Journal of Industrial Engineering 2 (2009) 61-67

The Capacitated Location-Allocation Problem with Interval Parameters

Hassan Shavandi

Department of Industrial Engineering, Sharif University of Technology, Tehran, Iran, shavandi@sharif.edu

Received 3 Dec., 2008; revised 15 Jan., 2009; accepted 12 Feb. 2009

Abstract

In this paper, we develop a capacitated location-covering model considering interval values for demand and service parameters. We also consider flexibility on distance standard for covering demand nodes by the servers. We use the satisfaction degree to represent the constraint of service capacity. The proposed model belongs to the class of mixed integer programming models. Our model can be reduced to the p-median problem in polynomial time so it is NP-Hard. A genetic algorithm is proposed to solve the developed model and experimental results of solving the model are presented.

Keywords: Location, Capacitated, Covering, Interval Parameters, Genetic Algorithms;

1. Introduction

Numerous models have been developed in the area of location-allocation (LA) problems so far. LA models can be divided into discrete and continues models. Our orientation in this paper is on discrete location-allocation models. For a short review of discrete location models, among the different classifications of discrete location-allocation models we adopt the one, which is proposed by Current et al. [4]. They introduced eight basic facility location models, which are set covering, maximal covering, *p*-center, *p*-dispersion, *p*-median, fixed charge, hub, and maxi-sum. Each of these models attempts to optimize a specific objective function by locating the new facilities within a network. All these models can be developed with multi-objective function, or with dynamic assumption, and with stochastic structure.

Since our model belongs to the class of maximal covering model, a short review of related models in deterministic and stochastic version is in order. Toregas et al. [16], developed the location set covering problem (LSCP) that is a version of set covering problem. This model was an attempt to locate the least number of servers to cover all the demand nodes with at least one server within the time or distance standard. This model's main drawback was its unrealistic assumption of unlimited budget that leads to the full coverage of all nodes. This led to emergence of the Maximal Covering Location Problem (MCLP) by Church and ReVelle [2]. This model sought to maximize the population of calls, which have a server within the time or distance standard while imposing a limit on the number of servers. As such, not all nodes receive coverage.

At first, let us examine uncapacitated LA problem (see [1], [2], [3]) in which the capacities of servers are limitless. In this case, it is easy to know that the nearest server should cover each demand node. Gradually, more researchers investigated capacitated LA problem (see [4], [5], [6]). Most of works were done for the deterministic case. However in practice many parameters such as, demand rate of demand nodes, are not deterministic and some stochastic and fuzzy programming models have been proposed for LA problem. Zhou [17], and Zhou and Liu [18], presented the expected cost minimization model, α-cost minimization model and probability maximization model for capacitated location-allocation problem with stochastic demands. Zhou and Liu [18], also presented the capacitated location-allocation model with fuzzy demands.

For the first time, Shavandi and Mahlooji [14], presented the idea of flexibility in the distance standard for covering through the fuzzy queuing location-allocation model. In this paper we want to develop the capacitated location-allocation model considering interval parameters as well as flexibility in distance standard for covering. We also propose a genetic algorithm to solving the developed model. There are many works in the literature for solution methods of capacitated location-allocation problem (see [6], [9]). Since our solution method is genetic algorithm, we refer the interested readers to ([11], [12], [13]).

2. . Preliminaries

In this section we present some definitions that are used in developing the model (Liu [10]).

Definition 1. Properties for two interval numbers $\hat{a}=[a^L, a^U]$ and $\hat{b}=[b^L, b^U]$ can be described as follows.

$$\hat{a}+\hat{b}=[a^L+b^L, a^U+b^U]$$

$$\hat{a}-\hat{b}=[a^L-b^U, a^U-b^L]$$

$$\lambda\hat{a}=\begin{cases}[\lambda a^L, \lambda a^U], & \lambda>0,\\ [\lambda a^U, \lambda a^L], & \lambda<0.\end{cases}$$

Definition 2. For two interval numbers $\hat{a}=[a^L, a^U]$ and $\hat{b}=[b^L, b^U]$, the satisfaction degree of $\hat{a}\le\hat{b}$, or the degree $\hat{a}$ dominate $\hat{b}$ can be defined as

$$P(\hat{a}\le\hat{b})=\frac{\max(0, b^U-a^L)-\max(0, b^L-a^U)}{(a^U-a^L)+(b^U-b^L)}$$

3. The Capacitated Interval Location Covering Problem (CILCP)

This section is devoted to the definition of the stages of building the model. At first, the problem definition is presented and then the stages to formulation the model will be presented.

3.1. *Problem Definition*

Suppose a network that the demand populations are present on the nodes, which we name demand nodes. The links between nodes are the available roads between the nodes and weighted by the shortest distance between nodes, which we call d_{ij}. There is a possibility to locate p facilities on the nodes of network to serve the demand nodes. The problem attempts to maximize the covering of demand nodes subject to limited service capacity and constraint on covering the nodes by standard covering distance. We assume that we can estimate the demand rate of demand nodes by interval numbers under uncertain situation. In service centers, there is flexibility on service rate and it can be stated as an interval number. In addition, we consider the flexibility on distance standard by a linear decay function. Therefore, to formulate the model according the above definition, the parameters, variables, and formulation process are presented in the following.

3.2. *The parameters and variables*

The following is a list of the parameters used in the model.

a_i: the population of node i (a crisp number)

$\hat{d}_i : (d_i^l, d_i^u)$: the demand call per unit time for service at node i (an interval number)

$\hat{C}_j : (C_j^l, C_j^u)$: the service capacity of server j per unit time (an interval number)

$\hat{D}_j : (D_j^l, D_j^u)$: the assigned demand calls to server j (an interval number)

α : the satisfaction degree of server capacity constraint inequality (a real number between zero and one)

λ_{ij} : the degree representing the distance between i and j (d_{ij}). it will take value 1 if d_{ij} be less than or equal to the distance standard, and zero when d_{ij} is more than the upper limit of distance standard. It will take a value between zero and one if d_{ij} is between distance standard and upper limit of it. λ_{ij} is calculated by a decay function that will be presented in section 3.2.

As far as the variables are concerned, our model incorporates the following types of variables which have to do with locating the servers and allocating the demand nodes.

X_{ij} : a 0-1 variable which assumes value 1 if node i is covered by server j, and 0 otherwise.

Y_j : a 0-1 variable ; it turns 1 if a server is located at node j and 0 otherwise.

3.3. *Model formulation*

According to the classical location covering models, the underlying assumption for node i to be covered by server j was that the distance from node i to server j must not exceed the distance standard. As such, a set called N_j was defined for each server j that includes any node whose distance from node j was less than or equal to the distance standard. The allocation variable X_{ij}, consequently, could assume value 1 only when node i is a member of set N_j. We want to consider the situation that node i can be covered by server j, if the distance between them is almost (approximately) less than or equal to the distance standard (flexibility in distance standard for covering). So to reach this aim we define λ_{ij} .

Let s, be the distance standard and u denotes the acceptable upper bound for the distance standard. Now, λ_{ij} can be calculated by a decay function as follow (see figure (1)) :

$$\lambda_{ij}=\begin{cases}0 & d_{ij}>u\\ \dfrac{u-d_{ij}}{u-s} & s\le d_{ij}<u\\ 1 & d_{ij}\le s\end{cases} \qquad (1)$$

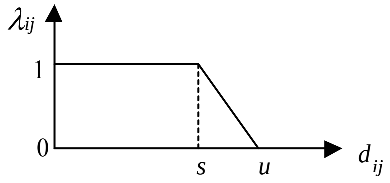

Fig. 1. The decay function that presents the flexible distance standard

Since the CILCP attempts to maximize the covered populations within the distance standard, we can write the objective function of model as:

$$Z = \sum_{j=1}^{n}\sum_{i=1}^{n} a_i \lambda_{ij} X_{ij} \tag{2}$$

Since the model attempts to maximize Z, only those X_{ij}'s assume value 1 whose λ_{ij} 's are not zero, i.e., only when the distance from node *i* to server *j* is approximately less than or equal to the distance standard.

There are four types of constraints incorporated in our model. The first constraint assures that variable X_{ij} can assume value 1 only when variable Y_j has already taken value 1, in other words, it makes certain that demand node *i* can be covered by server *j* only when a server is located at node *j*. This constraint appears in the model as :

$$X_{ij} \le Y_j \quad , \quad \forall\, i, j \tag{3}$$

The second constraint states that each node can only be covered by one server, i.e. ,

$$\sum_{j=1}^{n} X_{ij} \le 1 \quad , \quad \forall\, i \tag{4}$$

The third constraint specifies the total number of servers, i.e. ,

$$\sum_{j=1}^{n} Y_j \;=\; p \tag{5}$$

The last constraint states that the satisfaction degree of

$\hat{D}_j \le \hat{C}_j$, is at least   , and presented as

$$P(\hat{D}_j \le \hat{C}_j Y_j) \ge \alpha \qquad ,\forall j \tag{6}$$

Let $\hat{d}_i$ denote the demand rate for service at node *i*. If node *i* is covered by server *j*, X_{ij} will be equal to 1. Under these circumstances, the assigned demand rate to service center *j* is calculated as :

$$\hat{D}_j = \sum_{i=1}^{n} \hat{d}_i X_{ij} \tag{7}$$

This is so, because the assigned demand rate for service at each server is equal to the sum of the demand rates for all the nodes covered by that particular server, where $\hat{D}_j$ is an interval number as:

$$\hat{D}_j = (\sum_{i=1}^{n} d_i^{l} X_{ij} \ ,\ \sum_{i=1}^{n} d_i^{u} X_{ij} \ ) \tag{8}$$

So by applying the definition (2), to transform the constraint (6)   , we will have:

$$P(\hat{D}_j \le \hat{C}_j Y_j) = \frac{\left\{ \left| \max\left(0, C_j^U Y_j - \sum_i d_i^L X_{ij}\right) \right| - \left| \max\left(0, C_j^L Y_j - \sum_i d_i^U X_{ij}\right) \right| \right\}}{\left\{ \left| \left(\sum_i d_j^U X_{ij} - \sum_i d_i^L X_{ij} \right) \right| + \left| \left(C_j^U Y_j - C_j^L Y_j \right) \right| \right\}} \ge \alpha \tag{9}$$

By defining λ_j^1 and λ_j^2 as:

$$\lambda_j^1 = \max(0, C_j^U Y_j - \sum_i d_i^L X_{ij}), \tag{10}$$

$$\lambda_j^2 = \max(0, C_j^L Y_j - \sum_i d_i^U X_{ij}), \tag{11}$$

The relation (9) is converted as follows:

$$\lambda_j^1 - \lambda_j^2 \ge \alpha(\sum_i d_i^U X_{ij} - \sum_i d_i^L X_{ij}) + \alpha(C_j^U Y_j - C_j^L Y_j)\ \forall j \tag{12}$$

$$\lambda_j^1 \ge C_j^U Y_j - \sum_i d_i^L X_{ij}, \qquad \forall j \tag{13}$$

$$\lambda_j^2 \ge C_j^L Y_j - \sum_i d_i^U X_{ij}, \qquad \forall j \tag{14}$$

$$\lambda_j^1 \ge 0, \qquad \forall j \tag{15}$$

$$\lambda_j^2 \ge 0, \qquad \forall j \tag{16}$$

Therefore, relations (12) to (16) substitute the constraint (6). An analysis about relation (10), is presented here. Since, if D_j^L be equal or more than $C_j^U Y_j$ then the satisfaction degree of $\hat{D}_j \le \hat{C}_j Y_j$ is zero, however we want to have the satisfaction degree at least α, so the relation $C_j^U Y_j - \sum_i d_i^L X_{ij}$ will always be positive. Therefore the $\max(0, C_j^U Y_j - \sum_i d_i^L X_{ij})$ will be equal to $C_j^U Y_j - \sum_i d_i^L X_{ij}$. So we can use the following constraint instead of constraint (13),

$$\lambda_j^1 = C_j^U Y_j - \sum_i d_i^L X_{ij}, \qquad \forall j \tag{17}$$

Now, we can write the final 0-1 integer-programming model.

3.4. The CILCP mathematical model

On the basis of the discussion in previous sections, the CILCP is proposed as :

$$\text{Max } Z = \sum_{i,j} a_i \lambda_{ij} X_{ij}$$

s.t.

$$X_{ij} \le Y_j \qquad , \forall i, j$$

$$\sum_{j=1}^{n} X_{ij} \le 1 \qquad , \forall i$$

$$\sum_{j=1}^{n} Y_j = p$$

$$\lambda_j^1 - \lambda_j^2 \ge \alpha\left(\sum_i d_i^U X_{ij} - \sum_i d_i^L X_{ij}\right) + \alpha\left(C_j^U Y_j - C_j^L Y_j\right), \qquad \forall j,$$

$$\lambda_j^1 = C_j^U Y_j - \sum_i d_i^L X_{ij}, \qquad \forall j,$$

$$\lambda_j^2 \ge C_j^L Y_j - \sum_i d_i^U X_{ij}, \qquad \forall j,$$

$$X_{ij} = 0,1 \quad , \quad Y_j = 0,1 \quad , \quad \lambda_j^1 \ge 0 \quad , \quad \lambda_j^2 \ge 0, \qquad \forall i, j$$

4. Computational procedure

Since the p-median problem is NP-Hard ([8]), and the proposed 0-1 integer-programming model in this paper can be reduced to the p-median problem in polynomial time, so it is NP-Hard. Therefore, it cannot be solved in general to optimality and it is appropriate to develop a heuristic method to solve the problem within a reasonable time. We present a genetic algorithm to solve the CILCP in this section.

4.1. Genetic algorithms and the relevant literature

Hosage and Goodchild [7], first applied genetic algorithm to location-allocation problems. Dibble and Densham [5], proposed a genetic algorithm for the multi-criteria facility location problem. Moreno-Perez et al. [12], introduced another genetic algorithm for the p-median problem. Kratica et al. [9], developed a genetic algorithm for the simple plant location problem. Bozkaya et al. [1], developed an efficient genetic algorithm for the p-median problem that the preliminary results were presented at INFORMS conference in 1997 and the final paper was published in 2002. They proved that their algorithm was more effective than the other genetic algorithms for p-median problem. Shavandi and Mahlooji [15], introduced a genetic algorithm for their developed fuzzy queuing location problem. Since the structure of the fuzzy queuing location model presented by Shavandi and Mahlooji [15], is almost similar to the CILCP, it is possible to use some results from the genetic algorithm to advantage for our problem.

4.2. The proposed genetic algorithm

In this section we first define the notation and provide a general outline of the proposed GA and then describe each element of the algorithm.

4.2.1. Notation and outline

The notation that will be used in the proposed GA are as follows :

PZ : size of the population.
G : number of generations (stopping rule).
M : mutation rate.
Z_j : fitness value or the objective function value for solution *j*.
O : number of overlapping solutions from one generation to the next.

The general outline of the proposed GA is as follows:

Step 0 : Initialize *PZ*, *G*, and *M*.
Step 1 : Randomly generate the initial population of size *PZ*.
Step 2 : Repeat *G* times :
  Step 2.1 : Select the parent population with the size of $PZ/2$, according to the roulette wheel method.
  Step 2.2 : repeat (*PZ* -*O*) times :
    a. Select two parents *P1* and *P2* randomly.
    b. Apply the crossover operator to produce two offspring from *P1* and *P2*.
    c. With probability *M*, apply the mutation operator to the offspring.
  Step 2.3 : Replace the offspring with the current solutions while keeping the best solution found so far.
Step 3 : Print the best solution found so far.

We used *2n* (n is the number of nodes at network) for the parameter *G* of proposed GA.

4.2.2. Encoding

In the proposed GA, each chromosome is represented by an *n*-dimensional vector like $A = [\, a(1), a(2), \ldots, a(n) \,]$ whose *i*th entry stands for the *i*th node in the network. The value of the *i*th entry is the number of server which covers node *i*. If $a(j) = j$, it means that a server is located at node *j*. When $a(j) \neq j$, it means that node *j* is covered by server $a(j)$.

4.2.3. Population size

Based on the previous work and our limited experiments we found the best size of population as:

$$PZ = \ln\binom{n}{p}\ln\big(p(n-p)\big)$$

4.2.4. Parent selection

Once the best chromosome (solution) in the population is determined, the algorithm selects the other parent randomly according to the roulette wheel method. This method, first sorts the chromosomes of the population based on the fitness values in ascending order, and then the cumulative function of the sorted solutions according to fitness values is calculated. To choose the other parents a random number between zero and one is generated and multiplied by the total sum of the fitness values, the resulting value belongs to the interval formed by two consecutive values of the cumulative function. The number associated with the upper value is chosen as a parent.

4.2.5. Crossover

The production of offspring is implemented by first selecting a pair of parents, then applying the crossover operator to produce two offspring. Fixing the location of servers, the crossover tries to change the covered nodes. Thus, the difference between the generated offspring and parents has to do just with the covered nodes (location of servers are identical). This operator can be repeatedly applied on any pair of parents as many times as a random number between 1 and *p*. Each time the crossover operator is applied, one server from each pair of parent is chosen and their covered nodes are exchanged. To prevent generation of infeasible offspring, each exchange is preceded by a check on the service quality constraint to make sure that exchanging the covered nodes is feasible.

4.2.6. Mutation

Once the crossover operator is applied and offspring are generated, the mutation operator is applied on offspring with probability *M*. The mutation operator attempts to change the location of servers. This operator can be repeatedly applied on any of the offspring as many times as a random number between 1 and *p*. Each time the mutation operator is applied, one server and one of the nodes covered by that particular server are randomly chosen, the chosen server is relocated to the location of the chosen node, and finally, all other nodes covered by the server before relocation will remain under its coverage at the new location. Again, to prevent generation of infeasible offspring, the service quality constraint is checked on. Based on previous works and our limited experiments, the suitable value for *M*, was found to be between 0.2 and 0.3.

4.2.7. Replacement

Once the population of offspring are generated (and possibly mutated), they are used to replace the current population. The overlapping level *O* determines the number of common solutions in the population of two successive generations. For example, *O* = 0, implies a complete replacement of *PZ* offspring with the current population, whereas *O* = *PZ* – 1 means that only one solution is replaced in each generation. With respect to the experiments in Shavandi and Mahlooji ([11]), the best performance is achieved by using *O* = 1, where the single overlapping solution is the fittest or best solution. With our experiments, it was decided that *O* = 1, is suitable for the proposed GA.

4.3.Numerical results

To solve the problems, Delphi version 6.0 computer program was used to program the GA and IBM OSL v3, was used to obtain the optimum solution of the same problems. We wrote a random generator program, which generates problems randomly. In this section, we present some results obtained by solving such problems. We display the parameter values for a 15-node problem in Table (1), and solutions for two cases as well as comparison between optimal solution obtained by OSL and GA solution are presented in Tables (2), and (3). Table (2), represents the solutions for α=0.9 and in the optimal solution, the servers are located at nodes 8, 9, and 14, and nodes 7, and 12 were not covered by servers. However, in the case of α=0.8, as was shown in the table (3), the location of servers were changed and were located at nodes 8, 10, and 14, and the uncovered node is only 12. To summarize the performance of GA algorithm, based on solving problems whose partial results are presented here, Table (4), indicates that the overall average percentage of discrepancies between the optimal solutions from OSL and the GA solutions is less than 4.5%.

5.Conclusion

This work presents a model for location covering considering interval values for demands and service capacity. The model was transformed into a mixed integer-programming model. Since the proposed model computationally belongs to the class of NP-Hard problems, a genetic algorithm was developed to solve the model.

References

[1] B. Bozkaya, J. Zhang, E. Erkut, An efficient genetic algorithm for the *p*-median problem, in : Z. Drezner, H. W. Hamacher (Eds.), Facility Location : Applications and Theory. Springer, Heidelberg, 179-205, 2002.

[2] R. Church, C. Re Velle, The maximal covering location problem. Papers of the Regional Science Association, 32, 101-118, 1974.

[3] L. Cooper, Location-Allocation problems, Operations Research, 11, 331-344, 1963.

[4] J. Current, M. Daskin, D. Schilling, Discrete Network Location Models, in : Z. Drezner, H. W. Hamacher (Eds.), Facility Location : Applications and Theory. Springer, Heidelberg, 80-118, 2002.

[5] C. Dibble, P. J. Densham, Generating interesting alternatives in GIS and SDSS using genetic algorithm, GIS/LIS, 1993.

[6] A. T. Ernst, M. Krishnamoorthy, Solution algorithms for the capacitated single allocation hub location problem. Annals of Operations Research, 86, 141-159, 1999.

[7] C. M. Hosage, M. F. Goodchild, Discrete space location-allocation solutions from genetic algorithm. Annals of Operations Research, 6, 35-46, 1986.

[7] O. Kariv, S. L. Hakimi, An algorithmic approach to network location problems, Part 2 : The *p*-medians. SIAM Journal on Applied Mathematics 37, 539-560, 1979.

[8] J. Kratica, D. Tosic, V. Filipovic, I. Ljubic, Solving the simple plant location problem by genetic algorithm. Rairo Operations Research 35, 127-142, 2001.

[9] X. Liu, Measuring the satisfaction of constraints in fuzzy linear programming. Fuzzy Sets and Systems, 122, 263-275, 2001.

[10] R. Longendran, M. P. Terrell, Uncapacitated plant location-allocation problems with price sensitive stochastic demands. Computers and Operations Research, 15(2), 189-198, 1988.

[11] J. A. Moreno-Perez, J. M. Moreno-Vega, N. Mladenovic, Tabu search and simulated annealing in *p*-median problem. Operational Research Society Conference, Canada, Montreal, 1994.

[12] M. Ohlemuller, Tabu search for large location-allocation problems. Journal of Operational Research Society, 48(7), 745-750, 1997.

[13] H. Shavandi, H. Mahlooji, Fuzzy queuing location-allocation models for congested systems. International Journal of Industrial Engineering, 11(4), 364-376, 2004.

[14] H. Shavandi, H. Mahlooji, A fuzzy queuing maximal covering location-allocation model with a genetic algorithm for congested systems. Applied Mathematics and Computation, 181, 440- 456, 2006.

[15] C. Toregas, R. Swain, C. Re Velle, L. Bergman, The location of emergency service facilities. Operations Research, 19, 1363-1373, 1971.

[16] J. Zhou, Uncapacitated facility layout problem with stochastic demands. In proceeding of the sixth national conference of operations research society of china, china, 904-911, 2000.

[17] J. Zhou, B. Liu, New stochastic models for capacitated location-allocation problem. Computers and Industrial Engineering, 45(1), 111-125, 2003.

[18] J. Zhou, B. Liu, Modelling capacitated location-allocation with fuzzy demands. Computers and Industrial Engineering, 53, 454-468, 2007.

Appendix – The numerical results

Table 1
Constant data input for the problems whose solutions are presented in Tables 2 and 3

Number of nodes (n) = 15							$\hat{C}$ = (37 , 43)								
n	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
ai	937	503	524	654	585	597	580	679	782	914	582	628	854	695	912
d^L	2	3	7	6	5	2	4	1	1	7	9	8	4	3	5
d^U	5	8	11	12	9	6	8	5	5	13	13	12	9	8	10
								λ_{ij}							
1	1	1	.2	.5	1	0	.6	1	0	.9	.7	.14	.51	.3	0
2		1	.2	0	.5	.14	.32	1	.25	.64	.9	.15	.62	0	.7
3			1	0	.3	.5	.18	.51	.61	.71	.2	.02	0	1	.9
4				1	.21	.51	.54	.61	.12	.15	0	1	.29	.84	.17
5					1	1	.9	.8	.14	.21	.51	.3	0	1	.24
6						1	.2	1	0	.3	.6	.9	.4	.7	.6
7							1	.2	0	.9	.4	.61	.72	.1	.2
8								1	.6	0	.9	.8	.4	.7	.61
9									1	1	.3	.8	.47	.16	.92
10										1	.2	.7	.8	.14	.61
11											1	.9	.2	.4	.31
12												1	.2	.1	.09
13													1	.8	.12
14														1	0
15															1

Table 2
A comparison of the results obtained from the GA against OSL

Number of nodes (n) = 15		α = 0.9	p = 3			
OSL		Run-time ≅ 0	GA		Run-time ≅ 0	
obj	Loc	Covered nodes	obj	Loc	Covered nodes	%
8607	8	1, 2, 6, 8, 11	8311	8	1, 2, 5, 8, 11	3.4
	9	3, 9,10,15		9	3, 9, 10, 15	
	14	4, 5, 13, 14		14	4, 6, 13, 14	

$$\% = \frac{obj(OSL) - obj(GA)}{obj(OSL)} \times 100$$

Table 3
A comparison of the results obtained from the GA against OSL

Number of nodes (n) = 15		α = 0.8	p = 3			
OSL		Run-time ≅ 0	GA		Run-time ≅ 0	
obj	Loc	Covered nodes	obj	Loc	Covered nodes	%
8956	8	2, 6, 8, 11, 15	8526	8	2, 6, 8, 11, 13	4.7
	10	1, 7, 9,10,13		10	1, 3, 7, 9, 10, 15	
	14	3, 4, 5, 14		14	4, 5, 12, 14	

$$\% = \frac{obj(OSL) - obj(GA)}{obj(OSL)} \times 100$$

Table 4
A comparison between the optimal solution (OSL) and the solution from the GA

Number of nodes	Number of solved problems	Mean run-time (seconds)	Average discrepancy (%)
20	32	≅ 0	≅ 4.1
30	30	≅ 0	≅ 3.2
40	25	≅ 1	≅ 4.75
50	25	≅ 2	≅ 5.1
60	20	≅ 2	≅ 4.5
70	20	≅ 2	≅ 3.95
77	12	≅ 2	≅ 5.1
Overall average discrepancy between the optimal solutions from OSL and the GA solutions (%)			≅ 4.38