IEMS Vol. 8, No. 1, pp. 29-36, March 2009.

Multi-Exchange Neighborhood Search Heuristics for the Multi-Source Capacitated Facility Location Problem

Chiuh-Cheng Chyu†
Department of Industrial Engineering and Management
Yuan-Ze University, 135 Yuan-Tung Road, Chung-Li, 320, Taiwan.
Tel: 886-3-4638800 ext 2511, Fax: +886-3-4638907, E-mail: iehshsu@saturn.yzu.edu.tw

Wei-Shung Chang
Department of Industrial Engineering and Management
Yuan-Ze University, 135 Yuan-Tung Road, Chung-Li, 320, Taiwan.
Tel: 886-3-4638800 ext 2511, Fax: +886-3-4638907

Received July 17 2008/Accepted September 8 2008

Abstract. We present two local-search based metaheuristics for the multi-source capacitated facility location problem. In such a problem, each customer's demand can be supplied by one or more facilities. The problem is NP-hard and the number of locations in the optimal solution is unknown. To keep the search process effective, the proposed methods adopt the following features: (1) a multi-exchange neighborhood structure, (2) a tabu list that keeps track of recently visited solutions, and (3) a multi-start to enhance the diversified search paths. The transportation simplex method is applied in an efficient manner to obtain the optimal solutions to neighbors of the current solution under the algorithm framework. Two in-and-out selection rules are also proposed in the algorithms with the purpose of finding promising solutions in a short computational time. Our computational results for some of the benchmark instances, as well as some instances generated using a method in the literature, have demonstrated the effectiveness of this approach.

Keywords: Capacitated Facility Location Problems, Transportation Simplex Method, Sensitivity Analysis, Simulated Annealing, Variable Neighborhood Structures

1. INTRODUCTION

The facility location problem has been a popular research topic in operations research since the early 1960s. Given a set of potential locations for facility and a set of customers, the objective of the facility location problem is to open facility in such a way that the total cost of opening facility and satisfying customers' demand is the minimum. This problem has applications in deciding placements of warehouses, factories, hospitals, schools, and proxy servers on the web.

Generally, the facility location problem can be divided into two categories: uncapacitated facility location problems (UFLP) and capacitated facility location problems (CFLP). In the UFLP, each facility is assumed to have unlimited capacity. This is frequently the case in small electronic components such as capacitors, resistors, etc. The CFLP can be further divided into two classes: single-source (SSCFLP) and multi-source (MSCFLP). For the former, each customer's demand is supplied by exactly one facility. On the other hand, for the MSCFLP, a customer's demand is supplied by one or more facilities. The facility location problem without capacity constraints is NP-hard (Lenstra *et al.*, 1978), but this problem is not harder than the CFLP, regardless of being single-source or multi-source. Accordingly, both CFLPs are NP-hard. There has been much research focused on SSCFLP, but significantly less effort has been devoted to the MSCFLP.

A SSCFLP is more difficult for finding an optimal solution than the MSCFLP under the same problem structure. Given a set of opened facilities, the resulting SSCFLP is an optimal set partitioning problem, which is NP-hard, but the resulting MSCFLP is a transportation problem, which belongs to class P and can be solved optimally

† : Corresponding Author

by the well-known transportation simplex algorithm on average in polynomial time. The Lagrangian relaxation techniques have been extensively applied to solve the SSCFLP (Pirku, 1897, Sridharan, 1993, Beasley, 1993, Holmberg and Rönnqvist, 1999, Cortinhal and Captivo, 2003). These methods are different in that lower bounds and feasible solutions are generated. Recently, heuristics such as Tabu search, Simulated Annealing, and Genetic algorithms have been applied to solve the SSCFLP and its variants (Arostegui *et al*., 2006). Ahuja *et al*. (2004) developed a heuristic using very large-scale neighborhood structures (VLSN), including single-customer multi-exchanges, multi-customer multi-exchanges, and three types of facility moves-opening, closing, and transferring. Their computational results show that the VLSN outperforms the Holmberg's Largrangian heuristic for the large size benchmark instances generated by Holmberg *et al*. (1999).

Sirdharan (1995) provides a comprehensive survey for the MSCFLP. Most solution approaches are also based on Lagrangian relaxation. Korupolu and Plaxton (2000) present an analysis of a local search heuristic using the facility moves mentioned above for various facility location problems; in particular, they prove that their heuristic yields a solution with an (8 + ε) approximation bound for MSCFLP. Barahona and Chudak (2005) propose a heuristic combining the volume algorithm (Barahona and Anbil, 2000) and the randomized rounding algorithm for solving large scale instances with problem sizes from 300×300 to 1000×1000. The volume algorithm solves the Lagrangian relaxation problem for lower bounds, and the randomized rounding algorithm is used for producing feasible solutions. The algorithm terminates when the gap between the value of the current best feasible solution and the current lower bound becomes less than 1%, or the iterations of the volume algorithms have reached a preset value. Klose and Drexl (2005) propose a new lower bound method for the MSCFLP based on partitioning the plant set and employing column generation.

The optimal cost of a SSCFLP is in general larger than its corresponding MSCFLP since the feasible solution set of the former is included in that of the latter. In practice, unless it is specified that the single-source condition must be satisfied, solving the MSCFLP is more appropriate than solving the SSCFLP. This paper aims at developing a simple and effective approach to solve the MSCFLP. The approach covers the concepts and techniques that are familiar to most OR practitioners, and is easy to be implemented. Unlike many published articles that solve the MSCFLP using the Lagrangean relaxation method as the core technique, our approach adopts a simple solution representation structure, which can easily corperate with the transportation simplex method under a simple local search metaheuristic framework to solve the MSCFLP. To avoid spending much computational time on solving the transportation problem whenever a new solution list is generated, the proposed approach adopts different small-change neighborhood structures so that the reoptimization technique in the transportation simplex method can be applied to quickly find new optimal solutions. Accordingly, local-search based metaheuristics appear to be proper methods for solving the MSCFLP. In addition, because the set of the opened facilities in the optimal solution is unknown, variable neighborhood structure is a proper choice for designing an effective algorithm.

Based on the above observation and knowledge, we propose two local-search based algorithms for the MSCFLP: one uses adaptive simulated annealing (SA) with tabu list, and the other can be regarded as a variant of variable neighborhood search, which we will call variable neighborhood local search (VNLS). The MSCFLP has large-scale benchmark instances in the OR library. To test the performance of the proposed methods, additional test instances of various problem sizes that are solvable by the LINGO optimizer were generated.

The remainder of the paper is organized as follows: Section 2 describes the MSCFLP in detail and presents a mathematical model. Section 3 illustrates the adaptive SA and the VNLS. Section 4 presents the numerical results of the proposed algorithms using the instances in the OR Library, instances created by Holmberg *et al*. (1999), and instances created in this paper. Section 5 concludes the paper.

2. PROBLEM DESCRIPTION AND FORMULATION

The multi-source capacitated facility location problem can be described as follows: Let G = (F, C) be a bipartite network, where F is the set of facilities with cardinality |F| = M, and C is the set of customers with cardinality |C| = N. Let f_i denote the opening cost of facility i, C_{ij} the unit shipping cost from facility i to customer j, S_i the capacity of facility i, and D_j the demand of customer j. The problem is to find a subset $I \in F$ of facilities that should be opened, and a transportation assignment for shipping total demanded goods from these opened facilities to all customers such that the total cost is the minimum. The following is the integer programming model for this problem.

$$Minimize \quad \sum_{i=1}^{M}\sum_{j=1}^{N} c_{ij}x_{ij} + \sum_{i=1}^{M} f_i y_i \tag{1}$$

$$s.t. \quad \sum_{j=1}^{N} x_{ij} \le S_i y_i \qquad i = 1, \cdots, M \tag{2}$$

$$\sum_{i=1}^{M} x_{ij} \ge D_j \qquad j = 1, \cdots, N \tag{3}$$

$$y_i \in \{0, 1\} \qquad i = 1, \cdots, M \tag{4}$$

$$x_{ij} \ge 0 \qquad i = 1, \cdots, M;\ j = 1, \cdots, N \tag{5}$$

In the above mathematical model, x_{ij} is the number of units shipped from facility i to customer j, and y_i = 1 if facility i is open and 0 otherwise. The set of constraints (2) confines in such a way that the total supply by facility *i* can never exceed its capacity once this facility is open, and the set of constraints (3) restricts the demand of each customer to be satisfied. Expression (1) presents the minimization of the objective function which considers the facility opening cost and the transportation cost for shipping all required units.

3. LOCAL-SEARCH BASED ALGORITHMS

Two types of local-search based metaheuristics are developed to solve the MSCFLP: one is under the simulating annealing framework, and the other uses the variable neighborhood local search framework. The solution representation and basic neighborhood structures of either algorithm framework are the same, but they are employed in different manners with the objective of finding a near optimal solution in a short computational time.

3.1 Solution Representation and Neighborhood Structures

One of the main difficulties of dealing with the MSCFLP is that the number of opened facilities in an optimal solution is unknown. To avoid missing this optimal number, we define a basic neighborhood structure in which each neighboring unit contains four neighbors of three types: (1) one neighbor of opening one more facility, (2) one neighbor of closing one opened facility, and (3) two neighbors of closing one while opening another one. The best solution among these four neighbors will be selected as the candidate for contesting the next current solution. If the cost value of this candidate is smaller than that of the current solution or is larger but passes the acceptance test, then the candidate becomes the next current solution; otherwise, another neighboring unit is generated and the best one will again be selected as the candidate.

Given a selected set of open facilities, the resulting problem becomes a transportation problem which can be efficiently solved to optimality by the transportation simplex method. Thus, any solution can be represented by a selection list; i.e., the *i*th space of the list has value 1 if facility *i* is open; otherwise it has value 0. Figure 3.1 presents a solution list for a problem with five facilities, where facilities 1, 3, and 4 are opened and facilities 2 and 5 are closed, and displays an example of one-out one-in neighbor generation: close facility 3 of the current solution *x* and open facility 5.

	1	2	3	4	5
x	1	0	1	1	0
new	1	0	0	1	1

Figure 3.1. A one-out and one-in neighbor of solution *x*.

Examples of extensive neighborhood structure can be two-out, two-in, two-out and two-in, three-out and three-in, etc. It is intuitive that, as the problem size grows, combining the use of large and small neighborhoods will increase the chance of producing or promoting a favorable result since it *i*mplies that the algorithm will explore solution space more globally and locally.

3.2 Computational Issues on Generating a Neighboring Unit

When a neighbor is generated, it will be extremely inefficient if the new problem is solved by restarting the transportation simplex method. For example, in the one-out one-in case, since the neighbor has closed one facility (one row deleted) while opening another new one at the same time (another row added), the computational time will be greatly reduced if we continue the current optimal basic feasible solution to reoptimize the new problem. Our experimental results have proved that such is true. This simple technique will be of great help in solving large size problems.

Figure 3.2 shows the reoptimization method for a new neighbor of one-out and one-in type. In this example, facility 3 is out and facility 5 is in. The upper table is the optimal basic variable solution when facilities 1, 3, and 4 are open. The lower table shows the reoptimization method. Since the second row is to be removed, all transportation costs of this row are set to be a large number M except for the last column, which represents the unused capacity of facility 3. On the other hand, we add the selected facility 5 to the last row and assign the dummy slack as a basic variable with a value equal to the capacity of facility 5. By doing so, when the optimal solution to the augmented problem is obtained, the dummy slack of the second row has a value 200, and this row can then be deleted. Thus, the new optimal basic feasible solution without this row is optimal to the neighbor.

As for the other two basic types of neighbors, if a neighbor is one-out, all transportation costs of the selected row are assigned a big value M except for the last column. If a neighbor is one-in, then the selected facility is added to the last row with its dummy slack serving as a basic variable.

A neighbor defined by two-out and two-in can also be reoptimized by executing one-out one-in exchange two times in a row. The same approach is applied to finding the optimal solution to the following neighbors: k-out and k-in, k-out, and k-in for $k \geq 2$.

3.3 Neighbors Generation Method

Two rules are used to generate neighbors of the current solution: (1) random selection (2) average cost per unit shipped. In the random selection rule, both facility-outs and facility-ins are selected randomly and the resulting neighbor is accepted when the total supply-demand constraint is not violated. In the second rule, the average cost per shipped unit (denoted AC_i) for facility i is defined as follows.

$$AC_i = \frac{\text{transportation cost per shipment}}{\text{capacity used}} + \frac{\text{opening cost}}{\text{capacity used} \cdot (\beta \text{ shipments / period})} \tag{6}$$

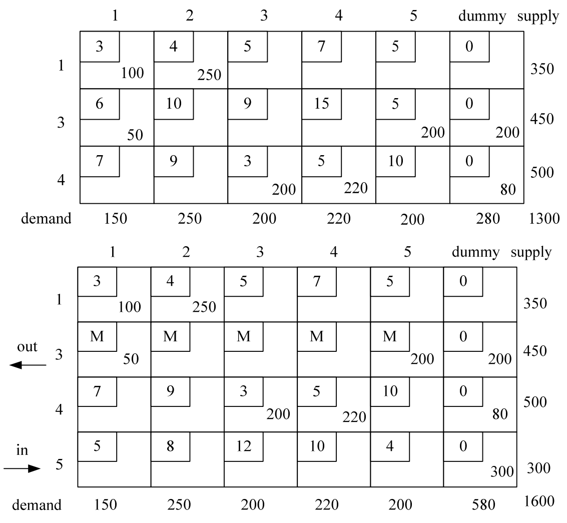
In the example of Figure 3.2, if the opening cost of facility 3 is 1000 and $\beta = 2$ shipments per period, then the average cost per shipped unit is

$$AC_3 = \frac{50 \cdot 6 + 200 \cdot 5}{450 - 200} + \frac{1000}{(450\text{-}200) \cdot 2} = 7.2$$

On the other hand, the entry average cost of the added facility 5, which has opening cost 600 per period, is computed based on the current status of the leaving facility 3:

$$AC_5^{entry} = \frac{50 \cdot 5 + 200 \cdot 4}{250} + \frac{600}{250 \cdot 2} = 5.4 < 7.2$$

It is clear that the optimal solution of the neighbor will be better than the current solution since the reoptimization has not been done yet. Under this rule, we choose an opened facility with the largest average cost to be closed, and choose a closed facility with the smallest average cost to be opened. In either selection, if there happens to be a tie, then one is randomly selected.



	1	2	3	4	5	dummy	supply
1	3 / 100	4 / 250	5	7	5	0	350
3	6 / 50	10	9	15	5 / 200	0 / 200	450
4	7	9	3 / 200	5 / 220	10	0 / 80	500
demand	150	250	200	220	200	280	1300

		1	2	3	4	5	dummy	supply
	1	3 / 100	4 / 250	5	7	5	0	350
out ←	3	M / 50	M	M	M	M / 200	0 / 200	450
	4	7	9	3 / 200	5 / 220	10	0 / 80	500
in →	5	5	8	12	10	4	0 / 300	300
	demand	150	250	200	220	200	580	1600

Figure 3.2. Reoptimizing a 1-out 1-in neighbor of the current solution.

3.4 Simulated Annealing

Simulated annealing (SA) is an iterative algorithm that consists of a chain of local searches. The search of SA starts from an initial feasible solution. During the search process, a neighbor is usually generated according to a simple rule, and the cost value of this neighbor is compared to that of the current solution. If the cost value of the neighbor is lower than that of the current solution, then this neighbor replaces the current solution. However, if the neighbor does not improve the current solution, there is still a chance of replacement according to the following probability function:

$$\Pr\{\text{acceptance of transition}\} = \exp\left(-\frac{\Delta C_i}{T_i}\right) \tag{7}$$

where ΔC_i is the cost difference between the neighbor and the current solution in iteration i, and T_i is the control parameter known as temperature. If the transition from the current solution to the neighbor is rejected, another solution in the neighborhood will be generated and evaluated.

Four SA procedures are considered in this paper: (1) std-SA-rdm, (2) adt-SA-rdm, (3) adt-SA-tb-rdm, and (4) adt-SA-tb-rule, where "std-SA" means that the algorithm uses standard SA procedure with geometric cooling scheme, "adt" means that the SA algorithm uses the adaptive annealing scheme according to the formula proposed by Aarts and Korst(1991), "tb" implies that the algorithm uses tabu list, "rdm" means that the algorithm uses random rule to select in and out facilities, and "rule" represents that the algorithm adopts the deterministic rule described in section 3.3.

Aarts and Korst (1991) provided an adaptive annealing formula shown below:

$$T_{i+1} = \frac{T_i}{1 + T_i \cdot Ln(1+\delta)/3\sigma_i} \tag{8}$$

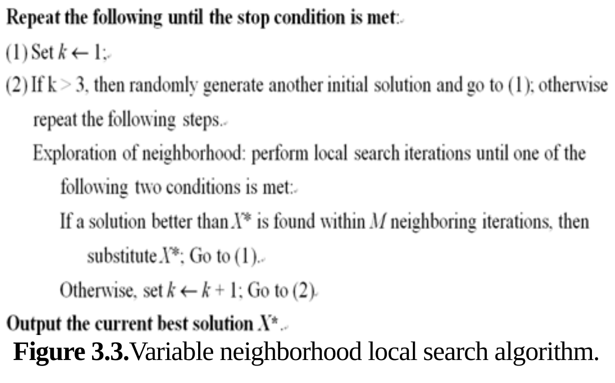
where T_i is the temperature in iteration i, σ_i is the standard deviation of the visited solutions in iteration i, and δ is an empirical distance parameter and normally takes the value 0.5. Small values of δ lead to large decrements in T_{i+1}. Aarts and Korst also provided a formula for the initial temperature T_0, but our experimental result indicates that the initial temperature does not work better than a constant obtained from the experiment.

In the proposed four SAs, every neighboring unit consists of four neighbors of different types: one-in, one-out, one-in one-out, and two-in two-out. Thus, an iteration of a neighboring unit will produce four solutions, and the best one of them is chosen to compete for the position of current representative solution. The proposed SAs use the strategy of restarting the algorithm when any of preset conditions is met. For each annealing temperature, the Markov chain performs ten transitions. A process will stop when one of the following two conditions is satisfied: (1) No further improvement occurs during the last 30 transitions, and (2) The maximum number of transitions has been reached. For each searching process of the three adaptive SA procedures, ten annealing temperatures are generated according to formula (8) with initial temperature set at 100, which is concluded based on our computational experiment. The algorithms will stop if ten restarting processes have been performed. The length of the tabu list is set as the number of facilities divided by five. In std-SA-rdm, the cooling scheme is $T_{i+1} = \alpha \cdot T_i$, $i = 1, \cdots, 50$ with $\alpha = 0.95$, and at each temperature the Markov chain performs 20 transitions. Introducing a tabu list will improve the performance of the proposed SA algorithm, as demonstrated by the experimental results with the instances provided by Holmberg (see Tables 4.1 - 4.3).

Each of the four SA procedures will compute at most one thousand neighboring unit iterations; in other words, a maximum total of four thousand solutions.

3.5 Variable Neighborhood Local Search

The variable neighborhood local search (VNLS) algorithm defines three neighborhood structures: N_1, N_2, and N_3. The neighboring unit of N_k consists of one k-in neighbor, one k-out neighbor, and two k-in k-out neighbors; i.e., an iteration of local search produces four solutions. Likewise, if the best of the four solutions is better than the current best one, then the current best will be substituted; otherwise, another neighboring unit is generated and compared. A tabu list is used to escape from searching the solution space that has previously been explored. The distance between the current best solution and its neighbors in N_k will become farther as k increases. The local search in N_1 is intensive, focusing on searching a local optimum within a small region near the current best solution, whereas the solution search in N_3 is more extensive, exploring a better solution at a farther distance from the current best solution. The parameter used for advancing to the next neighborhood is set to be M, where M is the number of facility locations in the problem. The algorithm stop condition is one thousand neighboring unit iterations. The random rule and deterministic rule described in section 3.3 are also applied in the VNLS, and will be denoted as VNLS-tb-rdm and VNLS-tb-rule, respectively. Figure 3.3 presents the algorithm framework.



```
Repeat the following until the stop condition is met:
(1) Set $k \leftarrow 1$;
(2) If $k > 3$, then randomly generate another initial solution and go to (1); otherwise
    repeat the following steps.
    Exploration of neighborhood: perform local search iterations until one of the
        following two conditions is met:
        If a solution better than $X^*$ is found within $M$ neighboring iterations, then
            substitute $X^*$; Go to (1).
        Otherwise, set $k \leftarrow k + 1$; Go to (2).
Output the current best solution $X^*$.
```

Figure 3.3. Variable neighborhood local search algorithm.

4. NUMERICAL RESULTS

In this section, the performance of the six local-search based algorithms is presented. These algorithms are encoded in Microsoft Visual C++ 6.0 and tested using a PC Pentium IV with 3.0GHz and 1024MB DDR2 SDRAM.

Two sets of benchmark instances are used to test the performance of the six algorithms. One set is taken from Holmberg (1999), which provides 71 instances. The number of facilities and customers grow as the instance number increases, ranging from 10×50 to 30×200, where the first figure represents the number of facility locations and the second one represents the number of customers. The other set has large 100×1000 sized problems, which are taken from the OR Library. The Holmberg instances are originally of single-source type; i.e., for each customer, only his demand quantity and the total transportation cost associated with each facility are given. The test instances can be converted into a multi-source type when these total transportation costs are divided by the corresponding demand quantities. These converted instances are then solved to optimality by LINGO 8.0.

In the experiment, all SA procedures set the initial temperature at 100. Both SA and VNLS set $\beta = 2$ whenever the deterministic rule is used. The algorithms solve each instance using five independent runs. Two performance measures for the algorithms are used: minimum deviation (min. dev.) and average deviation (ave. dev.) from the optimal value. The minimum deviation is defined as the ratio of the difference between the best value of the five runs and the optimal value over the optimal value, whereas the average deviation takes the difference between the average of the five runs and the optimal value. We can observe from Table 4.1 that all algorithms produce satisfactory results in terms of the minimum and the average deviations from the optimal values for large size instances of Holmberg, but the three algorithms, adt-SA-tb-rdm, adt-SA-tb-rule, and VNLS-tb-rdm, perform significantly better than the other three. It is little surprise that algorithm VNLS-tb-rule does not perform well on these instances. It is likely due to the following reasons: (1) Each of the ten searching processes only accepts solution that surpasses the current best one, (2) The set of facility locations in these instances is not sufficiently big and applying this rule is likely to converge quickly to a local optimal solution in a small region even though a tabu list is used, and (3) The solution searches with neighbor 3-in and with neighbor 3-out in N_3 are likely to be ineffective due to small or medium problem size, and supply-demand and cost structures; i.e., a neighbor of adding three facilities generally produces larger cost while a neighbor of closing three facilities is likely infeasible.

The parameter setting of β greatly influences the performance of the adt-SA-tb-rule. This algorithm performs very well on problems of large sizes with parameter $\beta \geq 1$, but performs poorly with parameter $\beta \geq 6$. Our experimental results show that a combination of β = 0.1, 0.5, 1.0 will find optimal solutions within five replication runs for all large size Holmberg instances.

A similar conclusion holds for these algorithms when all categories of Holmberg instances are considered. Table

4.2 displays the number of optimal solutions found in each category based on five runs. Again, the three algorithms with better performance significantly surpass the others. In particular, algorithm VNLS-tb-rdm seems to be the best among all, which is concluded according to the number of optimal solutions found in each category or the average deviation from the optimal values based on all Holmberg instances.

Table 4.3 shows the computational results of the algorithms for the large instances drawn from OR Library. Obviously, the two VNLS algorithms outperform the four SAs. In particular, VNLS-tb-rule clearly exceeds the performance of all the others in terms of deviations from optimal values, regardless of the average performance of five runs or the best performance of five runs. Furthermore, adt-SA-tb-rule and VNLS-tb-rule solve these OR library instances with shorter computational time than the others. The experimental results indicate that using the VNLS-tb-rule is more likely to produce good results in a short computational time for very large size instances.

In addition to the above benchmark instances, five 50×200 instances and five 100×200 instances are generated using the method described below. The LINGO 8.0 optimizer is capable of solving optimally the five 50×200 instances within one hour, but it will terminate with a local optimal solution for each of the larger size 100×200 instances within the software problem solving capacity. Thus, the computational time of the LINGO optimizer for the 100×200 instances was set to two hours, and the obtained upper bounds and lower bounds will be used for evaluating the performance of the proposed heuristic.

Table 4.3. Performance of the algorithms on large size instances (OR Library, 100×1000).

Algorithm	Deviation (%)		Ave. CPU sec.
	Min.	Ave.	
std-SA-rdm	4.58	2.01	402.60
adt-SA-tb-rdm	4.86	3.07	444.65
adt_SA_rdm	5.61	2.03	404.50
adt-SA-tb-rule	2.38	1.59	289.19
VNLS-tb-rdm	0.37	1.50	541.75
VNLS-tb-rule	0.02	0.81	371.37

The method to generate these ten instances is summarized below:

First of all, *m* two-dimension coordinates locations are randomly generated on the interval (100, 500). Then the heuristic for selecting the p-median by Teitz and Bart (1968) is used to choose the locations of candidate facilities from these coordinates. Finally, the locations of the *N* customers are selected from the remaining locations.

In the next step, the transportation cost from facility *i* to customer *j*, denoted C_{ij}, is a product of the distance d_{ij} and a random number from U [5, 10]; that is, a uniform distribution on the interval [5, 10]. To generate the demand for each customer of the instances, the procedure is as follows:

At first, the total demand D is chosen. Consequently, the average demand of each customer, d_{ave}, is D/*N*. The standard deviation of a customer's demand, d_{sd}, is drawn from a random number from $d_{ave} \cdot$ U [0.5, 1]. A custom-

Table 4.1. Performance of the algorithms on large instances from Holmberg instances

	30×150 (hol 25-40)			30×200(hol 56-71)		
Algorithm	Deviation (%)		Ave. CPU sec.	Deviation (%)		Ave. CPU sec.
	Min.	Ave.		Min	Ave.	
std-SA-rdm	0.06	0.38	10.76	0.39	0.94	20.38
adt-SA-tb-rdm	0.00	0.16	13.29	0.25	0.90	21.66
adt-SA-rdm	0.76	1.29	9.44	0.40	1.30	20.60
adt-SA-tb-rule	0.00	0.34	12.08	0.06	0.48	17.87
VNLS-tb-rdm	0.00	0.08	24.58	0.03	0.49	34.96
VNLS-tb-rule	0.20	1.41	19.70	0.61	1.80	26.43

Table 4.2. Number of optimal solutions found by the algorithms on all test instances.

Algorithm	10×50 (hol 1-12)	20×50 (hol 13-24)	30×150 (hol 25-40)	variation (hol 41-55)	30×200 (hol 56-71)	all (hol 1-71)
	# optimal solutions found within five runs					Deviation (%)
std-SA-rdm	12	12	14	13	9	0.59
adt-SA-tb-rdm	12	12	16	14	11	0.33
adt-SA-rdm	11	10	14	14	8	1.05
adt SA-tb-rule	12	12	16	12	13	0.28
VNLS-tb-rdm	12	12	16	14	13	0.19
VNLS-tb-rule	12	10	11	11	9	0.99

er's demand then takes a number randomly generated from U [d_{ave}- d_{sd}, d_{ave}+ d_{sd}]. Likewise, a similar procedure is used to determine the capacity of a facility based on a given total capacity of the facilities. The opening cost of facility i, f_i, is counted as a product of three numbers: a constant, capacity of facility i, and a random number from U[0.6, 0.8]. Two test sets are generated: one of problem size 50×200 and the other of problem size 100×200. Each set has five instances. The ratio of total capacity over total demand for the 50×200 instances is about 4, whereas that for the 100×200 is about 8.

An experiment was conducted to learn the computational complexity of the LINGO optimizer in solving the MSCFLP with respect to different capacity/demand ratios. In the experiment, three test instances of problem size 25 ×50 are generated for ratios with a range between 1.1 and 4. The experimental results indicate that the LINGO optimizer can optimally solve for small ratios MSCFLP in a short computational time, particularly when the ratios are between 1.05 and 1.4. The MSCFLP becomes hard when the capacity/demand ratio is large.

In the experiment on the 50×200 and 100×200 test instances, the VNLS algorithms solve each instance with five independent runs. The termination condition of each run on a 50×200 instance is 4,000 solutions, and on a 100 ×200 instance is 20,000 solutions. For both types of instances, the running times of the VNLS-tb-rule are shorter than those of the VNLS-tb-rdm, but the former behaves slightly worse than the latter in terms of the average performance and the best performance. Such results are similar to those of other types of instances. When compared with the LINGO optimizer, the VNLS algorithms are apparently better in practice. As perceived from Table 4.4, the LINGO optimizer will take on average 1422.8 seconds to find an optimal solution for the five 50×200 instances, whereas the two VNLS algorithms will find the optimal solution in a much shorter time with an average of approximately 50 seconds for rule and 75 seconds for random. Such estimates are taken based on the fact that the minimum of the five trials have found the optimal

Table 4.4. Performance of the VNLS algorithms on 50 facilities and 200 customers instances (5 replicates).

	VNLS-tb-rdm			VNLS-tb-rule			LINGO
No.	Deviation (%)		Ave. CPU sec.	Deviation (%)		Ave. CPU sec.	CPU sec.
	Min	Ave.		Min	Ave.		
1	0.00	0.29	30.63	0.00	0.28	20.39	2224
2	0.00	0.00	28.25	0.00	0.29	18.97	173
3	0.00	1.18	32.30	0.00	0.77	22.45	2647
4	0.00	0.00	31.88	0.00	0.61	22.00	1396
5	0.00	1.70	36.60	0.00	1.36	20.12	674
Ave.	0.00	0.63	31.20	0.00	0.66	20.79	1422.8

Table 4.5. Performance of LINGO on 100 facilities and 200 customs instances (2 hours).

No.	(1) LINGO upper bound	(2) LINGO lower bound	Gap =((1)-(2))/(2)	Ave. =((1)+(2))/2	CPU sec.
1	3686663	3657223	0.80%	3671943	7200
2	4001222	3890340	2.85%	3945781	7200
3	3779116	3744243	0.93%	3761680	7200
4	3981152	3939400	1.06%	3960276	7200
5	3903840	3809992	2.46%	3856916	7200

Table 4.6. Performance of VNLS algorithms on100 facilities and 200 customs instances (5 replicates).

	VNLS-tb-rdm				VNLS-tb-rule			
No.	Deviation (%)			Ave. CPU sec.	Deviation (%)			Ave. CPU sec.
	Min vs LINGO upper bound	Ave. vs LINGO lower bound	Ave. vs LINGO Ave.		Min vs LINGO upper bound	Ave. vs LINGO lower bound	Ave. vs LINGO Ave.	
1	0.00	0.80	0.40	30.63	0.00	1.13	0.72	20.39
2	-0.84	3.92	2.46	28.25	-0.84	5.41	3.93	18.97
3	-0.04	4.34	3.85	32.30	-0.04	6.15	5.66	22.45
4	-0.04	2.49	1.95	31.88	3.59	4.94	4.38	22.00
5	-0.87	3.74	2.48	36.60	-0.87	5.40	4.12	20.12
Ave.	-0.36	3.06	2.23	31.20	0.37	4.61	3.76	20.79

solutions for both algorithms and it is assumed that the optimal solution was found at the 2.5th trial. Meanwhile, the average deviations of the two VNLS algorithms are about the same. As a whole, the performance of the VNLS-tb-rule is superior to the VNLS-tb-rdm and LINGO optimizer.

In the experiment of 100×200 instances, the two VNLS algorithms are also superior to the LINGO optimizer in terms of solution quality and computational time. In Table 4.5, the LINGO optimizer can not optimally solve any of the five instances within two hours. The VNLS-tb-rdm can find better solutions than the LINGO optimizer in a much shorter running time for all the five instances, while the VNLS-tb-rule outperforms the LINGO optimizer in four of five instances. Table 4.6 presents the numerical results of both VNLS algorithms on these test instances. The performance measures using the average of upper bound and lower bound and using the lower bound show that both VNLS algorithms can find good solutions to these instances in short computational times. Similar to the numerical results of other instances, the VNLS-tb-rdm is better than the VNLS-tb-rule in solution quality but worse in computational time. The VNLS-tb-rdm can explore solutions in more diversified regions than the VNLS-tb-rule.

5. CONCLUSION

This paper presents two simple but effective local-search based algorithms for the multi-source capacitated facility location problem. One uses a SA framework and the other uses a variable neighborhood structures framework. The numerical results show that an SA with adaptive temperature control and tabu list achieves excellent performance on the test instances provided by Holmberg (1999). Also, a VNLS with random rule and tabu list obtains slightly better performance than the hybrid SA with either random rule or deterministic rule. Experiments on test instances of larger problem sizes also indicate that the performances of the two VNLS algorithms are superior to the commercial optimization software package-LINGO 8.0. In particular, algorithm VNLS-tb-rule works very well in terms of solution searching effectiveness and efficiency for the large size test instances in the OR library.

REFERENCES

Aart, E. H. L. and Korst, J. H. M. (1991), *Simulated annealing Boltzmann machines*: *a stochastic approach to combinatorial optimization and neural computing*, John Wiley, Chichester.

Ahuja, R. K., Liu, J., Orlin, J. B., Goodstein, J., and Mukherjee, A. (2004). A neighborhood search algorithm for the combined through and fleet assignment model with time windows, *Networks*, **44**(2), 160-171.

Arostegui, Jr., M. A., Kadipasaoglu, S. N., and Khumawala, B. M. (2006), An empirical comparison of tabu search, simulated annealing, and genetic algorithms for facilities location problems, *International Journal of Production Economics*, **103**(2), 742-754.

Barahona, F., Anbil, R. (2000), The volume algorithm: Producing primal solutions with a subgradient method, *Mathematical Programming*, Series B **87**(3), 385-399.

Barahona, F. and Chudak, F. A. (2005), Near-optimal solutions to large-scale facility location problems, *Discrete Optimization*, **2**(1), 35-50.

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

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

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

Klose, A. and Drexl, A., (2005), Lower bounds for the capacitated facility location problem based on column generation, Management Science, **51**(11), 1689-1705.

Korupolu, M. R., Plaxton, C. G., Rajaraman, R. (2000), Analysis of a local search heuristic for facility location problems, *Journal of Algorithms*, **37**(1), 146-188.

Lenstra, J. K., Rinnooy Kan, A. H. G. (1978), Complexity of scheduling under precedence constraints, *Operations Research*, **26**(1), 22-35.

Pirkul, H. (1987), Efficient algorithms for capacitated concentrator location problem, computers and operations Research, **14**(3), 197-208.

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

Sridharan R. (1995), The capacitated plant location problem, *European Journal of Operational Research*, **87** (2), 203-213.

Teitz, M. B. and Bart, P. (1968), Heuristic methods for estimating the generalized vertex median of a weighted graph, *Operations Research*, **16**(5), 955-961.