APPLYING GENETIC ALGORITHM ON SELECTING EMERGENT MEDICAL STATION BEFORE DISASTERS

Ming-Der May

Department of Industrial Engineering and Management, LungHwa Institute of Technology
Taoyuan, Taiwan, R.O.C.

Abstract: This study applying a mathematical approach to select facilities for the emergent rescue plan before the natural disasters, such as earthquakes, typhoons and floods. The hazardous area is divided into sub-area by the capacity of medical treatments and the distance between the habitants, and the emergent medical station is located at the position of the seed of this sub-area. When if any disaster happens, the wounded can be sent directly to the nearest medical station and will accept proper treatments. Such a problem can be formulated as the so-called Capacitated Clustering Problem (CCP). The CCP is to partition a group of n items (ex. the habitants) into k clusters (ex. Sub-areas) and the entities within a cluster should be as homogeneous as possible and under volume constraints. This study applies genetic algorithm (GA) to solve the CCP and the solution quality is compared with integer optimization software, LINDO. Further research of the emergency evacuation plan is another similar problem and worthy of more detailed study..

Key Words: capacitated clustering problem, genetic algorithms, binary coding, adaptive penalty function, medical station location

1. INTRODUCTION

This study applying a mathematical approach to select facilities for the emergent rescue plan before the natural disasters, such as earthquakes, typhoons and floods. The hazardous area is divided into sub-area by the capacity of medical treatments and the distance between the habitants, and the emergent medical station is located at the position of the seed of this sub-area. When if any disaster happens, the wounded can be sent directly to the nearest medical station and will accept proper treatments. Such a problem can be formulated as the so-called Capacitated Clustering Problem (CCP).

This capacitated clustering problem is defined as dividing *n* nodes or objects into *k* groups to minimize the assignment cost and to satisfy the capacity constraints. The CCP is a special case of the facility location problem and is closely related to the generalized assignment problem [4] and p-median problem. Garey and Johnson [5] indicated not only that the CCP is NP-complete, but also that exact optimization integer programming algorithms are ineffective for larger problems.

The generalized assignment algorithm developed by Fisher and Jaikumar [4] for vehicle routing ushered in the use of the CCP. They used very simple heuristics to solve the generalized assignment problem which is a subset within the larger vehicle routing problem. The generalized assignment problem, however, requires the pre-specification of seed nodes that serve as clustering points. In large problems, the choice of the seed nodes can be extremely difficult. Algorithms for CCP, on the other hand, explicitly choose the seeds as part of the algorithm. Mulvey and Beck [12] proposed heuristics and used randomly generated seeds as initial solutions for solving the CCP. Koskosidis and Powell [10] extended the work of Mulvey and Beck by proposing an iterative algorithm that was shown more effective for developing initial seeds.

Meta-Heuristics such as the genetic algorithms, simulated annealing and tabu search methods have become increasingly popular as a means to solve difficult combinatorial optimization problems with the type similar to those in operations research. However, there do not exist many research reports using GA to solve the CCP. Thangiah and Gubbi [15] proposed a genetic sectoring method to find good clusters of nodes for the vehicle routing problem with a "cluster-first route-second" problem solving strategy. Once the clusters are identified by the genetic search, classical insertion and post-optimization procedures are applied to produce the chosen routes. Although this approach is able to solve the CCP, it is designed on the Euclidean plane and may not suit the problems on the road networks in real world. On the other hand, Osman and Christofides [13] employed the hybrids of simulated annealing and tabu search methods to solve the CCP.

2. PROBLEM FORMULATION

The mathematical formulation shown below is the Capacitated Clustering Problem defined by Koskosidis [11].

$$\text{minimize} F(y,g) = \sum_{i \in I} \sum_{j \in J} c_{ij} y_{ij} \quad (1)$$

$$\text{subject to} \quad \sum_{i \in I} q_i y_{ij} \le V \qquad \forall\, j \in J \quad (2)$$

$$\sum_{j \in J} y_{ij} = 1 \qquad \forall\, i \in I \quad (3)$$

$$y_{ij} \le g_j \qquad \forall\, i \in I, j \in J \quad (4)$$

$$\sum_{j \in J} g_j \le K \quad (5)$$

$$y_{ij} = 0 \text{ or } 1 \qquad \forall\, i \in I, j \in J \quad (6)$$

$$g_j = 0 \text{ or } 1 \qquad \forall\, j \in J \quad (7)$$

Where

I : the set of habitant node, $i=1,2,...N$.
J : the set of candidate seed location j.
K : the number of available medical stations.
c_{ij} : the cost of traveling directly from habitant node i to j.
q_i : the number of habitants in node i.
V : the capacity of the medical station that serves the cluster

Binary Variables:

y_{ij} = 1 if node i belongs in the cluster of seed j
0 otherwise
g_j = 1 if node j is a seed node for medical station
0 otherwise

The variables y_{ij} are the assignment variables; and the variables g_j indicate whether a candidate seed j is selected. The cost coefficients c_{ij} measure the impedance between a node i and a seed j (e.g. c_{ij} could be the distance between i and j or a composite measurement of distance, travel time, etc.).The objective function of the CCP model is to minimize the total assignment cost of nodes to the cluster seeds.

Constraint (2) in the above formulation restricts the number of nodes assigned to a cluster (i.e., a medical station), such that the capacity of the medical station is not exceeded. Constraint (3) ensures that each node is assigned to the one, and the only one, seed j; and constraint (4) is to prevent assigning a node i to the candidate seed j which has not been selected as a seed (in which case y_{ij} =0). Finally, constraint (5) ensures that as many as K seeds are chosen.

3. APPLICATION OF GENETIC ALGORITHMS

Holland and his associates at the University of Michigan developed genetic algorithms in the 1960s and 1970s. The first full systematic treatment was contained in his book *Adaptation in Natural and Artificial System* [7] published in 1975. The guiding premise of GA is that the complex problem can be solved by simulating evolution via a computer algorithm. In the original GA conception, the process is viewed as a black box that provides evaluations of chromosomes (solutions). These evaluations are then used to bias the selection of chromosomes in a way that superior chromosomes (i.e. those with higher evaluations) will reproduce more often than that of the inferior.

The application of GA to solve CCP in the study has the following advantages. First, the CCP is a 0-1 integer problem and is therefore natural to represent the problem by the binary coding of GA. Likewise, this discrete feature needs the least effort when decoding the chromosomes to the original problem. Second, the CCP is a NP-Complete problem and the existing solution methods are complicated, therefore the application of GA will be worthy if it can provide better solutions with less or even the least problem-related information and knowledge.

3.1 Coding and Decoding for the CCP

There are several ways to represent the problem with proper coding for genetic search, such as binary coding, sequential coding and k-ary coding. In this study, the CCP is coded as binary strings for the following two reasons. First, the CCP is a 0-1 integer problem, thus, the application of a GA would be straightforward to encode a solution as a binary string. Second, during the genetic search, a binary string will naturally compatible with the standard GA operators, i.e. reproduction, crossover and mutation and will not generate infeasible offspring. This feature will save the computing resource and make it easy to implement.

In the formulation of the CCP, nodes are classified into clusters according the set of seeds. Therefore, the encoded chromosome has to find a seed node in each cluster. A natural representation might be a chromosome with two parts: one for nodes and the other for seeds. Thus, the developer can define a binary chromosome at the length of $2NK$, where N is the number of nodes and K is the number of available medical stations. In the part of nodes,

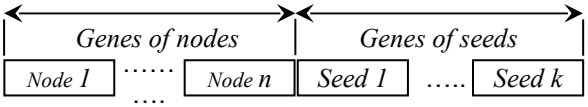
$$y_{ik} = \begin{cases} 1 & \text{if node } i \text{ is in the cluster } k \\ 0 & \text{otherwise.} \end{cases}$$

and in the part of seeds,

Although this approach is straightforward, it would expend a rather lengthy chromosome and many solutions would inevitably be infeasible.

Another approach used in this study is to define a chromosome at the length of $aN + bK$, where a and b are the number of binary digits to represent the nodes and seeds. Like the example of chromosome in Figure 1, the genes for nodes are divided into sub-groups for every *"a"* digits (in this example, a = 4). In the *ith* division, the binary digits are changed to decimal according to the value of j. The j is the

cluster that the *ith* node belongs to. There will be at most 16 alternative clusters for each node when *"a"* is equal to four. In the part of genes for seeds, every division of *b* (five in this example) digits represents the position of the seed node in the cluster. In the example of five digits, there are at most 32 positions in a cluster, therefore, it is the maximum number of nodes in a cluster. Since *a* and *b* are substantially less than *N* or *K*, we can see that *aN +bK* will be less than *2NK*. This will save memory space and computing time.



0110|0011|…|…..|0110|01101|10010|…|01110|

Figure 1. Chromosome for CCP

The initial population is generated randomly by GA. Every chromosome of the population will be decoded into a solution by the way depicted in Figure 1. The next step is to evaluate the fitness of every solution in order to generate a new population. The following sections will describe the operating strategies to generate a new population first and then to illustrate the methods to find fitness value for each chromosome.

3.2 Genetic Operating Strategy

After generating the initial population, the GAs apply three operators to find new populations. In this study, the binary coding will be inherently compatible with the traditional SGA operators, therefore, no additional effort is needed to design a new operator for this problem. However, many strategies are applied in this study to enhance the performance of GA as an optimizer.

3.2.1 Reproductive Strategy

When generating a new population, this study adopts De Jong's [2] concepts of *elitism* and *population overlaps*. An elitist strategy ensures the survival of a best individual so far by preserving it

$$g_{ik} = \begin{cases} 1 & \text{if node } i \text{ is the seed of cluster } k \\ 0 & \text{otherwise.} \end{cases}$$

and replacing only the remaining members of the population with new strings. Overlapping population take a stage further by replacing only a fraction *r* (replacement rate) of the population at each generation. Selection schemes also have to be considered here. The original *roulette-wheel* method is chosen here because it performs better than the other methods after empirical tests.

The scaling or normalizing of fitness for the members of population is necessary since the objective has to be changed from being minimized to being maximized. Therefore, a *reverse ranking* method, i.e. a lower fitness indicates a higher rank, is used.

3.2.2 Recombination Strategy

The original one-point crossover operator may not be the best approach for every problem. In this study, GA uses multipoint operator, where *n* crossover points are chosen randomly.

The question of how often crossover should be applied has been investigated experimentally in the literature. De Jong's [2] experiment suggested that a crossover rate of 0.6 was appropriate. Grefenstette [6] proposed 0.95, while Schaffer et al. [14] suggested it should vary with population size and string length. After testing several values, the crossover rate of 1.0 is chosen in this study.

The mutation rate that decide how often mutation should be applied is also obtained by experimentation. A value of 0.02 is selected and the type of mutation rate is *per-gene*, that is, every gene will have the possibility of 0.02 to mutate.

3.3 Fitness Evaluation

The fitness value is composed of the value of the objective function and the measurement of unfeasibility of the constraints.

3.3.1 Objective function

The objective function in the equation (1) of the CCP model minimizes the total assignment cost of nodes to the seed of the assigned cluster. The assignment cost is the distance of the shortest path from a node to its tagged seed. In this study, the matrices of the shortest path for every node to the others are stored that the GA can access in the process. The value of the objective function is the basic fitness for every individual string.

3.3.2 Constraints

In the formulation of the CCP, the constraints, from (2) to (5), must be included to obtain feasible solutions. Because of the genetic coding method of the CCP as illustrated in Figure 3, most of these constraints are conformed inherently. For instance, the constraint (3), i.e. each node has to be assigned to the *only one* seed, is satisfied automatically by the genes of nodes. The constraint (4) that ensures the assignment of a node to a selected seed is not necessary here, because the genes of seeds always assign a seed for each cluster. Similarly, the limitation of *k* seeds in constraint (5) is satisfied since the genes of seeds are decoded as many as *k* seeds.

Only the constraint (2) that the accumulated volume in each cluster should not exceed the capacity of the medical station has to be handled separately.

This constraint cannot be satisfied by the coding string inherently and must be done with other methods. In this study, we use the penalty method because it needs less modification and is easy to implement.

3.3.3 Adaptive Penalty Function Method (APF)

In fact, much experience in literature shows that a naïve attempt to use penalty function often fail. If the penalty is too small, many infeasible solutions are included to propagate; if the penalty is too large, the search is confined to the interior of the search space and far from the boundary of the feasible region. Note that it is common to find the global optimum on or near a constraint boundary.

To address this kind of failure, Fairley [3] and Coit, Smith and Tate [1] suggested using *adaptive penalty functions* in such situations. This study adopts the proposal by Coit et al. [1]. The rationale of this method is that moderately infeasible solutions will lie near the feasibility boundary and close to the optimal solution. The penalty function allows some unfeasibility, because the infeasible solutions have the potential to recombine or mutate to produce a near optimal solution. The fitness after being penalized by the capacity constraint (2) is derived from the following equation.

$$F_p(x) = F(x) + (F_{feas} - F_{all})\left(\frac{\Delta Capacity}{NFT_c}\right)^{k_c} \quad (8)$$

$F_p(x)$: *penalized objective function (fitness).*
$F(x)$: *unpenalized objective function value for solution* x.
F_{feas} : *the best feasible solution value yet found.*
F_{all} : *the unpenalized value of the best solution yet found.*
ΔCapacity: *the excessive amount for capacity constraint.*
NFT_c : *Near-feasible threshold for capacity constraint.*
k_c : *user specified severity parameter for capacity constraint.*

The penalty function will encourage GA to explore within the feasible region and the NFT_c neighborhood of the feasible region, and discourage search beyond that threshold. The NFT_c could be a static constant or a dynamic searching parameter changed as a function of generation number. This study applies the following equation to decrease the value of NFT_c as the algorithm progresses.

$$NFT_c = NFT_0 / Generations \quad (9)$$

The NFT_0 is some upper bound for NFT_c. Coit et al. [1] point out that the definition of NFT_c is not only problem specific, but also constraint specific. This study will show how the NFT_c, or actually the NFT_0, affects the performance of the algorithm in the next section of computational analysis.

3.5 Stop rules

Generally, the designated number of generation is often the signal to stop the algorithm. The other approaches might use a measure of convergence that requires the difference of the best fitness between two generations is less than a specified constant. Another stopping rule is to apply a target value to stop the algorithm. In this study, the number of generation is the major rule to stop the algorithm, and the target value is applied if other solution algorithm gives some lower bound.

3.6 Using SUGAL

SUGAL [soo-gall] is the **SU**nderland **G**enetic **Al**gorithm package, developed by Dr. Hunter at the University of Sunderland, England. It is a package, written in 'C' language, designed for experimentation with GA and related techniques [8, 9]. The package is employed to perform a GA research and provides a large number of options on configurations and extendibility. This feature enables the researchers to evaluate different operation strategies easily. Another major feature of the SUGAL is that the user needs only to provide a single C procedure, which can evaluate the fitness of a chromosome, and a simple main procedure, which hands over control to the SUGAL code. Although the user still has to write some computer codes, he will not have to intervene in the coding of other genetic operations and can focus on finding better solutions.

In the following computational study, the SUGAL is applied to solve the CCP according to the string coding, operational strategies and fitness evaluation methods that have been described in the above sections.

4. COMPUTATIONAL STUDY

In this section, the major objective is to evaluate the quality of solutions and computing efficiency of the proposed GA methods, and to compare the adaptive penalty function with different parameters each other.

The testing network, which has 103 nodes and 167 arcs, is the output from the map of Chung-Li City that is digitized by GIS software, Arc/Info. The set of habitant nodes are chosen randomly from the existing road network. Every node has a number of the network and the amount of habitant around it. Every arc of the network has two cost categories. They are the functional values of the corresponding arc length and traffic volume. The two kinds of cost categories represent the lengths of the arc in both directions and may be different to each other. The matrix of shortest path for every candidate node is

derived from this real road map and is saved as files that the GA can access it during the process. Table 1 lists the characteristics of the seven problems ..

Table 1. Characteristics of the first problem set

Prob.	No. of nodes	Medical Capacity	Total habitants
1	28	100	420
2	41	100	810
3	41	100	810
4	41	100	1220
5	41	230	1076
6	55	230	1279
7	46	100	1088

4.1 GA

This study applies SUGAL to perform the proposed GA methods. Since most of the problem-related processes, such as the chromosome decoding and fitness evaluation, are still controlled by the user, using SUGAL is not quite different from the traditional way that the user programs all the functions of the GA. In fact, a user can save time for program coding and try a good many of options to find the best way to apply the GA to solve a specific problem.

Some of the parameters are problem-specific such as a random seed and the chromosome length. The parameters that are common for every test problem are listed in Table2.

The improving directions of the GA for solving the CCP are to reduce the cost of assignment without violating the capacity constraint. In order to enhance the convergent speed and performance of the algorithm, the APF method is employed in two alternatives. The first, denoted as APF_1, has added the severity parameter $k_c = 2$ and assign the fitness a large constant penalty P if this chromosome is infeasible to capacity constraint. This approach encourages the GA to search more solutions that are feasible and decreases the possibility of the inclusion of infeasible ones.

Table 2. Common values of control parameters

Parameters	**Value**	**Parameters**	**Value**
Annealing decay	1.0	Mutation rate	0.02
Bias	2	Normalization	Reverse_
Crossover type	Npoint		rank_
Crossover points	15		linear
Crossover rate	1.0	Population	80
Elitism	On	Replacement condition	Anneale d
Mutation	Invert	Replacement rate	0.7
Mutation rate type	Per_gene	Replacement Selection	Ranked Roulette

Another approach (APF_2) has the severity parameter $k_c = 3$. This is simply an execution of the equation (8). Therefore, some infeasible solutions are encouraged to reproduce and the best fitness chromosome in a generation may be an infeasible solution. The solutions found by these two approaches are described in Table 3.

Table 3. Results of GA for the first problem set

Prob	Groups	APF_1		APF_2	
1	5	967.85[□]	806[$] (2110,$10^5$)[%]	959.30	2358 (2110,10^4)
2	11	1125.47	4324 (2110,10^5)	1106.6	5410 (2110,100)
	10	1171.70	6747 (1, 10^4)	1180	19484 (2110, 1000)
	9	1550.2	25506 (1110, 10^5)	1404.32	27300 (110,2000)
3	10	1492.7	8610 (2110, 10^5)	1230.60	27824 (1110,1000)
	9	1531.0	6151 (5, 10^6)	1572.13	1488 (2110, 1000)
4	16	865.25	22643 (110, 10^6)	836.899	1950 (110, 10^4)
5	5	1816.16	2515 (5,10^5)	1783.79	8660 (5, 10^4)
6	7	1818.63	3376 (2110,10^4)	1849.13	6704 (110, 1000)
	6	2253.28	8667 (5, 10^4)	2238.81	1224 (5, 1000)
7	13	1492.0	4625 (1234, 10^6)	1386.51	8986 (2110, 5000)

[□]Objective Value [%](Random seed, NFT_0)
[$]The number of generation that this solution is generated

The results presented in Table 3 include the values of the objective, random seeds, NFT_0 and the number of generation that the best solution appears. The computing time in a Pentium 133 PC is about 10 minutes for every 10,000 generations. Most of the acceptable solutions are obtained within 10,000 generations. The results indicate that the APF_2 finds better solutions in most of the cases. However, neither of the two methods is always superior to the other in the convergence speed. The mechanism of improving convergence speed in APF_1 possibly leads to limit on a narrow range of search space that the algorithm may fail to find better solutions. The profiles of convergence for APF_1 and APF_2 in problem 1 are depicted as in Figure 2.

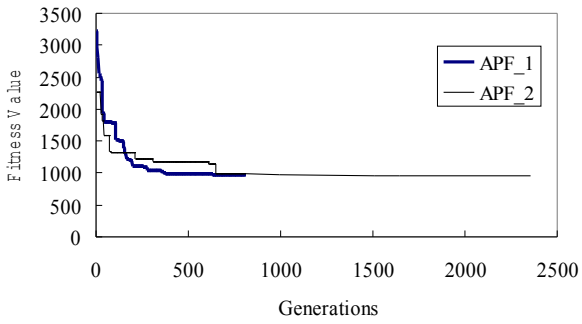


Figure 2. Comparison of APF_1 and APF_2 for problem 1.

The problem-specific parameter, NFT_0, affects both of the convergence speed and solution quality. When the NFT_0 is small, the algorithm converges quickly and is prone to become premature convergence. On the other hand, when the NFT_0 is large, the algorithm will search widely in the exploration phase to find better solution and may converge slowly. The performance of different NFT_0 for problem 1 is compared in Figure 3.

The effects of different random seeds are shown in Figure 4. Ten continuous seeds, from 2106 to 2115, are evaluated with problem 1 by the penalty function of APF_2 and the average fitness is about 1115.54. We can see that the performance of different seeds is steady in general except for two cases and the solutions converge within 10,000 generations

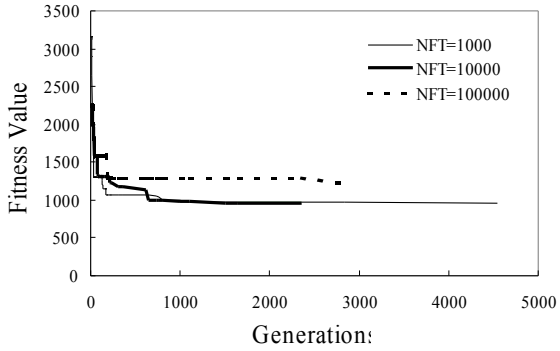


Figure 3. Comparing different NFT_0 for problem 1.

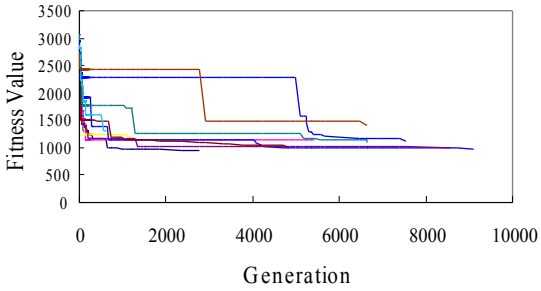


Figure 4. Comparison of ten continuous seeds for problem 1 with NFT_0 = 10,000.

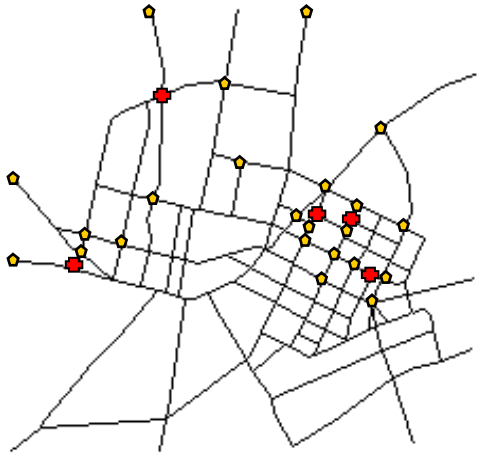


Figure 5. The location of medical stations and habitant nodes for test problem 1

The best result of test problem 1 is drawn as in Figure 5. There are 28 habitant nodes and five of them are chosen to be the medical stations. Each habitant node has been assigned to a medical station that will have enough capacity and has the minimum total distance.

5.3 LINDO

A hyper version of LINDO, which can handle the problems within 4000 integral variables, 2,000 rows, 4,000 columns and 64,000 non-zero items, is used to obtain the 0-1 integral solution of the CCP. Since the LINDO uses the branch and bound algorithm, the final solutions should be the optimal results and they can be supplied as benchmarks. However, only problems 1 and 5 achieve the final optimal results. There are 812 integer variables and 841 constraints in the problem 1. LINDO obtains the optimum at 1,799,391 pivots and ends at 2,049,575 pivots after the computing time about three and a half-hour on a Pentium 133 PC. The problem 5 takes about six hours. The other problems can not achieve the final optimal solutions after several times of re-computation. The temporary solutions are taken as the answers. Problems 6 and 7 have too many constraints to be solved by this LINDO hyper version. The results are shown at the column for LINDO in Table 4.

Most of the solutions from GA are very close to the results of LINDO. In problem 1, the solution from GA is nearly the same as the optimum solution from LINDO. Only two nodes are clustered in two different groups. Furthermore, the coding of a chromosome allows the GA to derive solutions with the number of medical stations as desired. Therefore, evaluating the feasibility of using fewer medical stations is relatively easy. Although using fewer medical stations implies a more tightly constrained problem, the GA results, which using fewer medical

stations, are still competitive with the results of LINDO as demonstrated in Table 4.

Table 4. Results comparison of the first problems set

Prob	Groups	Best of GA	LINDO
1	5	□959.30	959.299
		#0.0001%	
2	11	1106.6	*1003.85
		10.24%	
	10	1171.70	*1142.72
		2.54%	
	9	1404.32	*1341.24
		4.7%	
3	10	1230.6	*1241.40
		-0.8%	
	9	1531.0	*1425.71
		7.38%	
4	16	836.899	*746.74
		12.1%	
5	5	1783.79	1686.83
		5.7%	
6	7	1818.63	+N.A.
	6	2238.81	N.A.
7	13	1386.51	N.A.

□Objective Value

Percent deviation from LINDO solution.

*LINDO does not end with an optimal solution.

+The solution is not available, because this version of LINDO can not solve this problem.

6. CONCLUSION

This study applies a genetic algorithm to solve the capacitated clustering problem by a general-purpose GA program library, SUGAL. In the computational study, severn test problems are solved to evaluate the performance of the proposed GA method. The binary coding for this problem does not require modifying the genetic operators, and can inherently consider most of the constraints. Experimental results demonstrate that an adaptive penalty function applied to handle the capacity constraint can effectively guide the search direction. These two approaches enhance the solution quality and computational efficiency of GA for solving the CCP. Experimental results further demonstrate that the solution quality of GA is very competitive with LINDO and the computational time is significantly less than LINDO.

In this study, the derived locations of medical stations have the minimum total assignment distance to each habitant node. However, another critical factor should be the travel time from any habitant node to the medical stations. The further study might use the objective function of travel time and adds one more time constraint for each node. Correspondingly, the emergency evacuation plan is another similar problem and deserves further attention.

The CCP model can also be extended further. In equation (1) of the CCP formulation, the objection function does not consider the marginal cost of using more medical stations (groups). Therefore, the objective value may be lower if more medical stations are available. Further studies should consider the total cost of the facility and the assignment cost to simultaneously determine the number of clusters and the members within a cluster. Modifying the model proposed herein can be easily achieved by adding the medical stations cost to the objective function, or one more penalty for the use of more medical stations.

REFERENCES

[1] D. W. Coit, A. E. Smith and D. M. Tate, "Adaptive penalty methods for genetic optimization of constrained combinatorial problems," *INFORMS Journal on Computing,* Vol. 8, pp. 173-182. 1996.

[2] K. A. De Jong. *An analysis of the Behavior of a Class of Genetic Adaptive Systems*, Doctoral dissertation, University of Michigan, Ann Arbor, MI. 1975.

[3] A. Fairley. *Comparison of Methods of Choosing the Crossover Point in the Genetic Crossover Operation*, Technical Report, Department of Computer Science, University of Liverpool, Liverpool, UK. (1991).

[4] M. L. Fisher and R. Jaikumar, "A generalized assignment heuristic for vehicle routing," *Networks*, Vol. 11, pp. 109-124. 1981.

[5] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*, Freeman, San Francisco. 1979.

[6] J. J. Grefenstette. "Optimization of control parameters for genetic algorithms," *IEEE Transactions on Systems, Man and Cybernetics*, Vol. 16, pp. 122-128. 1986.

[7] J. H. Holland *Adaptation in Natural and Artificial Systems*, University of Michigan Press, Ann Arbor, MI., re-issued by MIT Press. 1992.

[8] A. Hunter. *SUGAL Programming Manual*, University of Sunderland, England. 1995.

[9] A. Hunter. *SUGAL User Manual*, University of Sunderland, England. 1995.

[10] I. Koskosidis and W. B. Powell, "Clustering algorithms for consolidation of node orders into vehicle shipments," *Transportation Research-Part B*, Vol. 26, pp. 365-379. 1992.

[11] I. Koskosidis. *Optimization Based Models and Algorithms for Routing and Scheduling with Time Window Constraints*, Doctoral dissertation, Department of Civil Engineering and Operations Research, Princeton University, New Jersey 1988.

[12] J. Mulvey and M. P. Beck, "Solving capacitated clustering problems," *European*

Journal of Operational Research, Vol. 18, pp. 339-348. 1984.

[13] I. H. Osman, and N. Christofides, "Capacitated clustering problems by hybrid simulated annealing and tabu search," *International Transactions in Operational Research*, Vol. 1, pp. 317-336. 1994.

[14] J. D. Schaffer, R. A. Caruana, L. J. Eshelman, and R. Das, "A study of control parameters affecting online performance of genetic algorithms for function optimization," *Proceedings of 3rd International Conference on Genetic Algorithm*, Morgan Kaufmann Publisher, San Mateo, CA, pp. 51-60. 1989.

[15] S. R. Thangiah and A. V. Gubbi, "Effect of genetic sectoring on vehicle routing problems with time windows," *IEEE International Conference on Developing and Managing Intelligent System Projects*, pp. 146-153. 1993.