Available online at www.sciencedirect.com
SCIENCE DIRECT®
Computers & Operations Research 32 (2005) 967–984
computers & operations research
www.elsevier.com/locate/dsw

Solving the uncapacitated hub location problem using genetic algorithms

H. Topcuoglu[a,*], F. Corut[a], M. Ermis[b], G. Yilmaz[a]

[a] *Computer Engineering Department, Marmara University, Goztepe Kampusu, 81040, Istanbul, Turkey*
[b] *Industrial Engineering Department, Air Force Academy, Istanbul, Turkey*

Abstract

Hub location problems are widely studied in the area of location theory, where they involve locating the hub facilities and designing the hub networks. In this paper, we present a new and robust solution based on a genetic search framework for the uncapacitated single allocation hub location problem (USAHLP). To present its effectiveness, we compare the solutions of our GA-based method with the best solutions presented in the literature by considering various problem sizes of the CAB data set and the AP data set. The experimental work demonstrates that even for larger problems the results of our method significantly surpass those of the related work with respect to both solution quality and the CPU time to obtain a solution. Specifically, the results from our method match the optimal solutions found in the literature for all test cases generated from the CAB data set with significantly less running time than the related work. For the AP data set, our solutions match the best solutions of the reference study with an average of 8 times less running time than the reference study. Its performance, robustness and substantially low computational effort justify the potential of our method for solving larger problem sizes.

Keywords: Hub location; Genetic algorithms; Heuristics

1. Introduction

In many transportation and telecommunication networks, the cost of carrying a unit of traffic between two points decreases as the capacity of the connection joining the two points increases. It is possible to facilitate this connection by building dedicated channels between each pair of nodes that communicate with each other. However, this would result in higher costs. Because of this fact, it is often convenient to design networks in which traffic is concentrated on high capacity links, even if

* Corresponding author.
E-mail address: haluk@eng.marmara.edu.tr (H. Topcuoglu).

0305-0548/$ - see front matter © 2003 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cor.2003.09.008

this traffic travels longer distances. In order to facilitate the flow of the traffic between nodes so as to decrease the overall cost of transportation, some centers known as hubs are introduced. Airline passenger flow, cargo or postal delivery networks, large telecommunication networks are examples of networks utilizing hubs.

There are different versions of the hub location problem [1]. There may be capacity restrictions on the volume of traffic a hub can collect, there may be a fixed cost associated with establishing any node as a hub and we may allocate nodes to one or more hubs [2]. However, in all of these variants, the objective function is to find the location of the hubs and the allocation of the nodes so that the total cost of the network is minimized. If all traffic flow via the hub nodes and each spoke (non-hub node) is allocated to a single hub node, this problem is specified as uncapacitated single allocation hub location problem (USAHLP). In this problem, the number of hubs is a decision variable and a fixed cost component is included in the formulation. If the number of hubs is fixed, say equal to p, this problem is referred to as an uncapacitated single allocation p-hub median problem (USApHMP).

The hub location problem was formulated as a quadratic integer program firstly by O'Kelly [1] and this problem is NP-hard even when the hub locations are fixed. The single allocation case has been studied by Aykin [3], Campbell [4,5], Klincewicz [6], O'Kelly et al. [7]. The proposed methods for USAHLP are mainly heuristics that do not guarantee optimal solutions, such as problem specific heuristics [1], tabu search [6], hybrid of genetic algorithms and tabu search [9], and neural networks [10].

Genetic Algorithms (GA) has been adapted to many operational research problems such as scheduling problems and the traveling salesman problem; however, there are very few applications to location-allocation problems. In this paper, we present a novel solution based on a genetic search framework for USAHLP. We compare the quality of solutions from our method with the solutions from GATS algorithm, which stands for "Genetic Algorithm and Tabu Search". The GATS algorithm is the most recent study for USAHLP and its results [9] match the best solutions found in the literature.

We evaluate the effectiveness of our method with both the CAB (Civil Aeronautics Board) data set and the AP (Australian Post) data set by considering the total transportation cost and the running time as the comparison metrics. The CAB data set [12] was originated by the CAB, and it has been the most popular data set used to benchmark algorithms for hub location problems. Based on the CAB data set, the results generated by our method match the best solutions found in the literature with an average of 8–14 times shorter running time than the related work. We also consider the AP data set [13] which was derived from a real-world location problem for AP. Similarly, our method obtains the best solutions found in the literature and it outperforms the related work with respect to the total cost of the network in 57% of the cases of AP data set with significantly less computational effort.

The remainder of this paper is organized as follows. In Section 2, we define the mathematical formulation of the problem that we address. Section 3 gives the details of our GA-based formulation, which is followed by the details of related work in Section 4. The experimental study with two data sets is given in Section 5. Section 6 presents conclusions and future work related to this problem.

2. Mathematical formulation

In this paper, we study the uncapacitated single allocation hub location problem (referred to as USAHLP), which is a location–allocation decision problem. A complete graph $G = (V, E)$ is given,

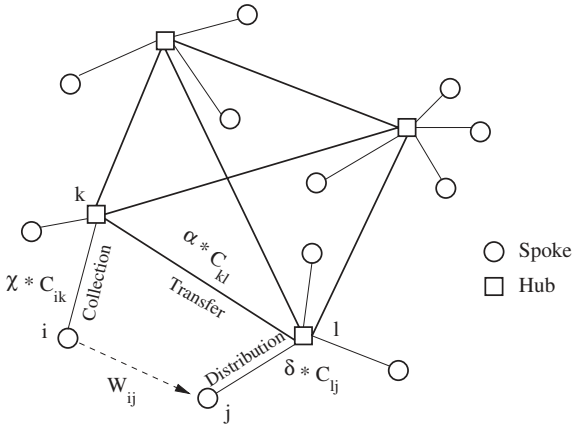

Fig. 1. An example hub network.

with a node set $V = 1, 2, \ldots, n$ where each node corresponds to origins, destinations and possible hub locations; and E is the set of edges between the nodes of the given network. The USAHLP was first formulated by O'Kelly [11] as a quadratic 0-1 optimization problem with linear constraints. The mixed integer linear programming (MILP) formulations were also developed [4,14]. Since the quadratic integer programming problem is well suited to mapping onto GA framework, we consider the O'Kelly's formulation.

Let N be the set of nodes, C_{ij} be the cost per unit flow from node i to node j, and W_{ij} be the amount of flow from node i to node j. For most of the applications, the cost C_{ij} is proportional to the distance between i and j, satisfying the triangle inequality. The path from an origin spoke i to destination spoke j includes three parts: collection from spoke i to the first hub k, transfer between the first hub k and the last hub l, and distribution from hub l to the destination spoke j. The cost per unit flow along this route ($i \to k \to l \to j$) is written as $\chi \times C_{ik} + \alpha \times C_{kl} + \delta \times C_{lj}$, where χ, α and δ are the collection, transfer and distribution costs respectively. In the literature, $\alpha < 1.0$ is considered as a discount factor to provide reduced unit costs between hubs, and $\chi = \delta = 1.0$. The postal delivery systems may have distinct values for these coefficients. The fixed cost of establishing a hub at node j is represented by f_j. An example of hub network with four hubs is given in Fig. 1.

The integer decision variable X_{ik} is 1 if node i is allocated to the hub located at node k, and it is equal to 0 otherwise. Each hub is assigned to itself; i.e., if node j is a hub, $X_{jj} = 1$. With these definitions, USAHLP is stated as follows:

$$\text{minimize} \quad f(x) = \sum_i \sum_j W_{ij} \sum_k \chi C_{ik} X_{ik} + \sum_i \sum_j W_{ji} \sum_l \delta C_{jl} X_{jl} + \sum_i \sum_k X_{ik} \sum_j \sum_l X_{jl} \alpha C_{kl} W_{ij} + \sum_j X_{jj} f_j \tag{1}$$

$$\text{subject to} \quad \sum_k X_{ik} = 1 \quad \forall i \in V, \tag{2}$$

$$X_{kk} - X_{ik} \geqslant 0 \quad \forall i,k \in V, \tag{3}$$

$$X_{ik} \in 0,1 \quad \forall i,k \in V. \tag{4}$$

The objective function given in Eq. (1) sums the costs of collection, transfer, distribution, and the costs of the hubs. The constraint given in Eq. (2) ensures that each node is assigned to exactly one hub. The inequality constraint given in Eq. (3) requires that node i is assigned to node k only if k is a hub. The last constraint requires that all variables be integers.

Considering symmetric costs simplifies the computations. Let O_i and D_i represent the total amount of flow originating and terminating at node i respectively, which are defined by

$$O_i = \sum_j W_{ij} \quad \text{and} \quad D_i = \sum_j W_{ji}. \tag{5}$$

Then, the objective function given in Eq. (1) can be rewritten as

minimize

$$f(x) = \sum_i \sum_k X_{ik} C_{ik} (\chi O_i + \delta D_i) + \sum_i \sum_k X_{ik} \sum_j \sum_l X_{jl} \alpha C_{kl} W_{ij} + \sum_j X_{jj} f_j. \tag{6}$$

The simplified version is considered since it computes the objective function more efficiently; and we use O_i and D_i terms for the initial population generation phase of our GA-based framework. On the other hand, we consider the original objective function for the AP data set, since the costs given in AP data set are not symmetric.

3. GA-formulation of the problem

A genetic algorithm (GA) is a search algorithm for finding the near-optimal solutions in large spaces, which is inspired from population genetics. The general idea was introduced by Holland [15] and genetic algorithms have been applied to a large set of problems in various fields in the literature. Two example sources for detailed information on GA are the books written by Goldberg [16] and Mitchell [17]. In this section, we present the details of our GA-based solution. The pseudocode of our method is presented in the appendix.

3.1. String representation

In our GA-based approach, each chromosome or bit string (i.e., an example solution) consists of two arrays: *HubArray* and *AssignArray*. The lengths of these arrays are equal to the number of nodes in the network. The *HubArray* includes 0's and 1's, where value 1 indicates that the node is a hub and value 0 indicates that it is a spoke. The *AssignArray* represents the assignments of the spokes to hubs. If node i is assigned to node k, its entry in the *AssignArray* has value k. Additionally, each hub is assigned to itself in the *AssignArray*.

3.2. Initial population generation

A predefined number of strings are generated for the initial population generation step. The first phase is to select the hubs; then, each spoke is assigned to the nearest hub based on distance values. The number of hubs is a decision variable for USAHLP. For 75% of the initial population, the number of hubs is set randomly from the range $[1,\ldots,n/4]$, where n is the total number of nodes. For the remaining 25% of the initial population, the number of hubs is set from the range $(n/4,\ldots,n/2]$. By this strategy, the total number of hubs in a network is up to half of the problem size.

After the number of hubs is set, the hubs are determined in the *HubArray*. By selecting the hubs, we consider the total amount of flow at each node i, which is equal to $O_i + D_i$. Here, O_i and D_i are the total amount of traffic originated by and destined for node i, respectively. Then, the nodes are sorted in a list, called *flow_list*, in decreasing order of total flows. For 75% of the initial population, the hubs in the *HubArray* are selected from two thirds of the nodes from the *flow_list* starting from the node with the highest total flow. This strategy provides the node which has higher amount of flow to become a hub with higher probability. For the remaining 25% of the population, hubs are selected from the whole set of nodes. These percentage values were determined with a set of experiments for specifying the control parameters of GA.

Specifically, if the population size is equal to 20 and the number of nodes is equal to 24, 15 individuals will have up to six hubs that are selected from the ones with the highest total flows; and the remaining five individuals will have up to 12 hubs that are selected from all nodes in the system. The number of generations is set to 200 in our experiments. Additionally, if the best fitness value does not change more than a predefined tolerance value, i.e., 0.0001 in our experiments, during the last 50 generations, it will be considered another stopping criteria.

3.3. Crossover operator

We consider fitness proportional selection with roulette wheel sampling in our GA-based framework. The fitness value of an individual is set with the reciprocal of the total cost of the individual given in Eq. (1). Elitism is applied in our experiments by placing the best chromosome at each generation in place of the worst individual with respect to total cost.

We apply single-point crossover operator on *HubArray* and *AssignArray* of the input strings by considering the same crossover point selected at random. The offsprings are generated by combining the left and right parts; which is followed by a phase for adjusting the offsprings, if necessary. After applying the crossover on *HubArrays*, if any of the offsprings does not have a single hub, or, if the number of hubs is equal to the number of nodes, then both of the offsprings will be discarded. For any offsprings of the *AssignArrays*, if a node i is assigned to another node which is not a hub anymore because of crossover on *HubArrays*, then node i is reassigned to the closest hub with respect to distance values.

In Fig. 2, the nodes 2 and 6 are hubs for S_1 and nodes 3 and 7 are hubs for S_2, initially. After applying the crossover on *HubArrays*, the hubs of the first offspring S_1^o are 2 and 7; and the hubs of the second offspring S_2^o are 3 and 6. After applying the crossover on *AssignArrays*, the assignments of nodes 1, 6 and 8 in S_1^* need rearrangement based on distance values, since node 3 and node 6 are not hubs any more in the *HubArray* S_1^0. A similar adjustment is done for the second *AssignArray* S_2^*.

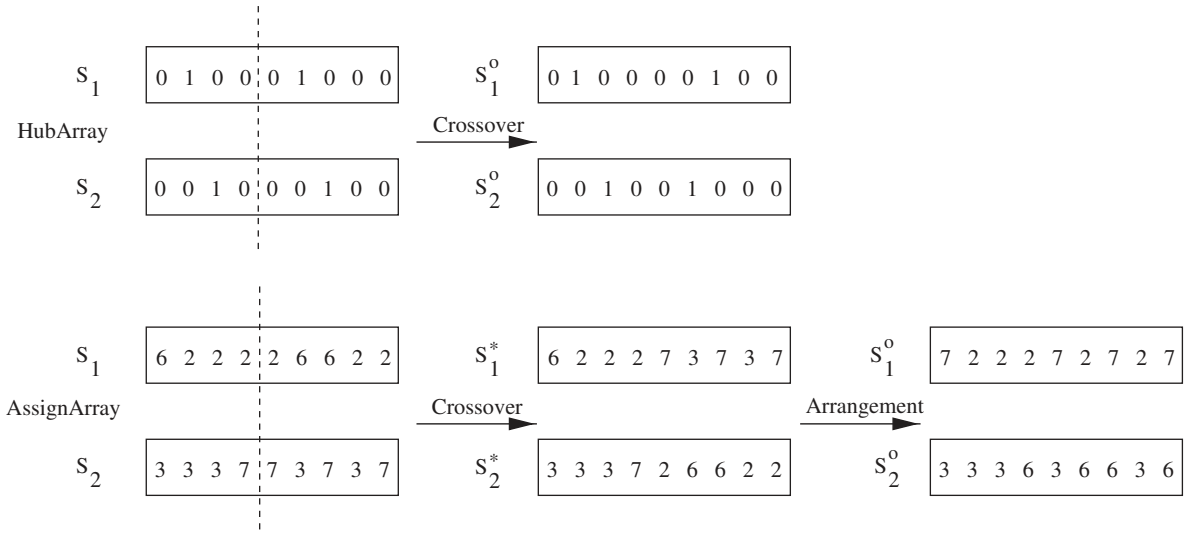

Fig. 2. An example use of the crossover operator.

3.4. Mutation operator

The mutation operator is applied only for the assignments of nodes. We consider two mutation operators, called *shift* and *exchange*. These are the extended versions of the moves given in [9], as part of its tabu search phase.

- *Shift*. This operator selects a spoke and reassigns it to another hub selected at random. If there is only one hub in the string, this mutation operator is not applied.
- *Exchange*. It selects two spokes at random and switches their assignments. Since we need at least two pairs of hubs and nodes, if there is only a single hub or only a single spoke, i.e., all other nodes are hubs in the string, then the exchange move is not applicable.

In our GA-based method, these two methods are applied for the selected individual, and the result with the better fitness value is selected with a predefined mutation rate.

3.5. Evolution of diversity

Population diversity has an important influence on the performance of genetic algorithms, and a fast decrease of diversity in the population guides to premature convergence. In order to study the evolution of diversity during the execution of a genetic algorithms, a measure for diversity should be presented.

We consider the hamming distance on *AssignArrays* for measuring diversity. If two *AssignArrays* S_1 and S_2 are given, the hamming distance between them is defined with the number of changes in the assignments required to transform S_1 into S_2. We define the diversity of a given population by the average distance between the individual with the best solution and all other individuals in the population. This method also measures the diversity between two solutions even if they use the same set of nodes as hubs.

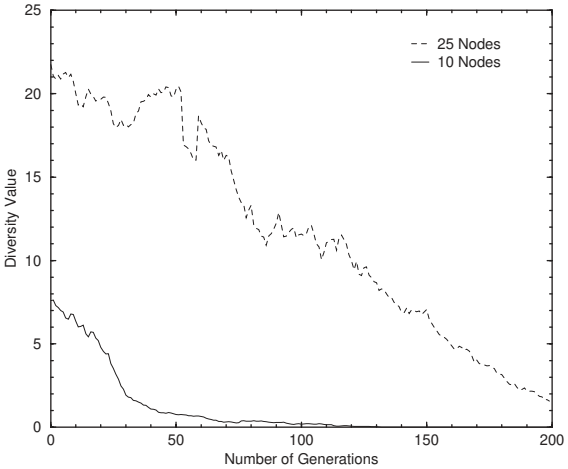

Fig. 3. Diversity values as number of generations is varied.

Fig. 3 gives diversity of the population for two problem sizes (10 and 25) of the CAB data set, which is explained in Section 5.1. For the problem size 10, initial population has a diversity value of 8, and for the problem size 25 it has a diversity value of 22. These diversity values indicate that our initial population phase generates quite distinct solutions for preventing premature convergence. The diversity values for the given two tests decrease with an increase in the number of generations.

4. Related work

The uncapacitated single allocation hub location problem (USAHLP) has received less attention in the literature than the uncapacitated single allocation p-hub median problem (USApHMP), which has been extensively studied [1,3–7,10]. The most recent study for USAHLP is the GATS algorithm [9], which stands for "genetic algorithm and tabu search". The results derived from the GATS algorithm matched the best solutions found in the literature so far. Therefore, we compared our GA-based method with GATS algorithm in our experimental study, with the two well-known data sets.

In the GATS algorithm, GA is used for selecting the number and the location of the hubs. A tabu search is used for assigning the spokes to the hubs. In the GA part, a possible solution is represented by a string of 0's for spokes, and 1's for hubs. In order to generate an initial population, hubs are located at random. It is followed by assigning the spokes to hubs based on distance values. The GA part includes a single-point crossover, and a mutation operator for changing a bit in the string from 0 to 1 or vice versa.

The tabu search part applies a shift move, or an exchange move, or both of them on the best solution found by the GA part, for a predefined number of iterations. If the result of tabu search improves the best cost so far, it is set as the best cost. The combination of GA and tabu search parts are repeated with new seed values. The stopping criteria for the GATS algorithm is three iterations in sequence yielding the same best solution.

5. Experimental study

In this section, we present the results of computational experiments to evaluate the effectiveness of our GA-based method. We compared the solutions from our GA-based framework with the solutions from the GATS algorithm, and the best solutions presented in the literature so far. The two comparison metrics are the total cost given in Eq. (1), and the running time to derive the solution.

We coded the three algorithms (our GA-based solution, the GATS algorithm and the simulated annealing heuristic explained in Section 5.2.1) in C programming language. The computational experiments of these algorithms were performed by the same machine, an Intel Pentium IV1.6 GHz. PC running the Linux operating system. The comparisons in this section are based on two different data sets: the CAB data set and AP data set.

5.1. CAB data set

In the literature, the CAB data set has been used frequently for evaluating the effectiveness of algorithms of hub location problems. The CAB data set [1] is based on airline passenger flow between 25 cities in 1970, where the distances between the cities satisfy triangular inequality, and flows between cities are symmetric.

Our experimental study on CAB data set is grouped into four problem sizes $n=\{10, 15, 20, 25\}$. For each problem size, we consider several experiments by varying the discount factor $\alpha \in \{0.2, 0.4, 0.6, 0.8, 1.0\}$, while keeping $\chi = \delta = 1$. For each discount factor, the fixed cost for establishing a node as a hub is varied as $f \in \{100, 150, 200, 250\}$. As a result, the experimental study includes a total of 80 different cases. For the CAB data set, the fixed cost of establishing a hub is considered to be equal for all nodes. Therefore, f was used in place of f_j, in this section.

A set of experiments were conducted to set the values of control parameters in our GA-based solution, which include the population size (P), number of generations (G), crossover probability (λ_C) and mutation probability (λ_M). The best quality of solutions from our GA method was observed when $P = G = 200$, $\lambda_C = 0.7$ and $\lambda_M = 0.4$. In our experimental study, the control parameters in the GATS algorithm were set with their values presented in [9].

Tables 1–4 give the results of our GA-method and the GATS algorithm for various problem sizes. In each table, the discount factor (α), the fixed cost of hubs (f), and the optimal costs are given. The optimal cost values of the USAHLP were derived by using the optimal costs of the USApHMP for various hub sizes, which are presented in [8]. Formally, it is measured by

$$Optimal\ \ Cost = \min_{p} \{OCost_p + p \times f\}, \tag{7}$$

where $OCost_p$ is the optimal cost value for the solution of USApHMP when the number of hubs is equal to p, and f is the cost of establishing a hub on a node.

The fourth columns of the tables present the corresponding set of hubs from the results generated by our GA-based framework. The GA.cost and GATS.cost columns are the total cost values of the solutions from the GA and GATS algorithms, respectively. It should be noted that the total costs for the GATS algorithm given in four tables match with their original results presented in [9]. They did not consider 16 problem cases when α is equal to 0.2, in their paper. The running time of algorithms were given in the GA.time, and the GATS.time columns, which are measured in CPU seconds.

Table 1
Results from GA and GATS algorithms when $n = 10$

α	f	Optimal					
		cost	GA.hubs	GA.cost	GATS.cost	GA.time	GATS.time
0.2	100	791.93	4,6,7	791.93	791.93	0.18	1.10
	150	915.99	7,9	915.99	915.99	0.18	2.18
	200	1015.99	7,9	1015.99	1015.99	0.17	1.31
	250	1115.99	7,9	1115.99	1115.99	0.16	1.05
0.4	100	867.91	4,6,7	867.91	867.91	0.18	1.10
	150	974.30	7,9	974.30	974.30	0.17	1.86
	200	1074.30	7,9	1074.30	1074.30	0.16	1.27
	250	1174.30	7,9	1174.30	1174.30	0.13	0.99
0.6	100	932.62	7,9	932.62	932.62	0.17	1.10
	150	1032.62	7,9	1032.62	1032.62	0.15	1.06
	200	1131.05	4	1131.05	1131.05	0.14	1.28
	250	1181.05	4	1181.05	1181.05	0.12	1.47
0.8	100	990.94	7,9	990.94	990.94	0.17	1.62
	150	1081.05	4	1081.05	1081.05	0.13	1.27
	200	1131.05	4	1131.05	1131.05	0.12	0.96
	250	1181.05	4	1181.05	1181.05	0.13	0.95
1.0	100	1031.05	4	1031.05	1031.05	0.14	2.15
	150	1081.05	4	1081.05	1081.05	0.13	1.00
	200	1131.05	4	1131.05	1131.05	0.14	0.99
	250	1181.05	4	1181.05	1181.05	0.13	0.92

The results from our algorithm matches the optimal solutions found in the literature for all combinations presented in the tables. For two cases out of 80, there were no optimal costs presented in the literature, and our algorithm provided the best results. The details of the cases are the followings:

- For the case of $n = 15$, discount factor 0.2, and fixed cost of 100, our algorithm and the GATS algorithm provide the total transportation cost of 1030.07 by selecting five hubs, which are 3,4,7,12, and 14 (see Table 2). However, the optimal solutions of USApHMP presented in the literature are for $p = \{2, 3, 4\}$. Among them, the best derived optimal solution of USAHLP is for the case of four hubs, which has a total cost of 1039.77. Since the five-hub case was not examined for USApHMP, this result is considered as a reference solution but not the optimal solution. Therefore the corresponding cell in the table is given with "***".
- A similar case is for the problem of 20 cities, when the discount factor is 0.2, and the fixed cost is equal to 100. The GA-based method chooses cities 4,7,12,14 and 17 as hubs by incurring a cost of 967.74. As in the previous case, the optimal cost was not presented.

Table 2
Results from GA and GATS algorithms when $n = 15$

α	f	Optimal					
		cost	GA.hubs	GA.cost	GATS.cost	GA.time	GATS.time
0.2	**100**	***	**3,4,7,12,14**	**1030.07**	**1030.07**	**0.36**	**5.54**
	150	1239.77	4,7,12,14	1239.77	1239.77	0.36	2.66
	200	1381.28	4,12	1381.28	1381.28	0.32	3.87
	250	1481.28	4,12	1481.28	1481.28	0.28	3.12
0.4	100	1179.71	4,7,12,14	1179.71	1179.71	0.35	4.15
	150	1355.09	4,7,12	1355.09	1355.09	0.33	2.68
	200	1462.62	4,12	1462.62	1462.62	0.32	2.59
	250	1556.66	4	1556.66	1556.66	0.30	2.49
0.6	100	1309.92	4,7,12	1309.92	1309.92	0.33	3.32
	150	**1443.97**	**4,12**	**1443.97**	**1456.66**	**0.30**	**8.52**
	200	1506.66	4	1506.66	1506.66	0.27	2.55
	250	1556.66	4	1556.66	1556.66	0.24	4.44
0.8	100	1390.76	4,11	1390.76	1390.76	0.32	4.02
	150	1456.66	4	1456.66	1456.66	0.26	3.33
	200	1506.66	4	1506.66	1506.66	0.24	4.33
	250	1556.66	4	1556.66	1556.66	0.26	2.47
1.0	100	1406.66	4	1406.66	1406.66	0.25	5.93
	150	1456.66	4	1456.66	1456.66	0.24	6.46
	200	1506.66	4	1506.66	1506.66	0.25	2.50
	250	1556.66	4	1556.66	1556.66	0.24	3.04

Additionally, the GATS algorithm cannot result in optimal cost values for two other cases, which are explained below:

- For the case of $n = 15$, the discount factor of 0.6 and the fixed cost of 150, the GATS algorithm chooses city 4 as the single hub, and it generates the total cost of 1456.66. However, our solution chooses cities 4 and 12 as hubs by incurring a total cost of 1443.97 (Table 2).
- For the case of $n = 25$, the discount factor of 1.0, and the fixed cost of 100, the hubs found by the GATS algorithm are cities 4 and 20 by incurring a cost of 1562.15. Our method outperforms GATS algorithm by providing a cost of 1559.19 with the selection of cities 7 and 19 as hubs.

Our GA-based method surpasses the related work with respect to efficiency, as well. It obtains optimal values in significantly less time than the GATS algorithm for all of the 80 test problems given. Specifically, when the problem size is 10, the average running time of GA method is 0.15. However, the GATS algorithm achieves the same optimum values by an average running time of 1.28, which is 8.5 times higher than our method. The difference is more significant for larger problem sizes. When $n=25$, the GATS algorithm requires almost 14 times more CPU time than our GA-based method. It is due to the fact that the GATS algorithm achieves the best solution using the tabu search phase by using shift moves, where the number of possible shift moves increases with an increase in problem size.

Table 3
Results from GA and GATS algorithms when $n = 20$

α	f	Optimal					
		cost	GA.hubs	GA.cost	GATS.cost	GA.time	GATS.time
0.2	**100**	***	**4,7,12,14,17**	**967.74**	**967.74**	**0.66**	**12.96**
	150	1174.53	4,12,17	1174.53	1174.53	0.58	7.09
	200	1324.53	4,12,17	1324.53	1324.53	0.70	9.54
	250	1474.53	4,12,17	1474.53	1474.53	0.67	9.49
0.4	100	1127.09	1,4,12,17	1127.09	1127.09	0.69	5.95
	150	1297.76	4,12,17	1297.76	1297.76	0.64	15.76
	200	1442.56	4,17	1442.56	1442.56	0.54	5.60
	250	1542.56	4,17	1542.56	1542.56	0.53	14.54
0.6	100	1269.15	1,4,12,17	1269.15	1269.15	0.73	8.70
	150	1406.04	4,17	1406.04	1406.04	0.74	5.35
	200	1506.04	4,17	1506.04	1506.04	0.70	5.44
	250	1570.91	6	1570.91	1570.91	0.57	5.25
0.8	100	1369.52	4,17	1369.52	1369.52	0.73	6.01
	150	1469.52	4,17	1469.52	1469.52	0.51	12.70
	200	1520.91	6	1520.91	1520.91	0.49	12.10
	250	1570.91	6	1570.91	1570.91	0.43	11.85
1.0	100	1410.07	4,20	1410.07	1410.07	0.70	10.22
	150	1470.91	6	1470.91	1470.91	0.53	6.86
	200	1520.91	6	1520.91	1520.91	0.43	9.60
	250	1570.91	6	1570.91	1570.91	0.53	10.63

5.2. AP data set

The AP data set was derived from the mail flows in an Australian city [13]. Since there are 200 nodes in the data set, it provides opportunities to study the real world problems of larger sizes. The problems of size $n = \{10, 20, 25, 40, 50, 100\}$ were obtained from the original data set. The values of the collection, transfer and distribution costs were set to $\chi = 3$, $\alpha = 0.75$ and $\delta = 2$, which holds $\chi > \delta > \alpha$. The flow volumes are not symmetric (i.e., $W_{ij} \neq W_{ji}$) and a mail can be sent from a postcode district to itself (i.e., $W_{ii} \neq 0$).

The AP data set includes different values of fixed costs to the different nodes in the network, which can be based on geographical or other factors. The two types of fixed costs given with the data set are the loose and the tight costs. For the tight version, the nodes with larger total flows are set with higher fixed costs, so that it will be harder to choose these nodes as hubs.

To the best of our knowledge, the AP data set was not used for USAHLP in the literature. The optimal cost values of USApHMP using AP data set were given for only a few problem size and hub size pairs [13]. The reference costs derived by adding optimal costs of the USApHMP to fixed costs, as in the CAB data set, may not provide tight bounds; since there are two types of fixed costs in the data set, and fixed costs vary among nodes. To provide tight reference costs, we developed an heuristic based on simulated annealing.

Table 4
Results from GA and GATS algorithms when $n = 25$

α	f	Optimal					
		cost	GA.hubs	GA.cost	GATS.cost	GA.time	GATS.time
0.2	100	1029.63	4,12,17,24	1029.63	1029.63	1.18	11.28
	150	1217.34	4,12,17	1217.34	1217.34	1.14	10.43
	200	1367.34	4,12,17	1367.34	1367.34	1.18	14.22
	250	1500.90	12,20	1500.90	1500.90	0.86	18.65
0.4	100	1187.51	1,4,12,17	1187.51	1187.51	0.97	19.63
	150	1351.69	4,12,18	1351.69	1351.69	1.02	16.57
	200	1501.62	12,20	1501.62	1501.62	1.25	23.87
	250	1601.62	12,20	1601.62	1601.62	0.86	10.15
0.6	100	1333.56	2,4,12	1333.56	1333.56	1.03	19.70
	150	1483.56	2,4,12	1483.56	1483.56	0.97	10.80
	200	1601.20	12,20	1601.20	1601.20	1.33	10.43
	250	1701.20	12,20	1701.20	1701.20	1.22	12.11
0.8	100	1458.83	2,4,12	1458.83	1458.83	0.92	14.27
	150	1594.08	12,20	1594.08	1594.08	1.14	11.31
	200	1690.57	5	1690.57	1690.57	0.74	15.74
	250	1740.57	5	1740.57	1740.57	0.70	10.44
1.0	**100**	**1559.19**	**7,19**	**1559.19**	**1562.15**	**0.87**	**14.11**
	150	1640.57	5	1640.57	1640.57	0.70	10.81
	200	1690.57	5	1690.57	1690.57	0.73	12.28
	250	1740.57	5	1740.57	1740.57	0.63	10.32

5.2.1. Reference study based on simulated annealing heuristic

In this section, we present problem-specific details and control parameters of the simulated annealing (SA) heuristic. Our SA implementation is a modified version of the one given in [13] for USApHMP. The first control parameter of the SA algorithm is the initial annealing temperature (T^0), which is calculated by considering the following formula [18]:

$$\gamma = \frac{m_1 + m_2 \times e^{-\Delta f/T^0}}{m_1 + m_2}. \tag{8}$$

Here, γ (the acceptance ratio) is set to 0.95 in our study. In this formula, m_1 represents the number of moves that have resulted in a decrease in the cost, m_2 represents the moves when the cost increased relative to the previous step, and Δf represents the average increase in cost for m_2 moves. The initial temperature is set with a temporary large value (100,000 in our experiments) for the Markov chain. After a full Markov chain is completed, the initial annealing temperature T^0 is derived using Eq. (8). We set the Markov chain length to $2np$, where n is the problem size and p is the hub size. We executed the simulated annealing heuristic by setting ($n = 50$, $p = 5$), which was resulted in assigning value of 224,000 for T^0.

A new term *cluster* is defined as the set of nodes that are allocated to the same hub. There are two main transitions provided for generating neighborhood solutions in the SA heuristic [13]: (a)

Table 5
Performance of algorithms for AP data set with loose fixed costs

n	SA algorithm		GA algorithm		GATS algorithm	
	Hubs	Cost	Hubs	Cost	Hubs	Cost
10	3,4,7	224249.82	3,4,7	224249.82	3,4,7	224249.82
20	7,14	234690.11	7,14	234690.11	7,14	234690.11
25	8,18	236649.69	8,18	236649.69	8,18	236649.69
40	14,28	240985.51	14,28	240985.51	14,19,29,35	266603.15
50	15,36	237420.69	15,36	237420.69	6,15,26,35,42,48	300226.47
100	29,73	238017.53	29,73	238017.53	No. of hubs = 21	695705.82
200	53,184	228944.77	53,184	228973.67	No. of hubs = 69	1967625.73

changing the allocation of a randomly chosen spoke to a different cluster, and (b) setting a randomly selected spoke as a hub in a randomly chosen cluster. One of these transitions is performed with respect to predefined transition probabilities. In order to escape the current local minimum during the execution of SA heuristic, a reheating mechanism is provided. We considered a cooling rate of 0.97 in our experiments.

5.2.2. *Results derived from the AP data set*

The GA and GATS algorithms were executed 100 times for each problem size using different seed values and the solution with the minimum cost was selected. In our experiments, the control parameters of the GA and the GATS algorithms were set with the values derived from the CAB data set. In order to provide the reference solution for each problem size n, the hub size is set from a range of $[1,\ldots,n/2]$, and the SA heuristic (our reference study) is executed 100 times with different set of random numbers for each hub size. The solution with the lowest cost value obtained becomes the reference solution. As an example, if the problem size is 40, the SA heuristic is executed for 20 cases by changing the hub size in the range of $[1,\ldots,20]$, which generates a total of 2000 solutions.

Tables 5 and 6 are the results of the experiments when the loose fixed costs are considered; and Tables 7 and 8 are the results for the tight fixed costs. Located "hubs" in solutions and "costs" of solutions for the three algorithms, the SA algorithm, the GA algorithm and the GATS algorithm, are presented in these tables. For the loose fixed costs, the results of our method match with the best solutions in the reference study for all problems sizes up to 200 nodes. Although the selected hubs from the SA algorithm and those from our algorithm are the same for the case of 200 nodes, the cost of our algorithm is slightly higher (by 0.012%) than the reference cost due to the different allocation of nodes 75 and 88 in the solutions.

As can be seen from Table 5, our algorithm substantially outperforms the GATS algorithm with an increase in problem size. Specifically, when $n=40$, the solution of the GATS algorithm has 11% more total cost than the solution of the GA algorithm. For larger problem sizes, the GATS algorithm uses a greater number of hubs which substantially increases its total cost of the network. When n is equal to 200, the GATS algorithm finds its best solution by locating 69 hubs that generates a total cost equal to 8.6 times that of the solution of our algorithm.

Table 6
Efficiency of algorithms with loose fixed costs, given in time (s) and SRun (Successful Run)

n	SA algorithm Time	GA algorithm		GATS algorithm	
		SRun	Time	SRun	Time
10	0.58	83	0.22	71	1.12
20	4.22	96	0.92	6	6.61
25	11.06	90	1.47	2	14.16
40	60.67	55	4.79	1	64.70
50	145.97	59	8.32	1	156.45
100	2116.78	2	54.23	1	1290.89
200	2550.67	1	439.08	1	> 2 days

Table 7
Performance of algorithms for AP data set with tight fixed costs

n	SA algorithm		GA algorithm		GATS algorithm	
	Hubs	Cost	Hubs	Cost	Hubs	Cost
10	4,5,10	263402.13	4,5,10	263402.13	4,5,10	263402.13
20	7,19	271128.41	7,19	271128.41	7,19	271128.41
25	13	295670.39	13	295670.39	13	295670.39
40	19	293163.38	19	293163.38	6,22,29	345386.77
50	24	300420.87	24	300420.87	1,5,24	411145.42
100	52	305101.07	52	305101.07	No. of hubs = 24	4369213.98
200	53,184	233537.93	53,184	233570.44	No. of hubs = 85	11911942.30

Table 8
Efficiency of algorithms with tight fixed costs, given in time (s) and SRun (Successful Run)

n	SA algorithm Time	GA algorithm		GATS algorithm	
		SRun	Time	SRun	Time
10	0.68	66	0.18	65	1.12
20	4.38	28	0.71	9	6.58
25	1.70	100	1.14	6	16.62
40	8.55	10	3.49	1	82.73
50	25.39	4	5.82	1	150.81
100	173.44	1	39.31	1	1344.77
200	3224.22	2	415.82	1	> 2 days

Our GA-based method is more efficient and it obtains the best results in significantly less CPU time (given in seconds) than both the related work and the reference study as well (Table 6). For the case of 10 nodes, the CPU time of the SA algorithm is equal to 2.63 times that of the GA algorithm. When we consider the overall time of the SA algorithm for finding the reference solution,

this gap becomes more significant; since the SA algorithm was actually tested with five different numbers of hubs in order to find the reference solution for $n = 10$. For the same problem size, the CPU time of the GATS algorithm is equal to 5 times that of our method.

Additionally, the two algorithms are compared with respect to *robustness* criteria, which is evaluated by *SRun*/*TotalRun*, where *SRun* (successful run) is the number of cases where the algorithm obtains its best result for the given problem size and *TotalRun* is the total number of runs. In our experiments, *TotalRun* is equal to 100. As can be seen from Table 6, the GA algorithm is substantially more robust than the GATS algorithm; and increasing the problem size decreases the robustness of the algorithms.

The same set of experiments were repeated with tight fixed costs (see Tables 7 and 8). As in the previous case, our algorithm significantly outperforms the related work with respect to total cost, CPU time and robustness measures. There is a considerable difference in hub costs among nodes for the case of tight fixed costs; and the GATS algorithm tends to use a greater number of hubs for larger problem sizes. Therefore, the performance gap between our method and the GATS algorithm for the case of tight fixed costs is much higher than the case of loose fixed costs.

6. Conclusions

In this paper, we applied the principles of genetic algorithms to solve the hub location problem. Our objective was to exploit the features of genetic algorithms in finding the number of hubs, the location of hubs, and the assignment of spokes to the hubs. A computational study based on several instances derived from CAB data set and AP data set was carried out to test the performance of our GA-based framework and the related work. The results clearly demonstrate that even for large problems our approach significantly surpasses the related work with respect to both solution quality given in total transportation cost and computational time. Additionally, our genetic search approach is good at diversifying the search and it is significantly more robust than the related work. A planned future work is the parallel implementation of our genetic search framework for solving syntactically generated, large-scale data sets.

Appendix.

In this section, we present a pseudo code of our GA-based method. We first present the outline of the main program, which is followed by the three procedures that were referred to in the program. Here, n is the problem size, and p is the number of hubs. The crossover and mutation probabilities is represented with P_C and P_M, respectively. The details can be found in Section 3.

GA_based_Heuristic(n)
Procedure *Initial_Population*(n)
while ((convergence is not achieved) **and**
($Number\_of\_Generation \leqslant GenerationSize$)) **do**
for $i = 1$ **to** PopulationSize/2 **do**
Randomly select two chromosomes: S_1, S_2

Procedure Crossover(S_1, S_2, S_1^*, S_2^*)
Procedure Mutation(S_1^*)
Procedure Mutation(S_2^*)
Insert offsprings into the new generation
endfor
Apply *elitism* for the new generation
endwhile
Report the best chromosome as the final solution

Procedure Initial_Population(n)
Compute the total amount flow of each node n_k, which is
equal to $Flow_k = O_k + D_k$.
Sort the nodes in decreasing order of $Flow_k$ values in a list, *flow_list*.
$Hub\_Set = \{n_k \mid n_k = flow\_list[i]\ \forall i,\ 1 \leqslant i \leqslant \frac{2}{3} \times n\}$.
for $i = 1$ **to** $(\frac{3}{4} \times PopulationSize)$ **do**
$p \leftarrow random[1..\frac{n}{4}]$
Select p hubs from *Hub_Set*, randomly.
Assign each spoke to the nearest hub based on distance values.
endfor
for $i = (\frac{3}{4} \times PopulationSize) + 1$ **to** $PopulationSize$ **do**
$p \leftarrow random(\frac{n}{4}..\frac{n}{2}]$
Select p hubs from all nodes randomly.
Assign each spoke to the nearest hub based on distance values.
endfor

Procedure Crossover(S_a, S_b, S_a^*, S_b^*)
If $random(0..1) \leqslant P_C$ **then**
Select crossover point randomly.
Apply single-point crossover on HubArray and AssignArray of S_a and S_b
to produce S_a^* and S_b^*.
If $(p = 0$ **or** $p = n)$ in (S_a^*, S_b^*) **then**
Discard S_a and S_b and repeat the selection step.
For each spoke n_k in (S_a^*, S_b^*) assigned to non-hub **do**
Re-assign n_k to the nearest hub in the corresponding offspring.
endif
else $S_a^* = S_a$ and $S_b^* = S_b$.

Procedure Mutation(S_c)
If $random(0..1) \leqslant P_M$ **then**
Select a spoke n_k at random.
Assign n_k to another hub selected at random.
Call this solution S_c^s. // result of shift move
Select two spokes n_x, n_y (that are assigned to different hubs) at random.

Switch the assignments of n_x and n_y.
Call this solution S_c^e. // result of exchange move
If (*shift* or *exchange* not applicable) **then**
Return S_c.
else
Return best of (S_c^s, S_c^e) with respect to fitness values.
endif
else Return S_c.

References

[1] O'Kelly M. A quadratic integer program for the location of interacting Hub facilities. European Journal of Operational Research 1987;32:393–404.
[2] Drezner Z, Hamacher HW. Facility location: applications and theory. Berlin: Springer; 2002.
[3] Aykin T. Lagrangian relaxation based approaches Hub-and-Spoke network design problem. European Journal of Operational Research 1994;79:501–23.
[4] Campbell JF. Integer programming formulations of discrete Hub location problems. European Journal of Operational Research 1994;72:387–405.
[5] Campbell JF. Hub location and the p-Hub median problem. Operations Research 1996;44:923–35.
[6] Klincewicz JG. Heuristics for the p-Hub location problem. European Journal of Operational Research 1991;79: 25–37.
[7] O'Kelly M, Skorin-Kapov D, Skorin-Kapov J. Lower bounds for the Hub location problem. Management Science 1995;41:713–21.
[8] Skorin-Kapov D, Skorin-Kapov J, O'Kelly M. Tight linear programming relaxations of uncapacitated P-Hub median problems. European Journal of Operational Research 1996;94:582–93.
[9] Sue Abdinnour-Helm. A hybrid heuristic for the uncapacitated Hub location problem. European Journal of Operational Research 1998;106:489–99.
[10] Smith KA, Krishnamoorthy M, Palaniswami M. Neural versus traditional approaches to the location of interacting Hub facilities. Location Science 1996;4:155–71.
[11] O'Kelly M. Hub facility location with fixed costs. The Journal of the Regional Science Association International 1992;71:293–306.
[12] Beasley JE. OR-library: distributing test problems by electronic mail. Journal of the Operational Research 1990;41:1069–72.
[13] Ernst AT, Krishnamoorthy M. Efficient algorithms for the uncapacitated single allocation p-Hub median problem. Location Science 1996;4:139–54.
[14] Ebery J. Solving large single allocation p-Hub problems with two or three Hubs. European Journal of Operational Research 2001;128:447–58.
[15] Holland JH. Adaptation in natural and artificial systems. Michigan: University of Michigan Press; 1975.
[16] Goldberg DE. Genetic algorithms in search, optimization and machine learning. Reading, MA: Addison-Wesley; 1989.
[17] Mitchell M. An introduction to genetic algorithms. Cambridge, MA: MIT Press; 1998.
[18] Aarts EHL, Korst J. Simulated annealing and Boltzmann machine. NY: Wiley; 1989.

Haluk Topcuoglu is an Assistant Professor in the Computer Engineering Department at Marmara University, Istanbul, Turkey. He received the B.S. and M.S. degrees in Computer Engineering from Bogazici University, Istanbul. He received the Ph.D. degree in Computer Science from Syracuse University in 1999. His research interests are focused on task scheduling in parallel and distributed systems, evolutionary computation, parallel programming techniques and applications.

Fatma Corut received her B.S. degree in Computer Engineering Department at Marmara University in 2002. She is currently graduate student and research assistant in the same department. Her research interests include genetic algorithms and parallel programming.

Murat Ermis is an instructor in industrial Engineering Department at Turkish Air Force Academy. He received his M.S. degree and PhD in industrial Engineering from Middle East Technical University and Istanbul Technical University respectively. His main research interests are focused on mathematical modeling, metaheuristics and scheduling. He has published in international conferences and journals related with these areas.

Gulsah Yilmaz received her B.S. degree in Computer Engineering Department at Marmara University in 2002. She is currently a graduate student in the same department.