	*SS*	(seconds)
p97	20	50	4448	4463	4690.9	4497.2	0.34	5.46	1.10	3.89
p98	20	50	10921	10921	11010.8	10951.1	0.00	0.82	0.28	2.23
p99	20	50	11117	11119	11354.6	11124.0	0.02	2.14	0.06	2.69
p100	20	50	9832	9834	10034.7	9846.5	0.02	2.06	0.15	2.56
p101	20	50	10816	10967	11391.2	11018.0	1.40	5.32	1.87	6.73
p102	20	50	4466	4503	4614.6	4528.0	0.83	3.33	1.39	2.92
p103	20	50	9881	9881	10177.9	9902.1	0.00	3.00	0.21	2.51
p104	20	50	39463	39667	41486.8	39811.5	0.52	5.13	0.88	4.22
Average							0.39	3.41	0.74	3.47

Table 9 Results for class G_9

Problem	*m*	*n*	*Opt*	*Best*	*Avrg*		*Gap*			CPU
					GRASP	*SS*	*Best*	*GRASP*	*SS*	(seconds)
p105	30	60	4701	4701	4837.5	4715.4	0.00	2.90	0.31	5.40
p106	30	60	5456	5456	5906.4	5471.0	0.00	8.25	0.27	5.45
p107	30	60	16781	16807	17259.2	16875.3	0.15	2.85	0.56	5.51
p108	30	60	14668	14671	15227.7	14701.9	0.02	3.82	0.23	5.42
p109	30	60	47249	47282	50099.7	47325.4	0.07	6.03	0.16	8.80
p110	30	60	41007	41028	43848.7	41056.2	0.05	6.93	0.12	8.53
p111	30	60	61633	61701	64652.4	61765.3	0.11	4.90	0.21	10.34
p112	30	60	17246	17246	17946.9	17246.0	0.00	4.06	0.00	3.12
Average							0.05	4.97	0.23	6.57

Table 10 Results for class G_{10}

Problem	*m*	*n*	*Opt*	*Best*	*Avrg*		*Gap*			CPU
					GRASP	*SS*	*Best*	*GRASP*	*SS*	(seconds)
p113	30	75	7887	7887	8068.4	7895.6	0.00	2.30	0.11	7.97
p114	30	75	5114	5159	5693.5	5187.5	0.88	11.33	1.44	7.42
p115	30	75	36022	36261	38072.9	36397.8	0.66	5.69	1.04	16.25
p116	30	75	17676	17676	17891.2	17679.8	0.00	1.22	0.02	9.06
p117	30	75	48701	48749	52059.1	48831.3	0.10	6.90	0.27	12.68
p118	30	75	66230	66290	68689.4	66354.6	0.09	3.71	0.19	14.54
p119	30	75	58964	58970	59846.4	59004.5	0.01	1.50	0.07	9.22
p120	30	75	79614	79678	83306.7	79776.1	0.08	4.64	0.20	11.45
Average							0.23	4.66	0.41	11.07

Table 11 Results for class G_{11}

Problem	*m*	*n*	*Opt*	*Best*	*Avrg*		*Gap*			CPU
					GRASP	*SS*	*Best*	*GRASP*	*SS*	(seconds)
p121	30	90	5937	5940	6055.4	5950.6	0.05	1.99	0.23	8.51
p122	30	90	9060	9123	9365.5	9128.7	0.70	3.37	0.76	11.98
p123	30	90	34652	34680	36012.7	34732.3	0.08	3.93	0.23	17.05
p124	30	90	30038	30050	31719.6	30076.1	0.04	5.60	0.13	20.36
p125	30	90	43853	43880	46339.4	43896.0	0.06	5.67	0.10	13.29
p126	30	90	69610	69821	73035.0	69898.7	0.30	4.92	0.41	20.38
p127	30	90	64474	64474	72569.7	64492.2	0.00	14.11	0.03	13.03
p128	30	90	49791	49791	51447.7	49794.1	0.00	3.33	0.01	10.87
Average							0.16	5.36	0.23	14.46

Table 12 Average deviation and optimal solutions for the first set of problems

Class	Average deviation from optimal			Number of optimal solutions	
	GRASP-Tabu	*SS*	*VLNS*	*SS*	*VLNS*
G_1	0.00	0.00	0.00	24/24	24/24
G_2	0.06	0.03	0.07	14/16	10/16
G_3	0.18	0.05	0.01	15/15	10/15
G_4	0.08	0.03	0.03	15/16	9/16

Table 13 Average deviation and optimal solutions for the second set of problems

Class	Average deviation from optimal			Number of optimal solutions	
	GRASP-Tabu	*SS*	*HB*	*SS*	*HB*
G_5	1.36	0.22	0.12	5/6	6/6
G_6	2.80	0.68	0.15	4/11	11/11
G_7	1.85	0.40	0.07	5/8	8/8
G_8	2.27	0.74	0.27	2/8	6/8
G_9	2.81	0.23	0.10	3/8	6/8
G_{10}	2.17	0.41	0.08	2/8	6/8
G_{11}	1.69	0.23	0.03	2/8	5/8

ever, in terms of the number of optimal solutions obtained (at least in one run), the proposed algorithm outperforms the VLNS procedure for all but one class of benchmark instances (class G_1). With respect to the second set of benchmark instances, although the proposed approach provides good quality solutions, the obtained results are outperformed by the Hybrid algorithm proposed in Delmaire et al. (1999b) both, in solution quality and in the number of optimal solutions obtained (see Table 13). This may be due to the fact that in Delmaire et al. (1999b), the strategies used for finding good solutions for the allocation problem, focus on instances where the capacity constraints are very tight in the optimal solution.

5 Conclusions

In this work we propose a scatter search approach for the Single Source Capacitated Facility Location Problem. The most distinctive characteristics of the proposed method are the following. The *Diversification Generation Method* uses a GRASP algorithm to provide a set of diverse trial solution. The *Solution Combination Method* combines solutions in three different ways that differ from each other in the criterion used to determine the initial assignment of each client. Then, an iterative approach is applied in order to improve the combined solution. Finally, the best solution found is improved with a simple tabu search algorithm.

To evaluate the performance of the proposed method several computational experiments have been performed over two different sets of test problems. Computational results demonstrate the effectiveness of the proposed approach since it provides high-quality solutions with reasonable computational effort.

Acknowledgements The authors are grateful to the referees for their valuable and insightful comments.

References

Ahuja, R., Orlin, J., Pallottino, S., Scaparra, M., & Scutellà, M. (2004). A multi-exchange heuristic for the single-source capacitated facility location problem. *Management Science*, *50*(6), 749–760.

Barceló, J., & Casanovas, J. (1984). A heuristic algorithm for the capacitated plant location problem. *European Journal of Operational Research*, *15*(2), 212–226.

Barceló, J., Fernández, E., Hallefjord, A., & Jörnsten, K. (1990). Lagrangean relaxation and constraint generation procedures for capacitated plant location problems with single sourcing. *Operations Research Spektrum*, *12*(12), 79–88.

Barceló, J., Fernández, E., & Jörnsten, K. (1991). Computational results from a new Lagrangean relaxation algorithm for the capacitated plant location problem. *European Journal of Operational Research*, *53*(1), 38–45.

Beasley, J. (1993). Lagrangean heuristics for location problems. *European Journal of Operational Research*, *65*(3), 383–399.

Cortinhal, M., & Captivo, M. (2003). Upper and lower bounds for the single source capacitated location problem. *European Journal of Operational Research*, *151*(2), 333–351.

Darby-Dowman, K., & Lewis, H. (1988). Lagrangian relaxation and the single source capacitated facility location problem. *Journal of the Operational Research Society*, *39*(11), 1035–1040.

Delmaire, H., Díaz, J., Fernández, E., & Ortega, M. (1999a). Comparing new heuristics for the pure integer capacitated plant location problem. *Investigación Operativa*, *1–3*, 217–242.

Delmaire, H., Díaz, J., Fernández, E., & Ortega, M. (1999b). Reactive grasp and tabu search based heuristics for the single source capacitated plant location problem. *Information Systems and Operational Research*, *37*(3), 194–225.

Díaz, J., & Fernández, E. (2002). A branch-and-price algorithm for the single-source capacitated plant location problem. *Journal of the Operational Research Society*, *53*(7), 728–740.

Feo, T., & Resende, M. (1995). Greedy randomized adaptive search procedures. *Journal of Global Optimization*, *6*(2), 109–133.

Glover, F. (1977). Heuristics for integer programming using surrogate constraints. *Decision Sciences*, *8*, 156–166.

Glover, F. (1997). A template for scatter search and path relinking. In J. Hao, E. Lutton, E. Rolland & M. Schoenauer (Eds.), *Lecture notes in computer science* (Vol. 1363, pp. 13–54).

Glover, F. (1999). Scatter search and path relinking, In D. Corne, M. Dorigo & F. Glover (Eds.), *New ideas in optimization* (pp. 297–316).

Hindi, K., & Pieńkosz, K. (1999). Efficient solution of large scale, single-source, capacitated plant location problem. *Journal of the Operational Research Society*, *50*(3), 268–274.

Holmberg, K., Rönnqvist, M., & Yuan, D. (1999). An exact algorithm for the capacitated facility location problems with single sourcing. *European Journal of Operational Research*, *113*(3), 544–559.

Klincewics, J., & Luss, H. (1986). A Lagrangian relaxation heuristic for the capacitated facility location with single-source constraints. *Journal of the Operational Research Society*, *37*(5), 495–500.

Laguna, M., & Martí, R. (2003). *Scatter search: methodology and implementations in C*. Boston: Kluwer Academic.

Martí, R. (2006). Scatter search: wellsprings and challenges. *European Journal of Operational Research*, *169*(2), 351–358.

Mirchandani, P. B., & Francis, R. L. e. (1990). *Discrete location theory*. Wiley: New York.

Neebe, A., & Rao, M. (1983). An algorithm for the fixed-charge assigning users to source problem. *Journal of the Operational Research Society*, *34*(11), 1107–1113.

Pirkul, H. (1987). Efficient algorithms for the capacitated concentrator problem. *Computers & Operations Research*, *14*(3), 197–208.

Rönnqvist, M., Tragantalerngsak, S., & Holt, J. (1999). A repeated matching heuristic for the single-source capacitated facility location problem. *European Journal of Operational Research*, *116*(1), 51–68.

Sridharan, R. (1993). A Lagrangian heuristic for the capacitated plant location problem with single source constraints. *European Journal of Operational Research*, *66*(3), 305–312.