



Solving bus terminal location problems using evolutionary algorithms

Reza Ghanbari, Nezam Mahdavi-Amiri*

Faculty of Mathematical Sciences, Sharif University of Technology, Tehran, P.O. Box 11365-94, Iran

ARTICLE INFO

Article history:

Received 12 December 2007

Accepted 17 January 2010

Available online 4 February 2010

Keywords:

Bus terminal location problem

Evolutionary algorithm

Memetic algorithm

Transportation

Simulated annealing

ABSTRACT

Bus terminal assignment with the objective of maximizing public transportation service is known as bus terminal location problem (BTLP). We formulate the BTLP, a problem of concern in transportation industry, as a p-uncapacitated facility location problem (p-UFLP) with distance constraint. The p-UFLP being NP-hard (Krarup and Pruzan, 1990), we propose evolutionary algorithms for its solution. According to the No Free Lunch theorem and the good efficiency of the distinctive preserve recombination (DPX) operator, we design a new recombination operator for solving a BTLP by new evolutionary and memetic algorithms namely, genetic local search algorithms (GLS). We also define the potential objective function (POF) for the nodes and design a new mutation operator based on POF. To make the memetic algorithm faster, we estimate the variation of the objective function based on POF in the local search as part of an operator in memetic algorithms. Finally, we explore numerically the performance of nine proposed algorithms on over a thousand randomly generated problems and select the best two algorithms for further testing. The comparative studies show that our new hybrid algorithm composing the evolutionary algorithm with the GLS outperforms the multistart simulated annealing algorithm.

© 2010 Elsevier B.V. All rights reserved.

1. Introduction

Consider n nodes (representing bus stations, metro stations, etc.) with known number of passengers. The aim is to locate a pre-specified number of locations (nodes) from among m terminals with known reachable neighborhoods so that the public service is maximized. This is called a bus terminal location problem (BTLP). It is clear that a BTLP is a special facility location problem; the reader is referred to Mirchandani and Francis [1], Korte and Vygen [2], Krarup and Pruzan [3], and Drezner and Hamacher [4]. There are several algorithms in the literature to solve different kinds of FLPs; e.g., exact algorithms, approximation algorithms, heuristics and metaheuristics. Many kinds of FLPs are NP-hard [5], and thus the exact algorithms (see [1,6], for comprehensive surveys) can only solve small instances of an FLP in a practical time. As approximate algorithms, Shmoys et al. [7] presented the first polynomial-time algorithm with a constant approximation factor. After 1997, other approximation algorithms have been developed (see [8–10]). Unfortunately, Guha and Khuller [11] established a lower bound for the approximation factor. Local searches and heuristics in all categories of FLP are active areas in combinatorial optimization. The first local search was introduced by Kuhlen and Hamburger [12]. Several authors developed local searches and

heuristics on FLP (see [13]). Other algorithms on FLP are dual-based methods (see [14–16]), hybrid multistart heuristics [17], variable neighborhood search [18] and others ([19,20] and [21]).

Since the early years of operations research studies, metaheuristics have been widely used to solve a variety of practical problems in multi-objective optimization (see [22,23]) or single objective optimization (see [24,25]). Many authors showed that metaheuristics are more efficient than other existing algorithms on other categories of FLP; e.g., simulated annealing by Alves and Almeida [26] and genetic algorithm by Kratica et al. [27].

Evolutionary algorithms, that is, genetic based algorithms using specialized data structures, have been successfully used to provide optimal or near optimal solutions for different optimization problems [28]. Genetic local search (GLS), sometimes named memetic algorithm, is a hybrid heuristic approach that combines the advantages of population-based search and local optimization. GLS is widely used in combinatorial optimization problems (see [29–32]). Based on the No Free Lunch (NFL) theorem [33] and global convexity property of combinatorial optimization discussed by Jaszkiwicz and Kominek [32], proper definitions of the genetic operators have critical influence on the performance of evolutionary algorithms and genetic local search for combinatorial optimization problems.

Here, we describe new evolutionary algorithms for solving a real life problem, the so-called bus terminal location problem, an important problem in transportation industries which can be formulated as a p-UFLP with distance constraint.

Based on the efficiency of the distance preserving recombination operator used in [30–32], we construct a new recombination

* Corresponding author. Tel.: +98 21 66165607; fax: +98 21 66005117.

E-mail addresses: rghanbari@mehr.sharif.edu (R. Ghanbari), nezamm@sharif.edu (N. Mahdavi-Amiri).

operator. On the other hand, we define a property of each facility named as potential objective function (POF) and then construct a new mutation operator based on the POF. In GLS, local searches are costly; therefore, we use the special stopping criteria in local search. Since, in local search, we must calculate value variations (ΔF) of the objective function, to select better solutions, we use an approximation of ΔF based on POF. We will save time in local search, if we can approximate ΔF .

The remainder of our work is organized as follows: Section 2 describes a mathematical model of BTLP. In Section 3, we explain the genetic algorithm and genetic local search. In Section 4, we describe, in detail, the adaptation of our algorithms to BTLP. Construction of test problems and computational experiments are reported in Section 5. Section 6 gives the concluding remarks.

2. Mathematical model

Consider J as a set of nodes (specified with their coordinates) in a city. Suppose that the number of entrances and exits of passengers in every node (potential of node) is available. Also, suppose I is the set of candidate nodes for bus terminals where each terminal, if established, can service nodes being in its reachable neighborhood. The setup costs of bus terminals are assumed to be equal, and thus, without loss of generality, we consider the setup costs as zero. The objective is to select p terminals to maximize the service function, where the service function is defined as in Definition 2.2 below.

Definition 2.1. For $i \in I$, the neighborhood of node i is shown by J_i^* and defined by

$$J_i^* = \{j \in J : c_{ij} \leq r\}, \quad (1)$$

where r is a constant and c_{ij} is the distance between node j and terminal i .

Let

$$J^* = \bigcup_{i \in S} J_i^*,$$

where $S \subseteq I$.

The service function is defined next.

Definition 2.2. Assume that the potential of node j is shown by d_j , the distance between node j and terminal i is shown by c_{ij} and f

is a decreasing function. For $S \subseteq I$, the service function is defined to be:

$$F(S) = \sum_{j \in J^*} d_j \times f(\min_{i \in S} \{c_{ij}\}). \quad (2)$$

Definition 2.3. Potential of object function (POF) for $i \in I$ is defined to be:

$$\text{POF}(i) = \sum_{j \in J_i^*} d_j \times f(c_{ij}). \quad (3)$$

The combinatorial formulation of BTLP can now be specified as follows.

Problem: Bus terminal location problem.

Instance: $I = \{i_1, i_2, \dots, i_m\}$, the set of candidate nodes (terminals) and $J = \{j_1, j_2, \dots, j_n\}$, the set of nodes in the city, $C = [c_{ij}]$, the distance matrix, f , a decreasing function, p , number of terminals to be set up, and $D = \{d_1, d_2, \dots, d_n\}$, the set of potential nodes corresponding to J .

$$\text{Output: } \arg \max \left\{ F(S) = \sum_{j \in J^*} d_j \times f(\min_{i \in S} \{c_{ij}\}) : S \subseteq I, |S| = p \right\}.$$

3. Genetic/evolutionary algorithm and genetic local search

3.1. Genetic/evolutionary algorithm

Genetic algorithms are search algorithms based on the mechanics of natural selection and natural genetics [34]. Evolutionary algorithms (EAs) are genetic algorithms with special data structures or special encodings of solutions or genetic operator based on the problem [28]. In Fig. 1, a genetic/evolutionary algorithm is given in general.

Fig. 1 is a general template for genetic/evolutionary algorithms. Based on NFL theorem (see [33]), for an optimization problem, operators and parameters of genetic/evolutionary algorithm must be carefully defined (determined). In the next section, we adapt the genetic/evolutionary algorithm for solving the BTLP.

```

Algorithm GA (EA).
Parameters: stopping criterion,  $np$  as a size of population,  $P_m, P_c$ .
Begin
  Initialize population  $P$ .
  Repeat
    Generate template population  $TP$  :
       $TP = \emptyset$ .
      For  $i:=1$  till  $np$  do
        Selection: Select two parents  $a, b \in P$  with a selection operator.
        Recombination: Generate probability  $r \in [0,1]$  randomly. If  $r < P_c$  then let  $c := \text{Recombination}(a,b)$  else with equal probability either set  $c := a$  or set  $c := b$ .
        Mutation: With probability  $P_m$ , let  $c := \text{mutation}(c)$ .
         $TP := TP \cup \{c\}$ .
      Endfor
    End of generate  $TP$ .
    Construct  $newP$  from  $P$  and  $TP$  by a selection rule.
    Let  $P := newP$ .
  Until stopping criterion is satisfied.
End.

```

Fig. 1. A genetic algorithm.

```

Procedure GLS.
Parameters: stopping criterion,  $np$  as the size of population,  $P_m, P_c$ .
Begin
  Initialize population  $P$ .
  For all  $p \in P$  : let  $p = Local\ Search(p)$ .
  Repeat
    Generate template population  $TP$  :
       $TP = \emptyset$ .
      For  $i:=1$  till  $np$  do
        Selection: Select two parents  $a, b \in P$  with selection operator.
        Recombination: Generate probability  $r \in [0,1]$  randomly. If
           $r < P_c$  then let  $c := Recombination(a,b)$  else with equal
          probability either set  $c := a$  or set  $c := b$ .
        Mutation: With probability  $P_m$ , let  $c := Mutation(c)$ .
        Local Search:  $c := Local\ search(c)$ .
         $TP := TP \cup \{c\}$ .
      Endfor
    End of generate  $TP$ .
    Construct  $newP$  from  $P$  and  $TP$  by a selection rule.
    Let  $P := newP$ .
  Until stopping criterion is satisfied.
End.

```

Fig. 2. A genetic local search.

3.2. Genetic local search

The idea of combining a genetic algorithm and heuristics to solve combinatorial optimization problems has been investigated by many researchers in the past two decades.

Genetic local search, sometimes named memetic algorithm, is a hybrid heuristic that combines the advantages of genetic algorithm and local search. GLS is used for solving many combinatorial optimization problems [29–32]. In GLS, the local search finds good solutions (not generally local optimal; e.g., [35]), and then genetic algorithm works on improving the solutions. In some cases, local heuristics or extensions of local search are used instead of local search; e.g., Taillard [36] used tabu search as a local heuristic. In Fig. 2, a GLS is given in detail.

Here, we assume that the local search procedure gives a good solution with no necessity to stop at a local optimal point. We use some stopping criterion in local search as described in Section 4. GA (EA) and GLS need to be adapted to the BTLP. For adaptation, encoding of solutions, producing the initial population, genetic operators and local search algorithm (in GLS) are clearly defined.

4. Adaptations

4.1. Coding and selection

We use a special coding of solutions. We describe coding of a solution with an example. Suppose $m=5$ and $p=2$ and terminals No. 1 and No. 4 are located. Then, the chromosome of this solution is:

1	4
---	---

We select two parents by the standard roulette wheel with linear ranking [23], in the generation step of the genetic/evolutionary algorithm and GLS. In each iteration of genetic/evolutionary algorithm and GLS, we construct the new population by:

$$newP = TP \cup \{e \text{ percent of elites in } P\}, \quad (4)$$

where e is a predefined parameter.

4.2. Initial population

Authors usually use a random population for the initial population but the efficiency of the genetic/evolutionary algorithm and GLS is usually increased if the quality of the initial population is better than the average of the random population. To achieve better solutions in the initial population, we use potential objective function for each $i \in I$. We construct a solution by selecting p different nodes from I using the roulette wheel on POF of each node.

On the other hand, in evolutionary algorithms, we prefer to distribute the initial population in the search space in order for the evolutionary algorithm to reach the global optimum. Therefore, we select half of the initial population randomly. Using these considerations, we generate the initial population by procedure GLP given in Fig. 3.

4.3. Genetic operator

4.3.1. Recombination operator

Recombination combines good properties of parents to create better offsprings. In other words, recombination determines the regions of search space where better solutions exist. According to the global convexity, the good solutions have many common properties; for example, the good solutions of TSP instances have common arcs. Based on the global convexity, Merz and Freisleben [30,31], and Jaszkiwicz and Kominek [32] designed many types of the distinctive preserve (DPX) recombination operator for TSP, vehicle routing and QAP. We assume that BTLP has global convexity based on specifications of Merz and Freisleben [31] and design a new DPX operator for BTLP as given in Fig. 4 below.

4.3.2. Mutation operator

Merz and Freisleben [31] pointed out that the mutation operator should attempt to focus the search on randomly chosen regions so that the algorithm would be able to identify solutions that are hard to find by the DPX operator. On the other hand, the efficiency of genetic algorithm can be increased by the existence of more consistency between the mutation operator and the recombination operator. We design two mutation operators for solving BTLP

```

Procedure GIP.
Parameter:  $n$  as a size of population.
Begin
   $IP := \emptyset$ .
  Random phase:
  Repeat
    Generate solution  $s_i \notin IP$  randomly.
    Let  $IP := IP \cup \{s_i\}$ .
  Until  $|IP| < \lfloor \frac{n}{2} \rfloor$ .
  Heuristic phase:
  Repeat
    Generate solution  $s_i \notin IP$  by selecting  $p$  nodes from  $I$  using
    Roulette wheel on POF of each node.
    Let  $IP := IP \cup \{s_i\}$ .
  Until  $|IP| < n$ .
End.

```

Fig. 3. A procedure for generating the initial population.

```

Procedure DPX.
Parameters:  $a, b$ .
Begin
   $CommonTer := a \setminus b$ .
   $t := p - \lfloor CommonTer \rfloor$ .
   $a\Delta b := (a - b) \cup (b - a)$ .
   $S :=$  randomly chosen subset with  $t$  elements of  $a\Delta b$ .
   $of1 := CommonTer \cup S$ .
   $of2 := CommonTer \cup (a\Delta b - S)$ .
End.

```

Fig. 4. Distinctive preserve recombination for BTLP.

and compare their efficiencies with the standard 2-swap mutation operator. In Mutation 1 described in Fig. 5, we expect to achieve a better solution.

In Mutation 2, we consider y as a binary string such that its i -th bit will be one if $i \in I$ is open and zero otherwise. We note that this mutation operator has a similar version in other combinatorial optimization problems [28,31]. Fig. 6 shows Mutation 2 in detail.

4.4. Local search

Many local searches are reported in the literature for facility location problems. According to the good results reported by Ghosh [21], we choose the 2-swap local search. We define the neighborhood in BTLP for the 2-swap as follows.

Definition 4.1. Assume that $S \subset I$ and $|S| = p$. The set of neighborhoods of S is:

$$N(S) = \{T | T \subset I, |T| = p, |S - T| = 1\}. \quad (5)$$

Usually, local search is implemented in two ways: the first approach makes use of the first improvement encountered, and the second approach uses the best neighborhood algorithm [25]. Our numerical experiments show that the best neighborhood algorithm does not necessarily produce better solutions as compared to the first approach while requiring longer CPU time. So, we use the first improvement approach as the local search for 2-swap.

In addition, in our numerical experiments we found out that the local search was the most time consuming part of the GLS algorithm. Therefore, to control the CPU time being used, we use

```

Procedure Mutation 1.
Parameter:  $S$ .
Begin
   $M := \{1, \dots, m\}$ .
   $closeT := M - S$ .
  Select  $i \in S$  randomly.
   $j^* := \arg(\max_{j \in closeT} (POF(j)))$ .
   $S := S - \{i\} \cup \{j^*\}$ .
  Return  $S$ .
End.

```

Fig. 5. Mutation 1.

Procedure Mutation 2.
Parameters: y as a binary string and r_0 as an integer number less than $\frac{m}{2}$.
Begin
 Select $l \in \{1, \dots, \frac{m}{2}\}$ randomly.
 Select $r \in \{1, \dots, r_0\}$ randomly.
 Invert substring of y from position l to position $l + r$.
 Return y .
End.

Fig. 6. Mutation 2.

number of iterations, number of improvements and maximum run time as the stopping criteria in local search as a component of GLS [25].

On the other hand, since the GLS algorithm is a combination of genetic algorithm and local search, we stop the local search when a good improvement in proportion to the initial solution is at hand. In this case, genetic algorithm works on the good solutions achieved by the local search.

Finally, we use the algorithm given in Fig. 7 with little change in the standard 2-swap as a local search for the GLS algorithm. In this algorithm, we consider $OpenT$ and $CloseT$ as arrays also, and for convenience, we use operator $+_c$ as in Definition 4.2 below.

Definition 4.2. Suppose $a, b \in N \cup \{0\}$ and $c \in N - \{1\}$. Then, operator $+_c$ is defined by:

$$a+_c b = \begin{cases} (a+b) - [(a+b)/c] \times c, & a+b > c \\ a+b & o.w. \end{cases} \quad (6)$$

It is necessary to note that computing the objective function is a hard work and uses expensive time. Thus, we use an estimate of the value of $\Delta F(OpenT, Temp) = \Delta F = F(Temp) - F(OpenT)$ in the local search procedure (see Fig. 7) by POF of each terminal.

Definition 4.3. Assuming $S, T \subset I$, $|T| = |S| = p$ and $|S - T| = 1$, there exist $s \in S$ and $t \in T$ such that $s \notin T$ and $t \notin S$, and the estimation of $\Delta F(S, T) = F(T) - F(S)$ is defined by:

$$\tilde{\Delta}F = \tilde{\Delta}F(S, T) = \text{POF}(t) - \text{POF}(s). \quad (7)$$

Remark. s and t in Definition 4.3 are unique.

Procedure Local Search.
Parameters: $OpenT$ as an initial solution, T_{\max} as the maximum of iterations, I_{\max} as the maximum of improvisations, $Time_{\max}$ as the maximum of CPU time, $pimprove$ as an improvement percentage.
Begin
 $CloseT := \{1, \dots, m\} - OpenT$.
 $i :=$ Select random number from $\{1, \dots, p\}$.
 $j :=$ Select random number from $\{1, \dots, m-p\}$.
 { $iter$ is number of iterations and $impro_iter$ is number of improvement obtained}
 Let $iter := 0$ and $impro_iter := 0$.
A:
 $i := i +_p 1$.
 $t_i := OpenT[i]$.
 $jj := 0$.
 Repeat
 $j := j +_{m-p} 1$ and $iter := iter + 1$.
 $Temp := OpenT \cup \{j\text{-th element of } CloseT\} - \{t_i\}$.
 If $F(Temp) - F(OpenT) > 0$ **then** $impro_iter := impro_iter + 1$ and $improved := (F(Temp) - F(OpenT)) / F(OpenT)$.
 Swap $OpenT[i]$ and $CloseT[j]$.
 If $improved > pimprove$ or $impro_iter > I_{\max}$ **then**
 Return $OpenT$ and **Stop**.
 Endif
 $pimprove := (pimproved - improved) / (1 + improved)$.
 $jj := 0$.
 Endif
 If run time $> Time_{\max}$ or $iter > T_{\max}$ **then**
 Return $OpenT$ and **Stop**.
 Endif
 $jj := jj + 1$.
 Until $jj < m-p$
 Go to A.
End.

Fig. 7. A local search.

Table 1
Groups of test problems.

Group No.	Coordinates of the square				d_{max}	Scale of neighborhood radius (S_r)
	x_{min}	y_{min}	x_{max}	y_{max}		
1	-100	-100	100	100	$2\sqrt{2} \times 100$	100
2	-10	-10	10	10	$2\sqrt{2} \times 10$	10
3	-1	-1	1	1	$2\sqrt{2} \times 10$	1
4	-10	-10	10	10	$2\sqrt{2} \times 100$	10
5	-100	-100	100	100	$2\sqrt{2} \times 1000$	100
6	-1	-1	1	1	$2\sqrt{2} \times 0.1$	1

In the calculation of $\Delta F(S,T)=F(T)-F(S)$, when $s \in S$ is closed and $t \in T$ is open, nodes are divided into four groups as follows here: Nodes with serving center not changed (like node 3 in Fig. 8). Nodes served before but not being served now (like node 1 in Fig. 8). Nodes not served before but being served now (like node 5 in Fig. 8). Nodes with serving center changed (like nodes 2 and 4 in Fig. 8).

Thus, $\tilde{\Delta}F$ is an approximation of $\Delta F(S,T)$, because in the calculation of $\tilde{\Delta}F$, we may not have a proper designation of serving centers to nodes (specially for nodes in groups 1 and 4).

5. Computational experiments

5.1. Medium sized problems

To select representative evolutionary and GLS algorithms we generated 6 groups of test problems testing the efficiency of our algorithms. We considered and implemented nine versions of our solution method in MATLAB 7.0 software environment. All implementations were run on a PC 3 GHz with 1 GB of RAM.

In constructing the test problems, we considered distance between nodes and terminals, scale of distances, density of nodes in area, the size of neighborhoods and scale of node potentials relative to the scale of distance between nodes. So, for each group of the test problems we considered a square in the plant with all the coordinates of the nodes and terminals created randomly using a uniform distribution. Each node’s potential had a random value between zero and d_{max} with a uniform distribution, where d_{max} was defined specifically for each group of the test problems. Table 1 specifies 6 groups of test problems used in our experiments.

To obtain the optimal solution for these test problems, we first used complete enumeration of the feasible solutions. We then ran our proposed algorithms (based on GA and GLS) for $p = m/4, m/2, 3m/4$, because the number of feasible solutions of BTLP is $\binom{m}{p}$ and we know that for the case $p = m/2$, the problem is at its worst case. On the other hand, $p = m/4$ can be a good test to determine the efficiency of the algorithms and estimation of function variations as defined in Definition 4.3. The value of $p = 3m/4$ is interesting in comparison with $p = m/4$, because the number of feasible solutions in both cases are approximately equal. In other words, we have $\binom{m}{m/4} \approx \binom{m}{3m/4}$, and so it is expected that the algorithms have competitive efficiencies. But in the case $p = 3m/4$, the density of terminals is more than the case $p = m/4$, and we can explore the efficiency of our proposed algorithms and estimation of function variations given by (7).

On the other hand, we solved each medium sized problem using various values of r in Definition 2.1 (values of r equal to $0.2 \times S_r, 0.5 \times S_r$ and $1 \times S_r$) and $n = 100, 200, 300, 500$ and 1000 . This way, we could study the effect of the size of the neighborhood of one terminal in the CPU Time and the quality of (7).

Our criteria for the efficiency and comparison of the algorithms are CPU Time and suboptimality as defined below:

$$\text{suboptimality}(S) = \frac{F(S^*) - F(S)}{F(S^*)}(100), \tag{8}$$

where S^* is the optimal solution of the problem.

Initially, an initial population of a fixed size was produced and fixed for all the algorithms. If an algorithm started with an initial population with size less than the fixed population, then we selected its corresponding initial population from the fixed initial population randomly. We used $f(x) = e^{-x}$ a decreasing function, for the Definition 1.2. The most important part of the implementation of a metaheuristic algorithm on a problem is the setting of its parameters. We implemented the genetic/evolutionary algorithm on test problem No. 1 with $m = 20, n = 100, r = 20, P_c \in \{0.9, 0.95, 0.98\}, P_m \in \{0.02, 0.05, 0.1\}$ and $e \in \{5, 10, 20, 30\}$, with e an percentage of the elites of the old population that was used in constructing the new population. We ran the algorithm 10 times and averaged the results. The best results were obtained with $P_c = 0.95, P_m = 0.1$ and $e = 20$.

In the genetic/evolutionary algorithm, we used the number of generations as the stopping criterion. We implemented GA/EA for different sizes of population and number of generations on test problem of group 1 with $m = 20, 25, 50, 75, n = 100, r = 20, p = m/2$ and ran the program 10 times. Then, we chose the best size of the population and the best number of generations in our implementation (see Table 2).

In GLS, we used improvement percentage, maximum CPU time, maximum number of improvements and number of iterations in addition to the size of population and number of generations. We considered 5 values for each parameter and ran each algorithm 5 times for the test problems in group No. 1 with $m = 20, 25, 50, 75, n = 100, r = 20$, and $p = m/2$. According to the results obtained, we set the parameters of GLS in our implementation (see Table 3).

Remark. To select the best parameters of GA/EA and GLS, we used the following approach to compute efficiency of the algorithms. Let a be an algorithm, s be the best solution obtained by algorithm a and t be the average CPU time for algorithm a . Then, the efficiency of algorithm a is defined to be:

$$\text{efficiency}(a) = e^{1-\text{Suboptimality}(s)} \times e^{-t}.$$

For each algorithm, we solved 1080 test problems 5 times. Table 5 shows the average of the results. We summarize the results in Table 5 by noting the sum of average of the results on all test problems.

Table 2
Parameters of genetic/evolutionary algorithm on BTLP.

	m			
	20	25	50	75
Size of population	15	15	40	60
Number of generations	10	10	20	50

Table 3
Parameters of GLS on BTLP.

m	Size of population	Number of generations	Improvement percentage	Maximum No. of iterations	Maximum No. of improvements	Maximum CPU time (s)
20	8	4	0.01	$1 \times (m - p)$	5	0.001
25	8	4	0.01	$1 \times (m - p)$	5	0.001
50	20	10	0.01	$1 \times (m - p)$	5	1
75	50	50	0.01	$1 \times (m - p)$	5	0.01

Table 4
Algorithms implemented for testing.

Name	Description	Mutation operator
EA1	Evolutionary algorithm (Fig. 1)	Mutation 1
EA2	Evolutionary algorithm (Fig. 1)	Standard 2-swap
EA3	Evolutionary algorithm (Fig. 1)	Mutation 2
GLS1	Genetic local search (Fig. 2)	Mutation 1
GLS2	Genetic local search (Fig. 2)	Standard 2-swap
GLS3	Genetic local search (Fig. 2)	Mutation 2
AGLS1	Genetic local search using (7)	Mutation 1
AGLS2	Genetic local search using (7)	Standard 2-swap
AGLS3	Genetic local search using (7)	Mutation 2

Remark. Our implemented algorithms are listed in Table 4 below.

Table 5 shows that the AGLS algorithms based on $\tilde{\Delta}F$ are more efficient in comparison with the GLS algorithms. We also see that while AGLS algorithms produce better solutions in comparison

with the GLS algorithms (the required CPU times in AGLS algorithms are about 0.2 of the GLS algorithms), it is not easy to compare AGLS and EA algorithms, because the solutions of EA algorithms are mostly more accurate than AGLS algorithms, but the CPU time used for EA algorithms are more than AGLS algorithms. According to the results obtained for both GLS and AGLS algorithms, the type of the mutation operator has no significant effect on suboptimality of solutions and CPU times, while for EA algorithms, the mutation operator is very decisive.

5.2. Large sized problems

Based on the results obtained for the nine algorithms on medium sized problems, we choose EA1 and AGLS1 as the representative of our proposed algorithms. To take advantage of both algorithms, we implemented a hybrid algorithm combining EA1 and AGLS1. The hybrid algorithm is based on EA1 and uses genetic local search using (7) in 20% of the population. This way, we will achieve accurate solutions in shorter times. We call this composite algorithm as

Table 5
Average results of all 1080 test problems for various algorithms.

	EA1	EA2	EA3	GLS1	GLS2	GLS3	AGLS1	AGLS2	AGLS3
CPU time(s)	17.07	17.1	17.08	26.05	26.03	26.04	5.68	5.69	5.67
Suboptimality	0.00006	0.00007	0.00007	0.00059	0.00059	0.00059	0.0005	0.0005	0.00051

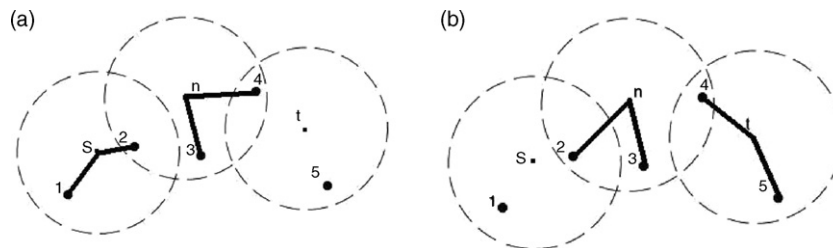


Fig. 8. In (a), the nodes s and n are open and in (b), the nodes n and t are open.

```

Algorithm Multi start simulated annealing (MSSA).
Parameter: stopping criterion.

Begin
    Set  $S := \emptyset$  and  $f^* := 0$ .
Repeat
    Select an initial solution  $p \notin S$  randomly and let  $S := S \cup \{p\}$ .
    Set  $p^* := SA(p)$  (see Figure 10).
    If  $F(p^*) > f^*$  then let  $f^* = F(p^*)$ .
Until stopping criterion is satisfied.
End.
    
```

Fig. 9. Multistart simulated annealing algorithm.

```

Algorithm Simulated Annealing (SA).
Parameter:  $x$  as an initial solution,  $T_0$  as an initial temperature,  $T_f$  as a final
temperature and  $L$ , the number of iteration in each temperature.
Begin
Set  $T := T_0$ .
Repeat
  For  $i=1$  till  $L$  do
    Select  $x_{new}$  from neighborhood of  $x$ .
    If  $F(x_{new}) > F(x)$  then set  $x := x_{new}$  else  $x := x_{new}$  with probability
      
$$e^{-\frac{F(x_{new}) - F(x)}{T \times F(x)}}$$

    End For.
  Set  $T := \alpha T$ .
Until  $T > T_f$ .
Return the best solution found.
End.

```

Fig. 10. Simulated annealing algorithm for maximization.

Table 6

Average results for EA1, HAGLS and MSSA on large sized problems.

$m(=n)$	p	Solution costs			CPU time (in minutes)		No. of best solutions found		
		EA1	HAGLS	MSSA	EA1 and MSSA	HAGLS	EA1	HAGLS	MSSA
250	64	119220.4	119188.3	118381.5	0.132604	0.132167	2	2	0
	125	124017.6	124045.8	123513.6	0.134167	0.129583	2	2	0
	187	124328.9	124300.2	124148.2	0.130833	0.123333	3	1	0
500	125	375074.4	377094	374902.8	1.329233	1.161525	1	3	0
	250	406949.1	407848.9	407819.6	1.325	1.14	0	2	2
	375	411095.5	411231.2	410947.5	1.291667	1.083333	0	4	0
750	187	873507.7	887684.2	875874.4	3.967917	3.282758	0	4	0
	375	980485.4	983936.2	979982.4	3.88625	3.154167	0	4	0
	562	995181.2	995704	995092.3	3.829167	3.033333	0	4	0
1000	250	1361313	1398179	1380022	9.252917	7.405417	0	4	0
	500	1601572	1611487	1603142	9.258333	7.291667	0	4	0
	750	1644308	1646697	1644934	8.9625	6.958333	0	4	0

HAGLS. We set the parameters of EA1 and HAGLS to be the same as the ones for the case $m = 75$. The number of iterations for EA1 is set to $0.8m$ and for HAGLS is set to $0.64m$.

To compare the efficiency of the algorithms for each m , we generated four problems (using the same conditions as the ones in the first group of Table 1). The performance of our algorithms was compared with the multistart simulated annealing (MSSA) (as given in Fig. 9), using $r = 20$. For SA (Fig. 10), after some numerical experiments, we set the initial temperature to 1400, $\alpha = 0.9$ and $L = 5$. The final temperature (T_f) is set to:

$$T_f = \frac{\varepsilon}{\ln(|S| - 1/\theta)},$$

with $\varepsilon = 0.1$, $\theta = 0.9$ and $|S| =$ number of feasible solutions [37].

We also used the neighborhood defined by (5) in the SA. We use the amount of the execution time spent by EA1 as the stopping criterion for MSSA. Table 6 shows the average cost solutions obtained for the algorithms, the average execution times for the algorithms on the four test problems, and number of the best solu-

tions found by each algorithm. It is evident that the new hybrid algorithm HAGLS outperforms both EA1 and MSSA algorithms in term of both execution times and number of best solutions found, specially on large sized problems (in 40 of the 48 problems, HAGLS found better solutions in shorter times).

6. Conclusions

We proposed algorithms, based on evolutionary and memetic algorithms, for solving the bus terminal location problem. We also defined a potential objective function for the nodes and used it in the proposed mutation operator and for estimating the variation of objective function in the local search as part of an operator in memetic algorithms. We proposed and tested a variety 9 algorithms on a collection of over a thousand medium sized problems and identified two representative algorithms, one being an evolutionary algorithm producing more accurate solutions and the other being a memetic algorithm performing a local search needing less execution time. Finally, for large sized problems, we proposed

a hybrid algorithm, composing the evolutionary algorithm with the local search applied to 20% of the population. The numerical test results on large sized problems showed the superiority of the hybrid algorithm over both the simple evolutionary algorithm and multistart simulated annealing on most problems.

Acknowledgement

The authors thank The Research Council of Sharif University of Technology for its support.

References

- [1] P.B. Mirchandani, R.L. Francis, *Discrete Location Theory*, Wiley-Interscience, New York, 1990.
- [2] B. Korte, B. Vygen, *Combinatorial Optimization: Theory and Algorithms*, Springer, 2006.
- [3] J. Krarup, M. Pruzan, The simple plant location problem: survey and synthesis, *European Journal of Operational Research* 12 (1985) 36–81.
- [4] Z. Drezner, H. Hamacher, *Facility Location: Applications and Theory*, Springer, 2002.
- [5] J. Krarup, M. Pruzan, Ingredients of location analysis, in: P.B. Mirchandani, R.L. Francis (Eds.), *Discrete Location Theory*, Wiley-Interscience, New York, 1990, pp. 1–54.
- [6] K. Holmberg, R. Mikael, D. Yuan, An exact algorithm for capacitated facility location problems with single sourcing, *European Journal of Operational Research* 113 (1999) 544–559.
- [7] D.B. Shmoys, É. Trados, K. Ardal, Approximation algorithms for facility location problems, in: *Proceedings of the 29th Annual ACM Symposium on Theory of Computing*, ACM Press, 1997, pp. 265–274.
- [8] K. Jain, M. Mahdian, E. Markakis, A. Saberi, V.V. Vazirani, Greedy facility location algorithms analyzed using dual facility with factor-revealing LP, *Journal of the ACM* 50 (2003) 795–824.
- [9] M. Mahdian, Y. Ye, J. Zhang, Improved approximation algorithms for metric facility location problems, in: *Proceeding of the 5th International Workshop on Approximation Algorithms for Combinatorial Optimization*, Lecture Notes in Computer Science (2462), Springer-Verlag, 2002, pp. 229–242.
- [10] F.A. Chudak, D. Williamson, Improved approximation algorithms for capacitated facility location problems, in: *Proceedings of the 7th Conference on Integer Programming and Combinatorial Optimization*, 1999, pp. 99–113.
- [11] S. Guha, S. Khuller, Greedy strikes back: improved facility location algorithms, *Journal of Algorithms* 31 (1999) 228–248.
- [12] A.A. Kuhen, M.J. Hamburger, A heuristic program for locating warehouses, *Management Science* 9 (1963) 643–666.
- [13] R.K. Madhukar, C.G. Plaxton, R. Rajaraman, Analysis of a local search heuristic for facility location problems, *Journal of Algorithms* 37 (2000) 146–188.
- [14] D. Erlenkotter, A dual-based procedure for uncapacitated facility location, *Operation Research* 26 (1978) 992–1009.
- [15] N. Mladenović, J. Brimberg, P. Hansen, A note on duality in the simple plant location problem, *European Journal of Operational Research* 174 (2006) 11–22.
- [16] F. Barahona, F. Chudak, Near Optimal Solution to Large Scale Facility Location Problems, Technical Report RC21606, IBM, Yorktown Height, NY, USA, 1999.
- [17] M.G.C. Resende, R.F. Werneck, A hybrid multistart heuristic for uncapacitated facility location problem, *European Journal of Operational Research* 174 (2006) 54–68.
- [18] P. Hansen, N. Mladenović, Variable neighborhood search: principles and applications, *European Journal of Operational Research* 130 (2001) 449–467.
- [19] M. Sun, Solving uncapacitated facility location problem using tabu search, *Computers & Operations Research* 33 (2006) 2563–2589.
- [20] L. Michel, V. Hentenryck, A simple tabu search for warehouse location, *European Journal of Operational Research* 157 (2003) 576–591.
- [21] D. Ghosh, Neighborhood search heuristics for uncapacitated facility location problem, *European Journal of Operational Research* 150 (2003) 150–162.
- [22] D.F. Jones, S.K. Mirrazavi, M. Tamiz, Multi-objective meta-heuristics: an overview of the current state-of-the-art, *European Journal of Operational Research* 137 (2002) 1–9.
- [23] K. Deb, *Multi-objective Optimization Using Evolutionary Algorithms*, John Wiley & Sons Ltd., 2001.
- [24] F. Glover, G.A. Kochenberger, *Handbook of Metaheuristics*, Kluwer Academic Publisher, 2003.
- [25] C.R. Reeves, *Modern Heuristic Techniques for Combinatorial Problems*, Blackwell Scientific Publication, Oxford, 1993.
- [26] M.L. Alves, M.T. Almeida, Simulated annealing algorithm for simple plant location problem: a computational study, *Revista Investigação Operacional* 12 (1992) 145–157.
- [27] J. Kratica, D. Tosic, V. Filipovic, I. Ljubic, Solving simple plant location problem by genetic algorithm, *RAIRO Operations Research* 35 (2001) 127–142.
- [28] Z. Michalewicz, *Genetic Algorithms + Data Structures = Evolution Program*, Springer-Verlag, Berlin, 1994.
- [29] A. Kolen, E. Pesch, Genetic local search in combinatorial optimization, *Discrete Applied Mathematics and Combinatorial Operations Research and Computer Science* 48 (1994) 273–284.
- [30] P. Merz, B. Freisleben, Genetic local search for the TSP: New results, in: *Proceedings of the 1997 IEEE International Conference on Evolutionary Computation*, IEEE Press, New York, 1997, pp. 159–164.
- [31] P. Merz, B. Freisleben, Fitness location landscape analysis and memetic algorithms for quadratic assignment problem, *IEEE Transaction on Evolutionary Computation* 4 (2000) 337–352.
- [32] A. Jaszkievicz, P. Kominek, Genetic local search with distance preserving recombination operator for a vehicle routing problem, *European Journal of Operational Research* 151 (2003) 352–364.
- [33] D.H. Wolpert, W.G. Macready, No free lunch theorem for optimization, *IEEE Transaction on Evolutionary Computation* 1 (1997) 67–82.
- [34] J. Drezo, A. Petrowsk, I.P. Siarry, E. Taillard, *Metaheuristics for Hard Optimization*, Springer, 2006.
- [35] N.J. Radcliffe, D.D. Surry, Formal memetic algorithms, in: T. Fogarty (Ed.), *Evolutionary Computing: AISB Workshop*, Springer-Verlag, 1994, pp. 1–16.
- [36] É.D. Taillard, Comparison of iterative searches for quadratic assignment problem, *Location Science* 3 (1995) 87–105.
- [37] M. Lundy, A. Mees, Convergence of an annealing algorithm, *Mathematical Programming* 34 (1986) 111–124.