The Scientific World Journal
Volume 2012, Article ID 576392, 7 pages
doi:10.1100/2012/576392

The ScientificWorldJOURNAL

Research Article

Combined Simulated Annealing Algorithm for the Discrete Facility Location Problem

Jin Qin,[1] Ling-lin Ni,[1,2] and Feng Shi[1]

[1] *School of Traffic and Transportation Engineering, Central South University, Changsha 410075, China*
[2] *Business Administration College, Zhejiang University of Finance & Economics, Hangzhou 310018, China*

Correspondence should be addressed to Jin Qin, csu_qinjin@hotmail.com

Received 21 May 2012; Accepted 28 June 2012

Academic Editors: C. W. Ahn, B. Alatas, P. Bala, P.-A. Hsiung, and Y. Jiang

The combined simulated annealing (CSA) algorithm was developed for the discrete facility location problem (DFLP) in the paper. The method is a two-layer algorithm, in which the external subalgorithm optimizes the decision of the facility location decision while the internal subalgorithm optimizes the decision of the allocation of customer's demand under the determined location decision. The performance of the CSA is tested by 30 instances with different sizes. The computational results show that CSA works much better than the previous algorithm on DFLP and offers a new reasonable alternative solution method to it.

1. Introduction

The classical facility location problem (FLP) is one of the most important models in combinatorial optimization, which is to determine the number and locations of the facilities and allocate customers to these facilities in such a way that the total cost is minimized. The FLP may be the most critical and most difficult decision in the designing of an efficient supply chain for the facilities are costly and difficult to reverse after being located. The problem is also encountered in other areas such as material distribution, transportation network, and telecommunication network.

The FLP can be classified in two categories as discrete problem and continuous problem according to whether the sets of demand points and facility locations are finite. The discrete facility location problem (DFLP) assumes that the solution space is discrete and generally the facilities are located on the nodes of the network, which brings a lot of complexities to the problem. And many practical problems without facilities to locate, such as cluster analysis, machine scheduling, economic lot sizing, portfolio management, and computer network design, can also be modeled as DFLP [1]. Due to its strategic nature, DFLP has been widely studied by researchers over many years, especially developing solution methods for the DFLP has been a hot topic of research for the last 30 years. Many successful contributions of the DFLP have been reported both in theory and in practice. The Daskin [2] and Melo et al. [1] provided thorough reviews of the DFLP.

The DFLP is a NP-hard problem, and this nature makes the exact algorithms only for small problems and heuristics the natural choice for larger instances. Therefore much attention has been focused on designing heuristics algorithms with good performance. Following the first heuristics algorithm presented by Shmoys et al. [3], there was a long list of work on designing heuristics algorithms for this problem over the years. As a result, four basic algorithms with different features have been emerged, namely, LP-rounding [3–8], primal-dual [9–11], dual-fitting [12–14], and local search [15–18]. Although LP-rounding could be used to design algorithm with better results than other three, it is noncombinatorial in nature and needs more CPU time. Primal-dual and dual-fitting method could be adapted to solve variants of the FLP but are less robust. Recently some applications have proved that local search is more powerful on the hard DFLP [16]. And along with the development of the computation method, more and more researchers use the heuristic algorithms based on the local search to solve the problem, such as simulated annealing [19–21] and genetic algorithm [20, 22].

In this paper, a general algorithmic framework of combined simulated annealing (CSA) is developed for the DFLP,

and its performance is compared with other existed solution methods.

The rest of this paper is organized as follows. In Section 2, we describe the general model of DFLP. The basic procedures of the SA and CSA are presented in Sections 3 and 4 respectively. Section 5 shows the computational results on test instances, and Section 6 concludes the paper.

2. The Formulation of DFLP Model

We start by giving the mathematical formulation of a general model for DFLP with a minimized objective function. In the model, $I = \{1, 2, \ldots, n\}$ is used to represent the set of the customers and $J = \{1, 2, \ldots, m\}$ is used to represent the set of the potential location sites. f_j is used to represent the fixed cost of opening a facility at site $j \in J$, v_j is the service capacity of facility at site j, d_i is the demand of customer $i \in I$, and c_{ij} is the cost of serving one unit of demand at customer i from site j, in other word, the unit variable shipping cost between customer i from site j. We reasonably assume that $c_{ij} \geq 0$, $f_j \geq 0$, and $v_j \geq 0$ for all $i \in I$, for all $j \in J$.

And two binary variables are set as:

$$X_j = \begin{cases} 1 & \text{if a facility setup on site } i; \\ 0 & \text{otherwise;} \end{cases}$$
$$Y_{ij} = \begin{cases} 1 & \text{if facility } j \text{ serves customer } i; \\ 0 & \text{otherwise.} \end{cases} \tag{1}$$

Then the general model of DFLP could be stated as the following linear mixed-integer program:

$$\min \quad \Phi = \sum_{j=1}^{m} f_j X_j + \sum_{i=1}^{n} \sum_{j=1}^{m} d_i c_{ij} Y_{ij} \tag{2}$$

$$\text{Subject to} \quad \sum_{j=1}^{m} Y_{ij} = 1 \quad \forall i \in I \tag{3}$$

$$\sum_{i=1}^{n} d_i Y_{ij} \leq v_j X_j \quad \forall j \in J \tag{4}$$

$$Y_{ij} - X_j \leq 0 \quad \forall i \in I, \ \forall j \in J \tag{5}$$

$$X_j \in \{0, 1\} \quad \forall j \in J \tag{6}$$

$$Y_{ij} \in \{0, 1\} \quad \forall i \in I, \ \forall j \in J. \tag{7}$$

The objective function (2) is to minimize the total system cost, including the location cost and the shipment cost. Constraint (3) is the demand constraint, which makes the demand of each customer be met; (4) is the variable upper bound constraint; (5) is the capacity constraint of facility; (6) and (7) are standard binary integrality constraints.

3. Simulated Annealing Algorithm for DFLP

Kirkpatrick et al. [23] introduced the concept of simulated annealing (SA) algorithmin 1983, which is a stochastic optimization technique. To be specific, SA is a probabilistic heuristic for the global optimization problems of finding a good approximation to the global optimum of a given objective function in the search space. It is often used when the solution space is discrete. In the searching process, the SA accepts not only better but also worse neighboring solutions with a certain probability. This means that the SA has the ability to escape from the local minima. Therefore, it can find high-quality solutions that do not strongly depend upon the choice of the initial solution compared to other local search algorithms. And its another advantage over the other heuristic algorithms is the ease of implementation. So we adopted SA as the basic solution method to solve the DFLP.

In last 30 years, SA has been studied widely and used extensively in many optimization problems [24–29], which have proved that SA is an effective tool for approximating globally optimal solutions to many NP-hard optimization problems.

In order to describe the procedure of the SA, $\mathbf{S}$, $\mathbf{S}'$, $\mathbf{S}''$, and $\overline{\mathbf{S}}$ are used to represent the different feasible solutions of the model; $D(T_i)$ is the cooling function of temperature, in which T_i is the current temperature value. T_f is the stop temperature value. N is the maximum iteration number at each temperature value. $\Phi(\mathbf{S})$ is used to represent the objective function value of the solution $\mathbf{S}$. According to Jayaraman and Ross [19], the SA for DFLP could be given as follows.

Step 1 (initialization). Set iteration counter $i = 1$. Generate an initial feasible solution $\mathbf{S}$ and regard $\mathbf{S}$ as the optimal solution. Set the initial temperature T_i and the stop temperature T_f are specified. Define the cooling function $D(T_i)$.

Step 2 (generate a feasible neighboring solution). Perform the neighboring function on current solution $\mathbf{S}$ and get the new neighboring solution $\mathbf{S}'$.

Step 3 (evaluate current solution with neighboring solution). If the objective function value of the new solution $\mathbf{S}'$ is no less than that of the current solution $\mathbf{S}$, namely, $\Phi(\mathbf{S}') \geq \Phi(\mathbf{S})$, then proceed to Step 4; otherwise, if $\Phi(\mathbf{S}') < \Phi(\mathbf{S})$, then $\mathbf{S} = \mathbf{S}'$, proceed to Step 5.

Step 4 (examine metropolis condition). Determine the difference ΔC between the incumbent solution $\mathbf{S}$ and the neighboring solution $\mathbf{S}'$, as $\Delta\Phi = \Phi(\mathbf{S}') - \Phi(\mathbf{S})$. Generate a random number ρ from the interval (0,1), if $\rho < \exp(-\Delta\Phi/T_i)$, then $\mathbf{S} = \mathbf{S}'$. Proceed to Step 5.

Step 5 (check increment counters). Set $i \leftarrow i + 1$. If $i \leq N$, then return to Step 2. Otherwise proceed to Step 6.

Step 6 (adjust temperature). Adjust temperature by the cooling function, Mathematically this is $T_i \leftarrow D(T_i)$.

Step 7 (convergence check). If $T_i \geq T_f$, then reset $i = 1$ and return to Step 2. Otherwise, stop and output the optimal solution $\mathbf{S}$.

4. Combined Simulated Annealing (CSA)

The solution of DFLP includes two parts: X_js and Y_{ij}s. X_j denotes whether or not open the facility at site j, while Y_{ij} denotes the service demand allocation. The two variables are interdependent and interactional. As each demand must allocate to an opening facility, we could can conclude that the variable Y_{ij} is subject to the variable X_j. This relation also can be seen from the constrain (4) in the model in Section 1.

CSA works in two layers as internal layer and external layer to solve the problem. The internal layer subalgorithm (ILSA) would optimize the facility location decision variable X_j. The external layer subalgorithm (ELSA) would perform the allocation optimization under fixed X_js which determined in the internal layer. In each layer the SA is used and they make up of the CSA. The method that divides the problem into two layers could make the search in the procedure explores in smaller solution space each time, so it increases the probability of obtaining the global optimal solution.

Some parameters should be initialized before the performance of CSA, including the initial temperature and stop temperature, cooling ruler of the temperature, iteration maximum in each temperature. The initialization of the parameters could be determined in similar ways to these in the SA. And the $\mathbf{S}$, $\mathbf{S}'$, $\mathbf{S}''$, $\overline{\mathbf{S}}$ are also used to represent the different feasible solutions here.

In addition, CSA must start with an initial solution or with a solution produced using a heuristic. In this work, we use the randomly generated initial solution, which proposed in Qin et al. [30].

4.1. Neighboring Functions. Similar to the SA, the CSA algorithm is an iterative search procedure based on the neighboring function. The quality of the optimal solution is very sensitive to the way that the candidate solutions are selected. Thus, the neighboring function is crucial to the good performance of the CSA algorithm.

The ELSA would optimize the facility location decision. So the neighboring function of the ELSA to modify the configuration of the current solution and generate a neighboring solution could perform three different operations:

(1) If the number of the located facilities is less than the allowed maximum M ($\sum_{j=1}^{m} X_j < M$), then select a candidate site i which satisfies $X_j = 0$ in current solution S randomly and set $X_j = 1$, namely, there would locate a new facility.

(2) If the number of the locate facility no less than 1 ($\sum_{j=1}^{m} X_j \geq 1$), then select a site j randomly which satisfies $X_j = 1$ in solution S and set $X_j = 0$, namely, there would close a open facility.

(3) Select a site j which satisfies $X_j = 0$ and another site j' which satisfies $X_{j'} = 1$ in current solution S, then set $X_j = 1$ and $X_{j'} = 0$.

In the implementation of the ELSA, we could select only one operation from the above three operations to perform the neighboring function each time. And after implementing the operation, it should allocate the customers' demand to the opened facilities again, as generate an initial solution again.

ILSA is to determine the demand allocation decision. According to the features of the allocation decision, there are two operations to generate the neighboring solutions in the ILSA.

(1) Select two allocation variables Y_{ij} and $Y_{ij'}$ that satisfy $Y_{ij} = 1$ and $Y_{ij'} = 0$; then set $Y_{ij'} = 1$, $Y_{ij} = 0$, namely, it allocates the demand of customer i from facility j to facility j'.

(2) Exchange the facilities which serve two customers re spectively with each other. To be specific, select four allocation variables as $y_{i_1 j_1} = 1$, $y_{i_2 j_2} = 1$, $y_{i_1 j_2} = 0$, $y_{i_2 j_1} = 0$, then set $y_{i_1 j_2} = 1$, $y_{i_2 j_1} = 1$, $y_{i_1 j_1} = 0$, $y_{i_2 j_2} = 0$.

Similarly, the neighboring function of the ILSA could select only one operation to perform each time. In addition, it must ensure that demands of all customers must be satisfied and the facilities have no capacity violations exist. Otherwise, it should return to the old solution and reselect an operation to perform.

4.2. The Procedure of CSA. The following is a step-by-step description of the procedure of CSA.

Step 1 (initialization). Set iteration counter $i = 1$, $k = 1$. Set the initial temperature T_i, and stop temperature T_f, the initial feasible solution is $\mathbf{S}$, and let $\mathbf{S}$ be the optimal solution at the same time. Define the cooling function $D(T_i)$. Generate a feasible solution by randomly allocation with capacity restricted.

Step 2 (check feasibilities). The method now checks the demand allocations to ensure that no capacity violations exist. The demands of the customers are also checked. If the solution $\mathbf{S}$ is not feasible, we should return to Step 1.

Step 3 (generate a neighboring solution). Perform the external layer neighboring function on solution $\mathbf{S}$, and generate a neighboring feasible solution $\mathbf{S}'$.

Step 4 (perform ILSA). *Step* 4.1. Set $k = 1$. Regard solution $\mathbf{S}'$ as the initial solution and the current optimal solution.

Step 4.2. Perform the internal layer neighboring function on solution $\mathbf{S}'$, and generate a neighboring feasible solution $\mathbf{S}''$.

Step 4.3. If $\Phi(\mathbf{S}'') < \Phi(\mathbf{S}')$, then set $\mathbf{S}' = \mathbf{S}''$; otherwise, generate a random number ρ from (0,1), if $\rho < \exp(-(\Phi(\mathbf{S}'') < \Phi(\mathbf{S}'))/T_i)$, then set $\mathbf{S}' = \mathbf{S}''$.

Step 4.4. Consider $k \leftarrow k + 1$. If $k \leq N_2$, then return to Step 2. Otherwise proceed to Step 5.

Step 4.5. Stop and output the optimal solution $\mathbf{S}'$, return to the ELSA.

Step 5 (save the global optimal solution). If $\Phi(\mathbf{S}') < \Phi(\overline{\mathbf{S}})$, then $\overline{\mathbf{S}} = \mathbf{S}'$.

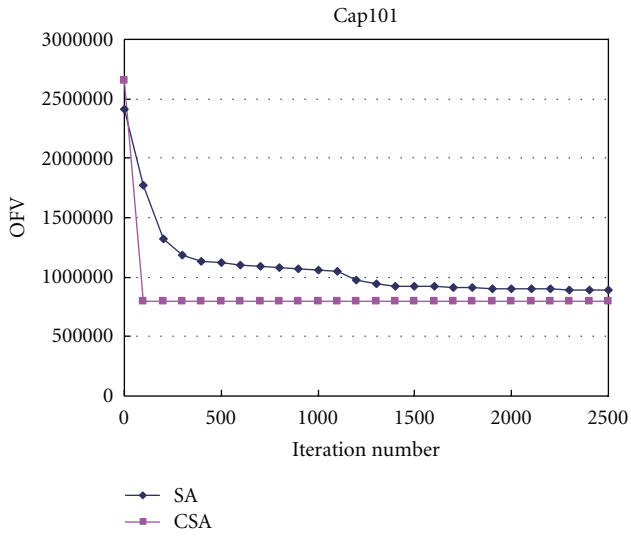

Figure 1: OFV versus iteration number in Cap101.

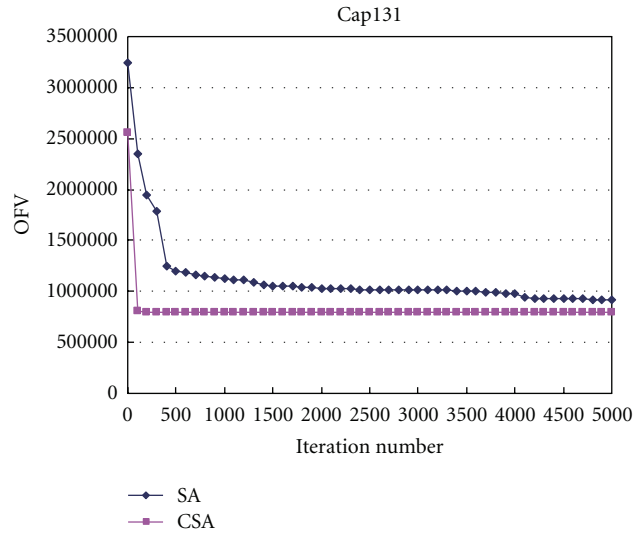

Figure 2: OFV versus iteration number in Cap131.

Step 6 (evaluate current solution). Evaluate current solution with neighboring solution. If $\Phi(\mathbf{S}') < \Phi(\mathbf{S})$, then let $\mathbf{S} = \mathbf{S}'$ proceed to Step 7; otherwise, proceed to Step 7.

Step 7 (examine metropolis condition). A random number ρ is generated from (0,1), if $\rho < \exp(-(\Phi(\mathbf{S}') - \Phi(\mathbf{S}))/T_i)$ then $\mathbf{S} = \mathbf{S}'$. Proceed to Step 8.

Step 8 (increment counters). Set $i \leftarrow i + 1$. If $i \leq N_1$, then return to Step 3; otherwise, proceed to Step 9.

Step 9 (adjust temperature). Adjust temperature by the cooling function: $T_i \leftarrow D(T_i)$.

Step 10 (convergence check). If $T_i \geq T_f$, then reset $i = 1$ and return to Step 3. Otherwise, stop and output the optimal solution $\overline{\mathbf{S}}$.

N_1, N_2 are the given maximum iteration number in ELSA and ILSA respectively. The Step 5 is to save the global optimal solution that has been found so far in the CSA. This operation does not take the acceptance probability of worse solution into consideration. So it could help the algorithm avoid losing the global optimal solution.

5. Computational Experiments

To assess the practical effectiveness of the proposed CSA algorithm, we use 30 instances with different sizes as benchmark problems. Twelve "Capacitated warehouse location" instances were given by Beasley [31], which are publicly available from the OR-Library and could be downloaded directly from the website: http://people.brunel.ac.uk/~mastjjb/jeb/orlib/capinfo.html. The other 18 instances were proposed by Ghosh [32].

To perform the CSA, the initial temperature T_1 could be set equal to the objective function value of the initial solution. The cooling ruler is equal-ratio cooling, and the cooling ratio $\alpha = 0.95$. The iteration maximum $N_1 = m$, $N_2 = 5n$, the stop temperature value $t_f = 0.001$. The SA is used to solve the instances too, and its iteration maximum under the same temperature is $5mn$; other parameters are same as CSA. So the total iteration numbers in the SA and CSA are equal. We also use the randomly generated method to find an initial feasible solution.

The model and CSA were implemented in Visual C# 2010. A personal computer with Intel E5800 CPU, 2G RAM and Windows XP Profession operating system was used for all tests.

Example 1 (OR-Library Instances). The instances in OR-Library are more than 150, and we selected 12 instances with different size to be solved by the CSA and compared with theirs optimal solution (all instances have been exactly solved by the Lindo software and provide their optimal solutions by the author).

Computational results of these OR-Library instances are reported in Table 1. Each row of the table gives the results of one individual instance. The instance names are originally used in the OR-Library. The gap in the table represented the relative error between the result and the optimal solution.

It can be observed from Table 1 that the CSA found optimal solutions for all these instances without any exception, while the SA didn't find anyone. The CPU time used by SA for each instance is from 3 to 8 times of that used by CSA.

The computation process of instances Cap101 and Cap131 with CSA and SA are depicted in Figures 1 and 2. The OFV is the objective function value. As shown in the figures, we can find that the convergence speed of CSA is more quickly than that of SA. The iteration time for converging to the optimal solution used by SA for each instance is from 10 to 25 times of that used by CSA.

Example 2 (The Ghosh Instances). The Ghosh instances are 90 instances in total, and with $n = m$ on all cases. These instances of the same size are divided into two categories,

Table 1: Comparison for OR-Library instances.

Instances			SA			CSA		
Name	Size ($m \times n$)	Optimal solution	Result	Gap (%)	CPU time$^{(S)}$	Result	Gap (%)	CPU time$^{(S)}$
Cap71	16×50	932615.750	1192413.575	6.995	4.233	932615.750	0	0.921
Cap72	16×50	977799.400	1012970.190	5.970	4.690	977799.400	0	1.599
Cap73	16×50	1010641.450	1100470.190	8.179	4.435	1010641.450	0	0.732
Cap74	16×50	1034976.975	1212970.191	12.276	6.438	1034976.975	0	1.499
Cap101	25×50	796648.440	886494.475	9.250	12.552	796648.440	0	2.192
Cap102	25×50	854704.200	998953.625	15.698	13.652	854704.200	0	2.612
Cap103	25×50	893782.1125	1002291.150	11.412	12.002	893782.1125	0	1.798
Cap104	25×50	928941.750	1177291.150	24.333	18.263	928941.750	0	5.048
Cap131	50×50	793439.5620	856571.455	6.607	54.688	793439.562	0	24.723
Cap132	50×50	851495.3250	1011571.450	16.732	45.563	851495.325	0	19.803
Cap133	50×50	893076.7120	1219071.450	32.485	56.549	893076.712	0	8.842
Cap134	50×50	928941.7500	1374071.452	41.690	49.156	928941.750	0	14.972

Table 2: Comparison for the Ghosh instances.

Size ($m \times n$)	Instances		Hybrid		CLM		GTS		TS		CSA	
	Class	Range	Result	CPU time[1]	Result	CPU time[1]	Result	CPU time[1]	Result	CPU time[1]	Result	CPU time
250×250	Symmetry	A	257806.8	4.328	257895.2	86.482	257832.6	2.828	257805.0	3.687	**257293.0**	66.652
250×250	Symmetry	B	276035.2	7.774	276352.2	34.634	276185.2	5.628	276035.2	5.347	**275560.0**	69.339
250×250	Symmetry	C	***333671.6***	8.702	***333671.6***	69.458	333820.0	9.878	***333671.6***	10.384	334127.0	70.588
250×250	Asymmetry	A	257923.4	4.636	258032.6	86.506	257978.4	2.618	257917.8	3.487	**257179.0**	81.706
250×250	Asymmetry	B	276053.2	8.082	276184.2	33.688	276467.2	5.790	276053.2	5.501	**275111.0**	75.473
250×250	Asymmetry	C	332897.2	7.776	333058.4	89.990	333237.6	9.196	**332897.2**	9.736	343270.0	70.329
500×500	Symmetry	A	511196.4	27.644	511487.2	946.028	511383.6	15.616	511180.4	14.835	**511113.0**	541.772
500×500	Symmetry	B	***537912.0***	34.196	538685.8	294.656	538480.4	31.432	***537912.0***	29.860	***537912.0***	579.244
500×500	Symmetry	C	621059.2	40.376	621172.8	437.462	621107.2	71.106	621059.2	67.551	**620112.0**	663.167
500×500	Asymmetry	A	511145.0	20.232	511393.4	921.208	511251.6	13.760	511140.0	13.072	**511097.0**	543.235
500×500	Asymmetry	B	537863.4	31.300	538421.0	311.344	538144.0	34.748	537847.6	33.011	**535665.0**	358.807
500×500	Asymmetry	C	621463.8	47.790	621990.8	388.210	621811.8	72.064	**621463.8**	68.461	630861.0	509.046
750×750	Symmetry	A	763706.6	49.214	763978.0	3650.662	763830.8	39.812	763693.4	37.821	**763417.0**	1250.775
750×750	Symmetry	B	796632.2	92.886	797173.4	1583.170	796919.0	93.352	796571.8	88.684	**794970.0**	1517.086
750×750	Symmetry	C	900272.0	113.640	900785.2	1194.534	901158.4	229.914	**900158.6**	218.418	926358.0	1387.104
750×750	Asymmetry	A	763731.2	59.130	764019.2	3658.588	763836.6	39.650	763717.0	34.667	**763498.0**	1548.608
750×750	Asymmetry	B	796396.8	73.322	796754.2	1606.778	796859.0	95.430	796374.4	90.660	**793668.0**	1598.631
750×750	Asymmetry	C	**900193.2**	112.994	900349.8	1325.812	900514.2	236.902	**900193.2**	205.060	919453.0	1337.671

Note: CPU time[1] is on Sun Enterprise 3000 Server ($4 \times$ CPU, 6 G memory).

symmetric, where the transportation cost $c_{ij} = c_{ji}$ holds, and asymmetric, where $c_{ij} = c_{ji}$ does not necessarily hold. Each group contains three values of $m \times n$: 250×250, 500×500, and 750×750. The instances in each category are further divided into three instance classes, and they differ in the range of values from which opening costs f_j are drawn, which could be chosen from the $[100, 200]$ (Range A), $[1000, 2000]$ (Range B), and $[10000, 20000]$ (Range C) randomly.

The optimal solutions of the Ghosh instances were not presented, but there used to be several methods to solve the instances and compared the results with each other in the literature [33]. And we compare the results of the CSA with them as reported in Table 2, in which we reported the results of the instances obtained by hybrid, CLM, GTS, TS, and CSA.

As shown in Table 2, for all instances, CSA found solutions better than those of other methods with only five exceptions, and found solutions with the same result for symmetric instance with $m \times n = 500 \times 500$ in type C. Even in the five exceptions, the gaps between the solutions were found by CSA and the best solutions found by other methods are very small, and the maximal relative gap is only 3.1% in asymmetric instance with $m \times n = 250 \times 250$ in type C.

6. Conclusions

The DFLP is to determinate the facility location and demand allocation so as to minimize the total cost, based on the demands of customers satisfied without violating the

capacity restriction of any facility. The CSA was proposed for the DFLP, which is a two-layer algorithm. Its external layer subalgorithm optimizes the facility location decision, while the internal layer subalgorithm optimizes the demand allocation under the fixed facility location which is determined by the external layer. The each local search process in CSA focuses on the smaller solution space, which could not only increase the probability of obtaining the global optimal solution, but also save computational time.

The performance of CSA is evaluated with 30 instances with different sizes. Those instances are solved by CSA in a very reasonable amount of time, and the solutions are compared with that of previous studies in the literature. It is showed that the new algorithm could give better results (or at least same) than the others for nearly all instances. Hence, the CSA works much better than the previous work and offers a reasonable alternative solution method to the DFLP.

Acknowledgment

This research was supported by the National Natural Science Foundation of China (Grant no. 71101155).

References

[1] M. T. Melo, S. Nickel, and F. Saldanha-da-Gama, "Facility location and supply chain management—a review," *European Journal of Operational Research*, vol. 196, no. 2, pp. 401–412, 2009.

[2] M. S. Daskin, "What you should know about location modeling," *Naval Research Logistics*, vol. 55, no. 4, pp. 283–294, 2008.

[3] D. B. Shmoys, E. Tardos, and K. Aardal, "Approximation algorithms for facility location problems," in *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, pp. 265–274, ACM Press, May 1997.

[4] A. F. Gabor and J. K. C. W. van Ommeren, "A new approximation algorithm for the multilevel facility location problem," *Discrete Applied Mathematics*, vol. 158, no. 5, pp. 453–460, 2010.

[5] Z. Wang, D. Du, A. F. Gabor, and D. Xu, "An approximation algorithm for the κ-level stochastic facility location problem," *Operations Research Letters*, vol. 38, no. 5, pp. 386–389, 2010.

[6] L. Yan and M. Chrobak, "Approximation algorithms for the Fault-Tolerant Facility Placement problem," *Information Processing Letters*, vol. 111, no. 11, pp. 545–549, 2011.

[7] A. Marín, "The discrete facility location problem with balanced allocation of customers," *European Journal of Operational Research*, vol. 210, no. 1, pp. 27–38, 2011.

[8] Y. Li, D. Xub, and D. Du, "Improved approximation algorithms for the robust fault-tolerant facility location problem," *Information Processing Letters*, vol. 112, no. 10, pp. 361–364, 2012.

[9] D. Fotakis, "A primal-dual algorithm for online non-uniform facility location," *Journal of Discrete Algorithms*, vol. 5, no. 1, pp. 141–148, 2007.

[10] J. Dias, M. Eugénia Captivo, and J. Clímaco, "Efficient primal-dual heuristic for a dynamic location problem," *Computers and Operations Research*, vol. 34, no. 6, pp. 1800–1823, 2007.

[11] D. Du, R. Lu, and D. Xu, "A primal-dual approximation algorithm for the facility location problem with submodular penalties," *Algorithmica*, vol. 63, no. 1-2, pp. 191–200, 2012.

[12] K. Jain, M. Mahdian, and A. Saberi, "A new greedy approach for facility location problems," in *Proceedings of the 34th Annual ACM Symposium on Theory of Computing (STOC '02)*, pp. 731–740, May 2002.

[13] K. Jain, M. Mahdian, E. Markakis, A. Saberi, and V. V. Vazirani, "Greedy facility location algorithms analyzed using dual fitting with factor-revealing LP," *Journal of the ACM*, vol. 50, no. 6, pp. 795–824, 2003.

[14] M. Mahdian, Y. Ye, and J. Zhang, "Approximation algorithms for metric facility location problems," *SIAM Journal on Computing*, vol. 36, no. 2, pp. 411–432, 2006.

[15] S. Guha and S. Khuller, "Greedy strikes back: improved facility location algorithms," *Journal of Algorithms*, vol. 31, no. 1, pp. 228–248, 1999.

[16] J. Zhang, B. Chen, and Y. Ye, "A multiexchange local search algorithm for the capacitated facility location problem," *Mathematics of Operations Research*, vol. 30, no. 2, pp. 389–403, 2005.

[17] J. K. Sankaran, "On solving large instances of the capacitated facility location problem," *European Journal of Operational Research*, vol. 178, no. 3, pp. 663–676, 2007.

[18] V. P. Nguyen, C. Prins, and C. Prodhon, "A multi-start evolutionary local search for the two-echelon location routing problem," *Engineering Applications of Artificial Intelligence*, vol. 25, no. 1, pp. 56–71, 2012.

[19] V. Jayaraman and A. Ross, "A simulated annealing methodology to distribution network design and management," *European Journal of Operational Research*, vol. 144, no. 3, pp. 629–645, 2003.

[20] M. A. Arostegui Jr., S. N. Kadipasaoglu, and B. M. Khumawala, "An empirical comparison of Tabu Search, Simulated Annealing, and Genetic Algorithms for facilities location problems," *International Journal of Production Economics*, vol. 103, no. 2, pp. 742–754, 2006.

[21] R. Şahin, "A simulated annealing algorithm for solving the bi-objective facility layout problem," *Expert Systems with Applications*, vol. 38, no. 4, pp. 4460–4465, 2011.

[22] M. Solimanpur and M. A. Kamran, "Solving facilities location problem in the presence of alternative processing routes using a genetic algorithm," *Computers & Industrial Engineering*, vol. 59, no. 4, pp. 830–839, 2010.

[23] S. Kirkpatrick, C. D. Gelett, and M. P. Vecchi, "Optimization by simulated annealing," *Science*, vol. 220, no. 4598, pp. 671–680, 1983.

[24] D. Bertsimas and J. Tsitsiklis, "Simulated annealing," *Statistical Science*, vol. 8, no. 1, pp. 10–15, 1993.

[25] N. Boissin and J. L. Lutton, "A parallel simulated annealing algorithm," *Parallel Computing*, vol. 19, no. 8, pp. 859–872, 1993.

[26] P. Tian, J. Ma, and D. M. Zhang, "Application of the simulated annealing algorithm to the combinatorial optimization problem with permutation property: an investigation of generation mechanism," *European Journal of Operational Research*, vol. 118, no. 1, pp. 81–94, 1999.

[27] B. Suman, N. Hoda, and S. Jha, "Orthogonal simulated annealing for multiobjective optimization," *Computers & Chemical Engineering*, vol. 34, no. 10, pp. 1618–1631, 2010.

[28] Z. Xinchao, "Simulated annealing algorithm with adaptive neighborhood," *Applied Soft Computing Journal*, vol. 11, no. 2, pp. 1827–1836, 2011.

[29] R. S. Tavares, T. C. Martins, and M. S. G. Tsuzuki, "Simulated annealing with adaptive neighborhood: a case study in off-line robot path planning," *Expert Systems with Applications*, vol. 38, no. 4, pp. 2951–2965, 2011.

[30] J. Qin, F. Shi, L. X. Miao, and G. J. Tan, "Optimal model and algorithm for multi-commodity logistics network design considering stochastic demand and inventory control," *System Engineering*, vol. 29, no. 4, pp. 176–183, 2009.

[31] J. E. Beasley, "Lagrangean heuristics for location problems," *European Journal of Operational Research*, vol. 65, no. 3, pp. 383–399, 1993.

[32] D. Ghosh, 2001, http://www.mpi-inf.mpg.de/departments/d1/projects/benchmarks/UflLib/ORLIB.html.

[33] D. Ghosh, "Neighborhood search heuristics for the uncapacitated facility location problem," *European Journal of Operational Research*, vol. 150, no. 1, pp. 150–162, 2003.