Local-Search Based Metaheuristics for the Multi-Source Capacitated Facility Location Problem

Wei-Shung Chang, Pong-Yeng Lei, Chiuh-Cheng Chyu*
Department of Industrial Engineering and Management
Yuan-Ze University, 135 Yuan-Tung Road, Chung-Li, 320, Taiwan.
*886-3-4638800 ext 2511, Fax: +886-3-4638907, Email: iehshsu@saturn.yzu.edu.tw

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. Transportation simplex method is applied in an efficient manner to obtain the optimal solutions to neighbors of the current solution under the algorithm framework. Our computational results for some of the benchmark instances demonstrate the effectiveness of this approach.

Keywords: Capacitated facility location problems, Transportation simplex method, Sensitivity analysis, Simulated annealing, Variable neighborhood search

1. Introduction

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 a single facility. On the other hand, the MSCFLP allows a customer's demand to be supplied by one or more facilities. The facility location problem without capacity constraints is NP-hard [1], but this problem is not harder than the CFLP, regardless of being single-source or multi-source. Accordingly, both CFLPs are NP-hard. Recently, there have been much research focused on SSCFLP, but very few are on MSCFLP.

A SSCFLP is more difficult in 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 by the well-known transportation simplex algorithm with an average in polynomial time. The Lagrangian relaxation techniques have been extensively applied to solve the SSCFLP [2-6]. These methods are different in that lower bounds and feasible solutions are generated. Recently, heuristics such as Tabu search, Simulated Annealing and Genetic algorithms are applied to solve the SSCFLP and its variants [7]. Ahuja et al. [8] develop 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. [5].

Sirdharan [9] provides a comprehensive survey for the MSCFLP. Most solution approaches are also based on Lagrangian relaxation. Korupolu and Plaxton [10] 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 $(8 + \varepsilon)$ approximation bound for MSCFLP. Barahona and Chudak [11] propose a heuristic combining the volume algorithm [12] and the randomized rounding algorithm for solving large scale instances with problem sizes from 300 x 300 to 1000 x 1000. The volume algorithm solves the Lagrangian relaxation problem for lower bounds, and the randomized rounding algorithm 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 [13] 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. In this paper, we propose two local-search metaheuristics for solving 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). Both methods adopt the multi-exchange neighborhood structure.

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 and the instances created by Holmberg et al. (1999). 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.

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

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

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

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

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

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 simulating annealing framework, and the other uses 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 the 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 involving four neighbors of three types: (1) opening one more facility, (2) closing one opened facility, and two neighbors of (3) 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 and open facility 5.

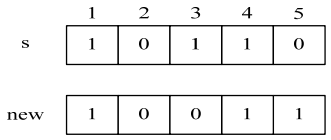

Figure 3.1 A one-out and one-in neighbor of current solutions

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 in producing or promoting a favorable result since it implies that the algorithm will explore solution space 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 result has proved 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. 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 ≥ 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.

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

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

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

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.

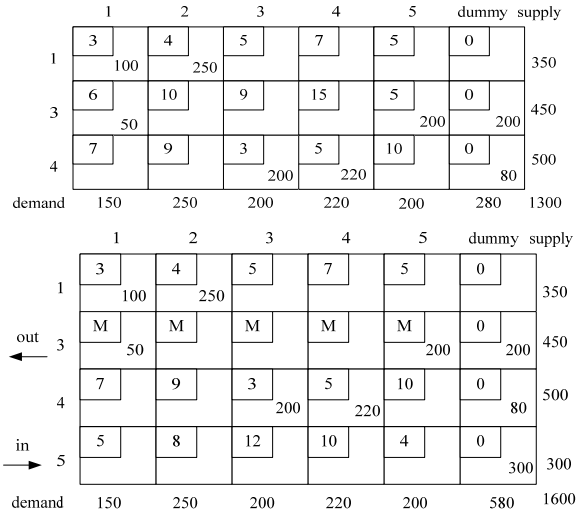

Figure 3.2 Reoptimizing a 1-out 1-in neighbor of the current solution

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:

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

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 [14], "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 [14] provided an adaptive annealing formula shown below:

$$T_{i+1} = \frac{T_i}{1 + T_i \cdot Ln(1+\delta)/3\sigma_i} \quad (7)$$

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 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, 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 (7) 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 $T_{i+1} = \alpha \cdot T_i$, $i = 1,...,50$ 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 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 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 optimal 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 for advancing 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 rue 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 run with PC Pentium IV 3.0GHz 1024MB (DDR2 SDRAM).

Two sets of benchmark instances are used to test the performance of the six algorithms. One set is taken from Holmberg [5], which provides 71 instances. The number of facilities and customers grow as the instance number increases, ranging from 10 x 50 to 30 x 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 size problems, 100 x 1000, which are taken from 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 and make five runs on each of the test instances. Both SA and VNLS set $\beta = 1$ whenever the deterministic rule is used. We can observe from Table 4.1 that all algorithms produce satisfactory results in terms of the average deviation 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 VNKS-tb-rule does not perform well on these instances. There could be 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 shortly 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.

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

Algorithm	30 x 150 (hol 25 - 40)			30 x 200(hol 56 - 71)		
	Deviation (%)		average CPU sec.	Deviation (%)		average CPU sec.
	min	average		min	average	
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 x 50 (hol 1 - 12)	20 x 50 (hol 13- 24)	30 x 150 (hol 25 - 40)	variation (hol 41 -55)	30 x 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

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 $\beta \le 1$, but performs poorly with parameter $\beta \ge 6$. Our experimental results show that a combination of $\beta = 0.1,\ 0.5,\ 1.0$ will find optimal solutions within five runs for all large size Holmberg instances.

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 excels all others in

terms of deviations from optimal values, regardless of the average performance of five runs or the best performance of five runs. The negative effect of this algorithm described previously disappears for solving such large size instances. 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 deterministic rule is more likely to produce good results in a short computational time for large size instances.

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

Algorithm	Deviation (%)		average CPU sec.
	min	average	
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

5. CONCLUSION

This paper presents two types of local-search based algorithms for the multi-source capacitated facility location problem. One type uses SA framework and the other type uses variable neighborhood search 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 [5]. Also, a VNLS with random rule and tabu list obtains slightly better performance than the hybrid SA with either random rule or deterministic rule. Algorithms VNLS-tb-rule works very well in terms of solution searching effectiveness and efficiency for problems of very large size. Our experiments indicate that the hybrid SAs are appropriate for solving small to medium size instances, while the VNLSs are more suitable for solving very large size instances.

The capacitated facility location problems are NP-hard, no matter it is single-source or multi-source. However, the multi-source problem is easier than the single-source problem in terms of computational complexity, and has smaller optimal cost. Thus, solving the multi-source case is more practical than solving the single-source case, unless it is specified that the single-source condition must be strictly enhanced.

References

[1] Lenstra, J.K., Rinnooy Kan, A.H.G., Complexity of Scheduling under Precedence Constraints, *Operations Research*, Vol. 26, No. 1, 22-35, 1978.

[2] Pirkul, H., Efficient Algorithms for Capacitated Concentrator Location Problem, *Computers and Operations Research*, Vol. 14, No. 3, 197-208, 1987.

[3] Sridharan, R., A Lagrangian heuristic for the capacitated plant location problem with single source constraints, European *Journal of Operational Research*, Vol. 66, No. 3, 305-312, 1993.

[4] Beasley, J.E., Lagrangian heuristics for location problems, *European Journal of Operational Research*, Vol. 65, No. 3, 383-399. 1993

[5] Holmberg, K., Rönnqvist, M., and Yuan, D, An exact algorithm for the capacitated facility location problems with single sourcing, *European Journal of Operational Research*, Vol. 113, No. 3, 544-559, 1999.

[6] Cortinhal, M.J., and Captivo, M.E., Upper and lower bounds for the single source capacitated location problem, *European Journal of Operational Research*, Vol. 151, No. 2, 333-351, 2003.

[7] Arostegui, Jr., M.A., Kadipasaoglu, S.N., and Khumawala, B.M., An empirical comparison of Tabu Search, Simulated Annealing, and Genetic Algorithms for facilities location problems, *International Journal of Production Economics*, Vol.103, No. 2, 742-754, 2006.

[8] Ahuja, R.K., Liu, J., Orlin, J.B., Goodstein, J., and Mukherjee, A., A neighborhood search algorithm for the combined through and fleet assignment model with time windows, *Networks*, Vol.44, No. 2, 160-171, 2004.

[9] Sridharan, R., The capacitated plant location problem, European *Journal of Operational Research*, Vol. 87, No. 2, 203-213, 1995.

[10] Korupolu, M.R., Plaxton, C.G., Rajaraman, R., Analysis of a Local Search Heuristic for Facility Location Problems, *Journal of Algorithms*, Vol.37, No. 1, 146-188, 2000.

[11] Barahona, F., Chudak, F.A., Near-optimal solutions to large-scale facility location problems, *Discrete Optimization*, Vol.2, No. 1, 35-50, 2005.

[12] Barahona, F., Anbil, R., The volume algorithm: Producing primal solutions with a subgradient method, *Mathematical Programming, Series B* 87, No. 3, 385-399, 2000.

[13] Klose, Drexl, Lower bounds for the capacitated facility location problem based on column generation, *Management Science*, Vol. 51, No. 11, 1689-1705, 2005.

[14] Aart, E.H.L., J.H.M., Korst, Simulated Annealing Boltzmann Machines: A Stochastic Approach to Combinatorial Optimization and Neural Computing, John Wiley, Chichester.

Author Biographies

Wei-Shung Chang is a Ph.D. student in the Department of Industrial Engineering and Management, Yuan-Ze University, Taiwan. His research interests include Supply Chain Management, PCB Assembly, and Production Planning and Scheduling.

Pong-Yeng Lei is a M.S. student in the department of Industrial Engineering and Management, Yuan-Ze University, Taiwan. His research interests include Supply Chain Management and Metaheuristics.

Chiuh-Cheng Chyu is an associate professor in the Department of Industrial Engineering and Management at Yuan-Ze University in Taiwan. His research interests include Applications of Operations Research, Bayesian Analysis, and Network Optimization.