

# Solving Bus Terminal Location Problem Using Genetic Algorithm

S. Babaie-Kafaki, R. Ghanbari, S.H. Nasser, and E. Ardil

**Abstract**—Bus networks design is an important problem in public transportation. The main step to this design, is determining the number of required terminals and their locations. This is an especial type of facility location problem, a large scale combinatorial optimization problem that requires a long time to be solved.

The genetic algorithm (GA) is a search and optimization technique which works based on evolutionary principle of natural chromosomes. Specifically, the evolution of chromosomes due to the action of crossover, mutation and natural selection of chromosomes based on Darwin's survival-of-the-fittest principle, are all artificially simulated to constitute a robust search and optimization procedure.

In this paper, we first state the problem as a mixed integer programming (MIP) problem. Then we design a new crossover and mutation for bus terminal location problem (BTLP). We tested the different parameters of genetic algorithm (for a sample problem) and obtained the optimal parameters for solving BTLP with numerical try and error.

**Keywords**—Bus networks, Genetic algorithm (GA), Location problem, Mixed integer programming (MIP).

## I. INTRODUCTION

GENETIC algorithms are nondeterministic stochastic search/optimization methods that utilize the theories of evolution and natural selection to solve a problem within a complex solution space.

As a brief comment on biological background, we can state that all living organisms consist of cells. In each cell, there is the same set of chromosomes. A chromosome's characteristic is determined by the gene. The set of chromosomes is called the population.

Genetic algorithms are the member of a wider family of algorithms, Evolutionary Algorithms (EA). The algorithms in EA share a common conceptual base of simulating the evolution of individual structures via processes of selection, crossover and mutation. The processes depend on the

perceived performance of the individual structures as defined by an environment.

GAs maintain a population of structures, that evolve according to rules of selection and other operators, that are referred to as "search operators" such as crossover (or recombination) and mutation. Each individual in the population receives a measure of its fitness in the environment. In the optimization problems, fitness of a chromosome is usually the value of its objective function. Selection focuses attention on high fitness individuals.

In this paper, we first formulate the BTLP in section 2 and then in section 3 we explain a way that the BTLP is encoded and prepared for solving with GA. Moreover, we state a way that the natural operators such as selection, crossover and mutation are simulated in sections 4 and 5. Finally, in section 6 we solve a sample problem and report some interesting results.

## II. FORMULATION OF BTLP AS AN MIP

A main step to bus networks design, is determining the number of required terminals and their locations. The BTLP is an especial type of facility location problem, a large scale combinatorial optimization that requires a long time to be solved.

For stating the BTLP, we consider a set of nodes (with their coordinates) in a city. We suppose that the number of enter and exit of passengers in every node (that is called potential of the node) is available. Also, we suppose that if a node is accepted as a bus terminal, it can service the other nodes that are located in its neighborhood.

**Definition 2.1.** The neighborhood of node  $i$  (that we named it  $J_i^*$ ) is the set of all nodes  $j$  such that the distance between  $i$  and  $j$  is less than or equal to  $r$  ( $r$  is a constant that is called the radius of the neighborhood).

Parameters of the BTLP are:

$J$  : The set of all nodes of the network.

$I$  : A subset of  $J$  that contains the candidate nodes for making bus terminals.

$c_{ij}$  : Distance between nodes  $i$  and  $j$ .

$d_j$  : Potential of node  $j$  for  $j \in J$ .

$k$  : Number of required terminals.

$f(c_{ij})$  : A decreasing exponential function of  $c_{ij}$  (in this

paper we have:  $f(c_{ij}) = e^{-c_{ij}}$ ).

S. Babaie-Kafaki is with (1) Islamic Azad University- Science and Research Branch- member of young research club, (2) Department of Mathematical Sciences, Sharif University of Technology, Tehran, Iran (e-mail: kafaki@math.sharif.ir).

R. Ghanbari is with Department of Mathematical Sciences, Sharif University of Technology, Tehran, Iran (e-mail: rghanbari@math.sharif.ir).

S.H. Nasser is with Department of Mathematical Sciences, Sharif University of Technology, Tehran, Iran (e-mail: nasser@math.sharif.ir).

E. Ardil is with Department of Computer Engineering, Trakya University, Edirne, Turkey (e-mail: ebruardil@trakya.edu.tr).

$J_i^*$ : A subset of  $J$  that contains all nodes that can be serviced from node  $i \in I$ .

$B$ : A very large number.

Variables of BTLP are:

$x_{ij}$ : The value of service that node  $j \in J_i^*$  can received from node  $i \in I$ .

$y_i$  (Binary variable):  $y_i = 1$ , if node  $i \in I$  is accepted as a bus terminal, else  $y_i = 0$ .

The BTLP is formulated as follows:

$$\begin{aligned} \text{Max} \quad & \sum_{i \in I} \sum_{j \in J_i^*} d_j f(c_{ij}) x_{ij} \\ \text{s.t.} \quad & \sum_{i \in I: j \in J_i^*} x_{ij} \leq 1 \quad \forall j \in J - I \\ & \sum_{j \in J_i^*} x_{ij} \leq B \cdot y_i \quad \forall i \in I \quad (2.1) \\ & \sum_{i \in I} y_i = k \quad (2.2) \\ & x_{ij} \geq 0 \quad \forall i \in I, j \in J_i^* \\ & y_i = 0, 1 \quad \forall i \in I \end{aligned}$$

The constraint (2.1) shows that just if node  $i$  is accepted as a bus terminal ( $y_i = 1$ ), it can service the other nodes in its neighborhood. Also, the constraint (2.2) controls the number of required terminals. As an important point, if node  $j$  in a feasible solution of BTLP is located in  $m$  neighborhoods related to  $i_1, \dots, i_m \in I$ , then the values of  $x_{ij}$  for  $i = i_1, \dots, i_m$ , depends on the distances directly.

### III. ENCODING

The chromosomes should in some way contain information about the feasible solutions of BTLP. In this paper, we use binary encoding that is due to [6]. In binary encoding every chromosome is a string of bits (genes), equal to 0 or 1.

We explain the procedure of encoding in this paper by an example. Assume that  $k = 2$  and the candidate nodes (among 60 nodes) are: 1, 6, 19, 44, and 58, respectively. Then consider chromosome  $y$  in the following form:

1	0	1	0	0
---	---	---	---	---

This chromosome shows a feasible solution in which just nodes 1 and 19 are accepted as bus terminals.

It is obvious that each chromosome is just related to one feasible solution and conversely, each feasible solution is just related to one chromosome, so for a chromosome like  $y$ , we can compute the value of objective function as its fitness.

### IV. SELECTION

According to Darwin's evolution theory, the best individuals should survive and create new offspring. There are many methods for selection the best chromosomes from a population, for example roulette wheel selection, Boltzman selection, tournament selection, rank selection, steady state selection and so on. In this paper we use roulette wheel selection.

In roulette wheel selection, parents are selected according to their fitness. The better the chromosomes are, the more chance to be selected they have.

Imagine a roulette wheel (pie chart) where all chromosomes in the population are placed according to their normalized fitness. Then a random number is generated that decides which chromosome should be selected. Chromosomes with bigger fitness values will be selected more times since they occupy more space on the pie.

### V. Crossover AND MUTATION

Selection alone can not introduce any new chromosomes into the population, i.e., it can not find new points in the search space. These are generated by genetically-inspired operators, of which the most well known are crossover and mutation. Crossover is sometimes referred to as recombination, too.

The crossover and mutation are most important parts of a genetic algorithm. The performance of the algorithm is mainly influenced by these operators. Usually, there is predefined probability of procreation via each of these operators. Traditionally, these probability values are selected such that crossover is the most frequently used, with mutation being resorted to only relatively. Here we named the probability related to crossover by  $pc \in [0.8, 0.99]$  and the probability related to mutation by  $pm \in [0.01, 0.20]$ . Of the two operators, mutation involves only a single parent and result in the creation of a single offspring. The standard crossover operator called simple crossover has numerous variants such as partially-mapped, position-based, order-based, subtour chunking, cyclic, acyclic, inversion, and edge-recombination crossovers. All of them involve two parents. The detailed description and discussion of the various types of crossover operators is given in [4].

Crossover takes two chromosomes and combines them (with a random process) to produce a new chromosome (or two chromosomes). In this paper for performance of the crossover on two chromosomes  $y'$  and  $y''$ , we keep the common 0 and 1 in the chromosomes  $y'$  and  $y''$  (as good (1) or bad (0) properties of the chromosomes) for the new chromosome  $y$ , and then for necessary positions (genes) equal to 1 (that we named them full positions) to complete  $y$  as a chromosome with  $k$  full positions, we use an alternative and stochastic procedure.

For example, consider the chromosomes  $y'$  and  $y''$  in the following forms:

$Y'$ : 

1	0	1	0	0	0	0	1	1	1	0	1
---	---	---	---	---	---	---	---	---	---	---	---

$Y''$ : 

1	1	0	0	0	1	1	0	1	0	0	1
---	---	---	---	---	---	---	---	---	---	---	---

The output of the crossover operator on these chromosomes may be in this form:

$Y$ : 

1	1	0	0	0	0	0	1	1	1	0	1
---	---	---	---	---	---	---	---	---	---	---	---

Mutation is applied to each child usually after crossover. It randomly alters each gene with a small probability (usually less than 0.2). Mutation provides a small amount of random search, and helps to ensure that no point in the search space has a zero probability of being examined.

In this paper, for the mutation operator on chromosome  $y$ , we chose randomly a full position of chromosome  $y$  and change it to zero, instead we consider index  $K$  such that:

$$\sum_{j \in J_K^*} d_j = \max_{\{i: y_i=0\}} \sum_{j \in J_i^*} d_j$$

and let  $y_K = 1$ . This mutation may improve the objective function, too.

VI. NUMERICAL RESULTS

In this section, we consider a test problem with these properties:

$$|J| = 60, |I| = 20, k = 5.$$

We have the points in  $J$  with their coordinates, so it is easy to determine the distances. We prepared the software to simulating the operators in GA with MATLAB. We tried to determine the optimal values of  $pc$  and  $pm$ . So, we changed these parameters in the intervals that we explained in section 4.

Our study shows that the optimal parameters are  $pc = 0.93$  and  $pm = 0.02$ . As a result, the plot of increase of objective function with respect to the number of iterations of GA with optimal parameters is shown. Note that, in the Table I the parameters are defined as follows:

*I*VOF (Initial Value of Objective Function): Optimal value of objective function in the initial population,

*F*VOF (Final Value of Objective Function): Optimal value of objective function in the final population,

*Avg*: Average of a column.

For the optimal parameters the results are:

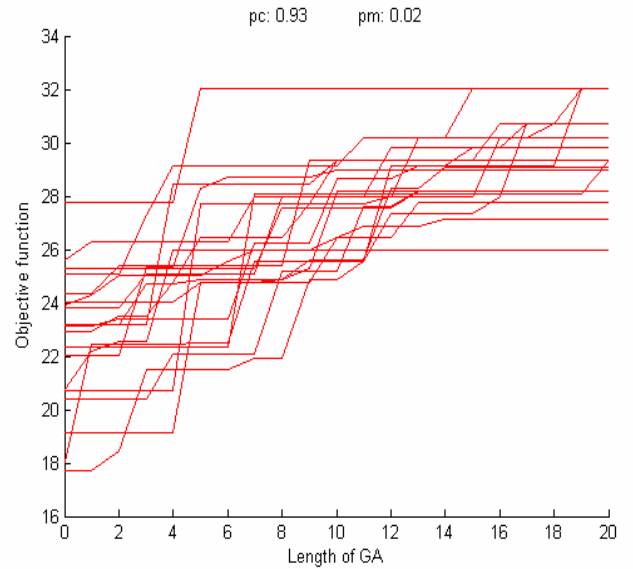


Fig. 1 The plots of objective function in different run of GA with the optimal parameters

TABLE I  
NUMERICAL RESULTS FOR THE OPTIMAL PARAMETERS

<b>pc: 0.93</b>		<b>pm: 0.02</b>
<b>I</b> VOF	<b>F</b> VOF	<b>T</b> ime(S)
23.16	29.10	0.86(S)
27.79	32.02	0.05(S)
24.00	28.18	0.09(S)
22.04	29.00	0.02(S)
20.78	27.11	0.11(S)
20.38	28.18	0.02(S)
22.93	29.32	0.02(S)
25.27	28.18	0.02(S)
25.08	32.02	0.02(S)
23.19	29.32	0.02(S)
23.90	25.98	0.02(S)
24.35	32.02	0.02(S)
20.69	30.18	0.02(S)
19.14	27.79	0.02(S)
23.12	32.02	0.02(S)
22.36	29.32	0.02(S)
18.02	30.69	0.02(S)
25.58	29.81	0.02(S)
17.71	30.69	0.02(S)
23.82	30.69	0.02(S)
<b>Avg: 29.58</b>		<b>Avg: 0.07(S)</b>

Also some useful results are as follows:

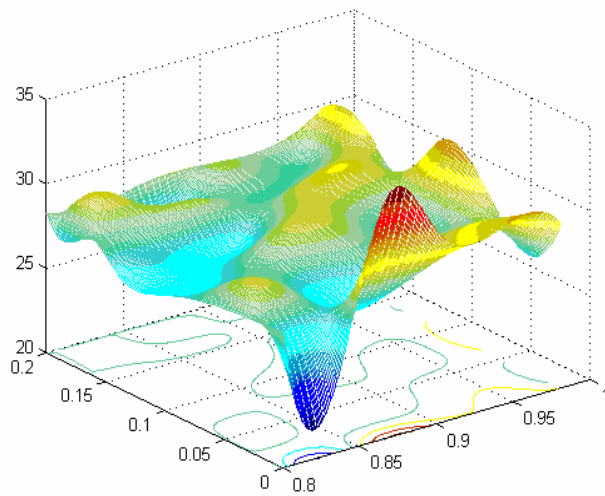


Fig. 2 Plot of the average values of objective function with respect to  $pc$  and  $pm$

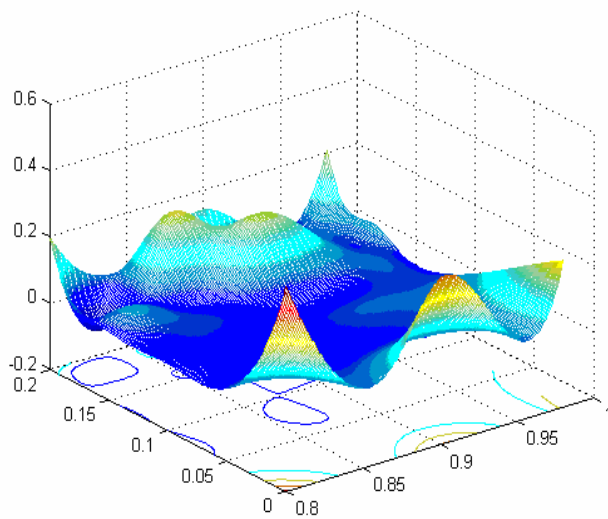


Fig. 3 Plot of the average values of the time with respect to  $pc$  and  $pm$

#### REFERENCES

- [1] H. Aashtiani, B. Hejazi, *Solving Bus Terminal Location Problem Using Simulated Annealing Method*, to appear in: Esteghlal, Volume 2, March 2001 (in Persian).
- [2] Stephen P. Bradley, Arnoldo C. Hax, Thomas L. Magnanti, *Applied Mathematical Programming*, ADDISON-WESLEY PUBLISHING COMPANY, 1976.
- [3] Robert S. Garfinkel, George L. Nemhauser, *Integer Programming*, JOHN WILEY & SONS, 1972.
- [4] D.E. Goldberg, *Genetic Algorithms in Search, Optimization and Machine Learning*, ADDISON-WESLEY PUBLISHING COMPANY, 1989.
- [5] D. Goldberg and R. Lingle, *Alleles, loci and traveling salesman problem*, In J.J. Grefenstette, editor, *Proceedings of International Conference on GAs*, Lawrence Erlbaum, 1985.
- [6] J.H. Holland, *Adaptation in Natural and Artificial Systems*, The University of Michigan Press, Ann Arbor, Michigan, 1975.
- [7] Alden H. Wright, *Genetic Algorithm for Real Parameter Optimization*, to appear in: *Foundations of Genetic Algorithms*, 1991.
- [8] R. Ghanbari, *Jaszkievicz's Genetic Local search for Solving Multi-Objective Combinatorial Optimization*, M.Sc. Thesis, Supervisor: N. Mahdavi-Amiri, Department of Mathematical Sciences, sharif University of Technology, November 2004.