Control and Cybernetics

vol. **37** (2008) No. 3

An efficient genetic algorithm for the uncapacitated multiple allocation p-hub median problem*

by

Zorica Stanimirović

Faculty of Mathematics, University of Belgrade
Studentski trg 16/IV, 11 000 Belgrade, Serbia
e-mail: zoricast@matf.bg.ac.yu, zoricast@mi.sanu.ac.yu

Abstract: In this paper the Uncapacitated Multiple Allocation p-hub Median Problem (the UMApHMP) is considered. A new heuristic method based on a genetic algorithm approach (GA) for solving UMApHMP is proposed. The described GA uses binary representation of the solutions. Genetic operators which keep the feasibility of individuals in the population are designed and implemented. The mutation operator with frozen bits is used to increase the diversibility of the genetic material. The running time of the GA is improved by caching technique. Proposed GA approach is benchmarked on the well known CAB and AP data sets and compared with the existing methods for solving the UMApHMP. Computational results show that the GA quickly reaches all previously known optimal solutions, and also gives results on large scale AP instances (up to n=200, p=20) that were not considered in the literature so far.

Keywords: p-hub problem, genetic algorithms, discrete location and assignment

1. Introduction

Hub networks are widely used in modern transport and telecommunication systems. Instead of serving each user from its assigned facility with a direct link, hub networks route the flow via established hub network. Hubs serve as consolidation and connection points between two locations. There is economy of scale incorporated by a discount factor for transportation between the hubs and no direct transportation between two non-hub nodes is allowed. By employing hub nodes as switching points in the network, and by increasing transportation between them, capacity network can be used more efficiently.

The hub location problem is concerned with locating hub facilities and allocating non-hub nodes to hubs. Depending on how the non-hub nodes are allocated to the hubs, there are two basic allocation schemes in the hub network: single allocation and multiple allocation scheme.

*Submitted: January 2006; Accepted: September 2008.

1) In the single allocation scheme each origin/destination node must be assigned to exactly one hub. All of the flow from/to each non-hub node is transported only via specified hub.
2) Multiple allocation scheme allows each non-hub node to communicate with more than one hub.

The difference between single and multiple allocation schemes is illustrated in Fig.1. In a network with $n = 5$ nodes, given by their (x, y) coordinates in the plane, $p = 2$ hub nodes are to be located in order to minimize the overall transportation costs. The corresponding distance matrix is:

$$D = \begin{pmatrix} 0 & 4 & \sqrt{26} & 3 & \sqrt{26} \\ 4 & 0 & \sqrt{2} & 5 & \sqrt{34} \\ \sqrt{26} & \sqrt{2} & 0 & \sqrt{29} & 4\sqrt{2} \\ 3 & 5 & \sqrt{29} & 0 & \sqrt{5} \\ \sqrt{26} & \sqrt{34} & 4\sqrt{2} & \sqrt{5} & 0 \end{pmatrix}.$$

The unit rates for transportation origin-hub, hub-hub and hub-destination are equal to 1, 0.75 and 1 respectively. The optimal solutions in single/multiple allocation case are presented in the middle/on the right side of Fig.1. As can be seen from the diagrams, the optimal solutions differ significantly, not only in the allocations but also in the hub locations.

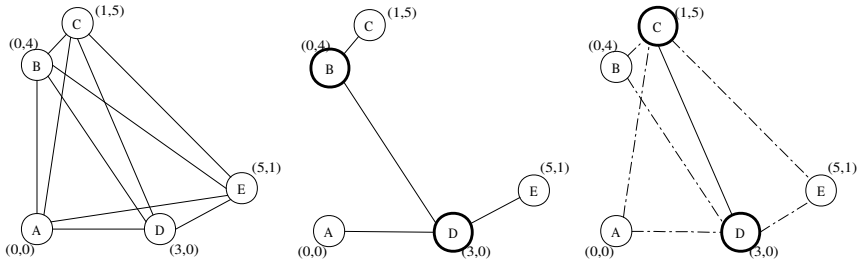

Figure 1. The optimal solution to a single and a multiple allocation p-hub problem in a network with $n = 5$, $p = 2$, $\chi = \delta = 1$, $\alpha = 0.75$

In the case of single allocation scheme, hubs are located at nodes **B** and **D**, while non-hub nodes A and E are allocated to (exactly one) hub **D** and non-hub node C is allocated to (exactly one) hub **B**. In the multiple allocation case, nodes **C** and **D** are chosen to be hubs, while non-hub nodes are allowed to communicate via more than one hub: non-hub nodes A, B and E may communicate via hub **C** or **D**, depending on the transportation cost. For example, for the transportation from origin node A to destination node E via hub **C**, transportation costs along the path "A-**C**-E" are $\sqrt{2} + \sqrt{34} = 7.25$. If we choose to transport via hub **D**, the transportation costs for the path "A-**D**-E" are lower $3 + \sqrt{5} = 5.23$. However, for the transportation from origin-node A to destination-node B we

choose the hub **C**, because the cost "A-**C**-B" $\sqrt{26}+\sqrt{2}=6.51$ is lower that the costs "A-**D**-B" $3+5=8$ and "A-**D**-**C**-B" $3+0.75\cdot\sqrt{29}+\sqrt{2}=8.453$. The minimum overall transportation costs in the single/multiple allocation cases are equal to 62.402 and 58.566 respectively. The reduction in total cost in the example given in Fig.1 by allowing multiple allocation is around 6.15% .

Hub location models may involve capacity restrictions on the hubs, fixed costs on both hub and non-hub nodes, predetermined number of hubs etc. If the number of hubs to be located is fixed to p, we are dealing with p-hub location problems. Capacitated versions of the hub location problems are also considered, but the nature of capacities may be different. The transport between hubs or between hub and non-hub nodes can be limited. There are also variants of capacitated hub problems that consider limits on the flow into/through each hub node.

Many variants of the hub location problems have been studied in the literature, due to their important application in practice. Typical applications of hub location problems are: telecommunication systems, postal and other delivery networks, airline passenger travel, cargo delivery, computer networks, etc. Reviews of hub location problems and their classification can be found in Campbell et al. (2002), Campbell (1996).

Most of hub location research has been devoted to hub median problems, in which the main goal is to design a network with hubs in order to minimize the total transportation cost and possibly the costs of establishing such a network. However, the p-hub median formulation can sometimes lead to unsatisfactory results, for example, when the worst origin-destination distance (cost) is important. This may happen, for example, in designing fast delivery systems, where the upper bound on the delivery time has to be observed.

Difficulties of this kind can be avoided by using the p-hub center formulation, which was first introduced and discussed by Campbell (1994). He defined three types of p-hub center problems according to different objectives: minimization of the maximum cost for any origin-destination pair (important in transporting perishable or time-sensitive items), minimization of the maximum transportation cost between any pair of nodes (important in hub networks with certain limitations on the arcs), and minimization of the maximum transportation cost between a hub and non-hub node (important in hub networks with some special attributes of hub-hub links and/or limitations on hub-origin/destination links).

In the literature, there are more complex and realistic hub location models that arise from practice. Instead of having constant discount factor for all hub-hub flows, there are models with flow-based discounts (Bryan, 1998, and O'Kelly, 1998) which have shown to be appropriate for air freight in order to make allocation decisions based on the ability of flows to capture scale economies. O'Kelly and Bryan (1998) proposed a non-linear cost-function which allows cost to increase at a decreasing rate as flows increase.

In Bryan and O'Kelly (1999) the authors considered the flow capacities and minimum flows on inter-hub links and flow-dependent costs in all network links.

Non-linear cost-functions are also proposed in Horner and O'Kelly (2001), Wagner (2004b) and Kimms (2005).

There are also hub models that include hub arcs (arcs with discounted cost rates), Nickel et al. (2001), Campbell et al. (2003, 2005a, b). The objective in these models is to locate a fixed number of hub arcs in order to minimize the overall cost. In Podnar et al. (2002) the authors considered a flow threshold model where they do not locate hubs but they decide on the links with reduced unit transportation costs. In their model, the cost of flow is reduced according to a prescribed discount factor, if the flow through that link is larger than a given threshold value. The variants of flow threshold model are also considered in Aykin (1995), Podnar et al. (2002), Podnar and Skorin-Kapov (2003), Skorin-Kapov (2001, 2005). This model encourages the concentration of flows and use of a relatively small number of links, reflecting actual characteristics of the network (that is important for designing high-speed telecommunication networks, urban public transportation networks,..).

Another type of hub models, called the latest arrival hub location problems, are introduced by Kara and Tansel (1999). They observed that the time spent at hubs (for sorting, loading and unloading the flow) may be significant comparing to total transportation time. The solution was to impose a maximum travel time constraint and then to minimize the cost of setting up such a network (the number of hubs required), resulting in various hub covering problems. These models have significant application in designing cargo delivery systems (Kara and Tansel, 1999, 2003; Wagner, 2004a; Ernst et al., 2005, and Yaman, 2005).

Hub location models with competition Marianov et al. (1999) are suitable for both air passenger and cargo transportation. In these models, the customer capture from competitor hubs is sought, which happens whenever the location of a new hub results in the reduction of time or distance needed from the traffic generated by the passenger to travel from the origin to the destination node.

2. Mathematical formulation

In this paper the Uncapacitated Multiple Allocation p-hub Median Problem (UMApHMP) is studied. Campbell was the first to formulate the UMApHMP as a linear integer program in Campbell (1992). Several improvements of this formulation arise in the literature: Skorin-Kapov et al. (1996), Ernst (1998a) and Boland et al. (2004). The mixed integer linear programming formulation proposed in Boland et al. (2004) is used in this paper.

Consider a set $I = 1, ..., n$ of n distinct nodes in the network, where each node refers to origin/destination or potential hub location. The distance from node i to node j is C_{ij}, and the triangle inequality may be assumed (Campbell et al., 2002). The demand from an origin i to a destination j is denoted with W_{ij}. The number of hubs to be located is fixed to p. Each path from an origin to destination node consists of three components: transfer from an origin to the first hub, transfer between the hubs and finally distribution from the last

hub to a destination. Parameters χ and δ denote unit costs for collection and distribution, while α represents a discount factor for the transportation between hubs. Decision variables H_j, Z_{ik}, ${Y_{kl}}^i$ and ${X_{lj}}^i$ are used in the formulation as follows:

$H_j = 1$, if a hub is located at node j, 0 otherwise
Z_{ik} = the amount of flow from node i that is collected at hub k
${Y_{kl}}^i$ = the amount of flow from node i that is collected at hub k, and transported via hub l
${X_{lj}}^i$ = the amount of flow from node i to destination j that is distributed via hub l.

The UMApHMP assumes the multiple allocation scheme, which allows each non-hub node to be allocated to more than one hub. The objective is to locate exactly p hub facilities, such that the total flow cost is minimized. Using the notation mentioned above, the problem can be written as:

$$\min \sum_i [\chi \sum_k C_{ik} Z_{ik} + \alpha \sum_k \sum_l C_{kl} {Y_{kl}}^i + \delta \sum_l \sum_j C_{lj} {X_{lj}}^i] \tag{1}$$

subject to:

$$\sum_i H_i = p \tag{2}$$

$$\sum_k Z_{ik} = \sum_j W_{ij} \quad \text{for every } i \tag{3}$$

$$\sum_l X_{lj}^i = W_{ij} \quad \text{for every } i, j \tag{4}$$

$$\sum_l Y_{kl}^i + \sum_j X_{kj}^i - \sum_l Y_{lk}^i - Z_{ik} = 0 \quad \text{for every } i, k \tag{5}$$

$$Z_{ik} \leq \sum_j W_{ij} H_k \quad \text{for every } i, k \tag{6}$$

$$\sum_i X_{lj}^i \leq \sum_i W_{ij} H_l \quad \text{for every } l, j \tag{7}$$

$$X_{lj}^i, Y_{kl}^i, Z_{ik} \geq 0, H_k \in \{0, 1\} \quad \text{for every } i, j, k, l. \tag{8}$$

The objective function (1) minimizes the sum of the origin-hub, hub-hub and hub-destination flow costs multiplied with parameters χ, α and δ respectively. Constraint (2) limits the number of located hubs to p, while (3)-(5) represent the divergence equations for the network flow problem for each node i. Constraints (6) and (7) prevent direct communication between non-hub nodes, while (8) reflects non-negative and/or binary representation of decision variables.

The UMApHMP is known to be NP-hard, with exception of some special cases (for example when matrix of flows W_{ik} is sparse) that are solvable in polynomial time. If the set of hubs is fixed, the problem can also be polynomially solved using the shortest-path algorithm in $\mathrm{O}(n^2\mathrm{p})$ time.

2.1. Previous work

Due to the number of important applications in practice, hub location problems have received a lot attention in the past decade. However, most of the work has concentrated on the single allocation case. A recent survey of various algorithms that have been applied to hub location problems up to now can be found in Alumur and Kara (2008).

Several approaches for solving the the UMApHMP have been proposed so far. Campbell (1996) developed a greedy exchange heuristic for this problem. Aykin (1995) described an enumeration and greedy interchange method for the UMApHMP and its variants. Ernst and Krishnamoorthy (1998a) proposed an exact LP based Branch-and-Bound method (BnB) and two heuristic methods for the UMApHMP: a shortest-path based heuristic and an explicit enumeration heuristic. The authors first enumerate all possible hub locations for the UMApHMP. Once the hub locations are fixed, the allocation of non-hub nodes is determined by using shortest-paths via the located hubs. This algorithm is exponential in p, polynomial in n, and considering the number of hubs in problem instances to be relatively small ($p \leq 5$), this approach gives exact solutions in reasonable computing time. For larger instances, the shortest path method can be combined with a Branch-and-Bound algorithm of Ernst and Krishnamoorthy (1998b). The shortest path problems are solved to obtain lower bounds that are used in a Branch-and-Bound scheme to obtain exact solutions. In Ernst and Krishnamoorthy (1998b) the authors presented computational results of the shortest path method only for CAB ($n \leq 25$, $p \leq 4$) and smaller size AP ($n \leq 50$, $p \leq 5$) instances. Hybridization with BnB gave results on some larger AP instances ($n = 100$, $p \leq 5$ and $n = 200$, $p = 2, 3$).

Boland et al. (2004) developed an exact Branch-and-Bound method for solving multiple allocation hub location problems using pre-processing and cutting algorithms. They first obtain good upper bounds that are used to cut the size of branch and bound tree, but this approach gives results only on CAB ($n \leq 25$, $p \leq 4$) and smaller size AP instances ($n \leq 50$, $p \leq 5$).

A special case of the UMApHMP, called 1-stop multiple allocation p-hub median problem is considered by Sasaki, Suzuki and Drezner (1999). In this problem, each origin-destination path is allowed to use only one hub. The authors proposed a mixed integer formulation of the problem and two algorithms for solving it: an exact BnB algorithm and a greedy-based heuristic method. The proposed methods were tested on the standard CAB data set with up to $n = 25$ nodes.

Many papers in the literature deal with single allocation variant of the p-hub median problem-USApHMP: Campbell (1994), Skorin-Kapov et al. (1996), Ernst and Krishnamoorthy (1996) and Ebery (2001). The USApHMP is also NP-hard, even if the hub locations are fixed Kara (1999). Obviously, the solutions to the UMApHMP represent a lower bound for the optimal solution to the USApHMP. This fact was used in Campbell (1996) to develop two heuris-

tic methods, which derive solution to the USApHMP from the solution to the UMApHMP. Various heuristic methods for solving the USApHMP have been proposed up to now: Exchange heuristic (Klincewicz, 1991), tabu-search heuristic (Klincewicz, 1992), Skorin-Kapov (1994), lower bounding method (O'Kelly et al., 1995), simulated annealing heuristic (Abadinour-Helm, 1998; Ernst and Krishnamoorthy 1996), etc.

Exact solution methods, as well as their hybridizations with heuristics for solving the USApHMP and its variants are described in Sohn and Park (1997, 2000), Ernst and Krishnamoorthy (1998b), Pirkul and Schilling (1998), Ebery (2001), Elhedhli and Hu (2005) etc.

3. Genetic algorithm

GA is a problem-solving metaheuristic based on the concept of natural evolution. The main idea was introduced by Holland (1975), and in past three decades the development of the GA theory and its applications are rapidly growing.

The GA approach uses the analogies between the individuals in the nature and problem solutions. The main structure that GA is working with is a population of individuals. Each individual is encoded as a string of characters from some alphabet, and it corresponds to one solution in the search space. For each individual the fitness value is computed. It carries the information about the solution quality, and it is not necessarily equal to the objective function. From generation to generation the GA tries to produce the improvement of quality of every solution, as well as better average fitness of the whole population. It is obtained by using genetic operators: selection, crossover and mutation. For more information about GA see Bäck et al. (2000a, b).

The basic scheme of the GA can be represented as:

```
Input_Data(); Population_Init(); while not Finish() do
    for i:=1 to Npop do
        pi := Objective_Function(i);
    endfor
    Fitness_Function();
    Selection();
    Crossover();
    Mutation();
endwhile Output_Data();

/* Npop denotes the number of individuals in a population and pi is
objective value of i-th individual */
```

Very successful GA applications to some NP-hard problems are given in Kratica (2000), Kratica et al. (2001), Ljubić (2004) and Raidl and Ljubić (2002). Genetic algorithms can also be combined with exact Branch-and-Cut-and-Price method (Ljubić, 2004).

GA approaches for solving some other hub location problems of smaller dimensions are described in Abadinour-Helm (1998, 2001), Abadinour-Helm and and Venkataramanan (1998) and Topcuoglu et al. (2005). Unfortunately, they apply simple GA with roulette-wheel selection, one-point crossover and simple mutation. In the literature, GAs are also sucessfully applied to different large-scaled hub location problems: the Uncapacitated Single Allocation Hub Location Problem-USAHLP in Abadinour-Helm and Venkataramanan (1998), the Uncapacitated Multiple Allocation p-Hub Center Problem-UMApHCP in Kratica and Stanimirovi (2006), the Uncapacitated Single Allocation p-Hub Median problem in Kratica et al. (2006). Although these problems are similar by names, evolutionary based approaches proposed up to now for solving these problems have quite different characteristics. For example, different allocation schemes in the UMApHMP and the USApHMP have great impact on the problem complexity. For the fixed set of hubs, the multiple allocation sub-problem is solved in polynomial $O(n^2p)$ time, while the single allocation sub-problem remains NP-hard. Therefore, genetic algorithms proposed up to now for solving other hub location problems, can not be applied to the UMApHMP. Therefore, a new GA approach is designed and described in the next section.

4. Proposed genetic algorithm

4.1. Representation and objective function

The binary encoding of the individuals is used in this GA implementation. Each solution is represented by a binary string of length n. Gene 1 in the genetic code denotes that particular hub is established, while gene 0 shows that it is not. Since users can be assigned only to opened hub facilities, only array (H_j) is obtained from the genetic code and the values of Z_{ik}, Y^i_{kl} and X^i_{lj} are calculated during the evaluation of the objective function.

For fixed set of hubs (H_j), the modified version of the well-known Floyd-Warsall shortest path algorithm (Ahuja et al., 1993; Ernst and Krishnamoorthy, 1998a), is used. After finding shortest paths between all pair of nodes, it is simple to evaluate objective function only by summing the shortest distances origin-hub, hub-hub and hub-destination, multiplied with flows and corresponding cost parameters χ, α and δ.

4.2. Genetic operators

Selection The GA implementation uses the fine-grained tournament selection (FGTS), proposed in Filipović (1998), that is an improvement of the standard tournament selection operator. Instead of integer parameter N_{tour} - the size of tournament group, the FGTS depends on real parameter F_{tour} - desired average tournament size. The FGTS operator uses two types of tournaments. The first type is held k_1 times and its size is $[F_{tour}]+1$. The second type is performed k_2 times with $[F_{tour}]$ individuals participating. Since the value $F_{tour} = 5.4$ is

used in this FGTS implementation, the corresponding values k_1 and k_2 (for 50 non-elitist individuals) are 20 and 30 respectively. The running time of FGTS operator is O(n * F_{tour}). In practice, F_{tour} is considered to be constant (not depending on n), that gives $O(n)$ time complexity. For detailed information about the FGTS operator see Filipović (1998 and 2006).

Crossover After a pair of parents is selected, a crossover operator is applied to them, producing two offspring. The basic crossover exchanges segments of two parents' genetic codes after the crossover point that is randomly chosen. A simple exchange of the two segments may produce incorrect offspring for the UMApHMP (the number of ones in the code may become different from p), although the parents had exactly p ones in their genetic codes. To overcome this problem, the basic crossover is modified. Modified crossover operator is simultaneously tracing genetic codes of the parents from right to left, searching for the position i on which the first parent has 1 and second 0. The individuals exchange genes on the found position (identified as crossover point), and similar process is performed starting from the left side of genetic code. Operator is searching the position j where the first parent has 0 and the second 1. Genes are exchanged on the j-th position, and the number of located hubs in both individuals is unchanged. Described process is repeated until $j \geq i$ (see Fig.2).

```
parent1: 001100110101  --->  001100110101  --->
parent2: 011110100001        011110100001
                               ->j i<-

011100110001  --->  011100110001 --->  011101100001 offspring1
001101000101        001101000101       001100010101 offspring2
 j       i             ->j i<-              j i
```

Figure 2. Modified crossover operator

The crossover is performed with the rate $p_{cross} = 0.85$. It means that around 85% pairs of individuals take part in producing offspring.

Mutation Offspring generated by the crossover operator are subject to mutation with frozen bits. Mutation operator is performed by changing a randomly selected gene in the genetic code (0 to 1, 1 to 0), with basic mutation rate of 0.4/n for non-frozen bits and 1.0/n for frozen bits. These mutation rates are constant through GA generations. In each individual the numbers of mutated ones and zeros are counted and compared. In case these numbers are not equal, it is necessary to mutate additional genes in order to equalize them. In this way mutation operator preserves p ones in the genetic code and keeps the mutated individual feasible.

During the GA execution it may happen that (almost) all individuals in the population have the same gene on a certain position, as it can be seen from

```
gen.code 1: 0110010110
gen.code 2: 1100010110
gen.code 3: 0111000110
gen.code 4: 0110010101
gen.code 5: 0111000110
frozen    :  F  F FF
```

Figure 3. Frozen genes

Fig.3. These genes are called frozen (denoted with 'F' in Fig.3). If the number of frozen genes is l, the search space becomes 2^l times smaller, and the possibility of premature convergence rapidly increases. Selection and crossover operator can not change bit value of any frozen gene, and the basic mutation rate is often insufficiently small to restore the lost subregions of the search space. If the basic mutation rate is significantly increased, genetic algorithm becomes a random search. For this reason, the mutation rate is increased only on frozen genes, but not more than few times. In this GA implementation, frozen genes are mutated with 2.5 times higher rate than non-frozen ones (1.0/n instead of 0.4/n).

4.3. Generation replacement strategy

The initial population, which numbers 150 individuals, is randomly generated. This approach provides maximal diversity of the genetic material and better gradient of the objective function. One third of the population is replaced in every generation, except for the best 100 individuals that are directly passing to the next generation, preserving highly fitted genes. The objective values of elite individuals do not need recalculation, since each of them is evaluated in one of the previous generations. This approach is denoted as steady-state replacement with elitist strategy in the literature (Kratica, 2000; Kratica et al., 2001).

In order to obtain more correct individuals in the initial population, the probability of generating ones in genetic codes is set to p/n. The individuals that have k, $k \neq p$ ones in their genetic code are incorrect, and they are corrected by adding/erasing $|p-k|$ ones at/from the end of genetic code. The applied genetic operators preserve the fixed number of hubs, so that incorrect individuals do not appear in the following generations.

Duplicated individuals are removed in every GA generation. Their fitness values are set to zero, so that selection operator prevents them to enter the next generation. This is a very effective method for saving the diversity of genetic material and keeping the algorithm away from premature convergence. Individuals with the same objective function but different genetic codes, in some cases may dominate in the population. If their codes are similar, GA can lead to local optimum. For that reason, it is useful to limit their appearance to some constant N_{rv} (it is set to 40 in this GA application).

4.4. Caching GA

The running time of the GA is improved by caching (Kratica, 1999, 2000). The objective functions evaluations are stored in a cache-queue data structure. When the same code is obtained again, its function value is taken from the hash-queue table. The least recently used (LRU) strategy is used for caching GA. The number of cached function values is limited to $N_{cache} = 5000$ in this implementation.

5. Computational results

5.1. Instances and computational environment

In this section the computational results of the GA are presented. All tests were carried out on an AMD K7 1.33GHz with 256 MB memory. The algorithm was coded in C programming language. Two sets of ORLIB (Beasley, 1996) instances were used:

- **CAB (Civil Aeronautics Board)** data set, based on airline passenger flow between cities of United States. It contains 60 instances with up to 25 nodes and up to 4 hubs. Collection and distribution costs χ and δ are equal to one, while transferring cost α takes values from 0.2 to 1. The distances between cities satisfy the triangle inequality, and the flow is symmetric. Detailed information about CAB instances can be found in Beasley (1996) and Campbell (1996).
- **AP (Australian Post)** data set is derived from the study of Australian postal delivery system. Its largest instance includes 200 nodes (representing postcode districts), but smaller ones with 10, 20, 25, 50, 100 nodes can be obtained through the aggregation of nodes. The number of hubs (mail sorting/consolidation centres) in tested instances is up to 20. The flow matrix W_{ij} is non-symmetric and $W_{ii} \neq 0$, since the mail can be sent from one place to itself. The AP data set can also be taken from Beasley (1996).

5.2. Results of the GA and comparisons with other methods

The parameters mentioned above, that proved to be robust and appropriate for this problem, are used. The maximal number of generations is $N_{gen} = 500$ for smaller, and $N_{gen} = 5000$ for larger problem instances. Algorithm also stops if the best individual or the best objective value remained unchanged through $N_{rep} = 200$ ($N_{rep} = 2000$) successive generations, respectively. On all the instances we considered, this criterion allowed GA to converge to high-quality solutions. Only minor or no improvements in the quality of final solutions can be expected when prolonging the runs, as it can be seen from the Tables 1-3.

Table 1 provides results of the GA approach for CAB instances, while Table 2 and Table 3 contain results obtained for smaller/larger AP instances respec-

tively. The GA was run 20 times on each instance, except for larger AP instances (with $n \geq 100$) that were run only 10 times, because of time consuming objective function computation.

In the first column instance dimensions (n, p and possibly α) are given. The second column contains optimal solution of the current instance, if it is previously known. If it is not, the dash (-) appears. The best value of the GA is given in the next column, with mark *opt* in cases when GA reached optimal solution known in advance. Average time needed to detect the best value is given in $t[s]$ column, while $t_{tot}[s]$ represents the total time needed to execute all 500/5000 generations. The GA concept cannot prove optimality and an adequate finishing criterion that will fine-tune solution quality does not exist. Therefore, the algorithm runs through additional $t_{tot} - t$ time (until finishing criterion is satisfied), although it already reached its best/optimal solution. On average, the best/optimal value has been reached after *gen* generations.

The solution quality in all 20/10 executions is evaluated as a percentage gap with respect to the optimal cost OPT_{sol} or GA_{best}, with standard deviation of the average gap σ. The last two columns are related to caching: *eval* represents the average number of needed evaluations, while *cache*[%] displays savings (in percent) achieved by using the caching technique.

It is evident from Tables 1 and 2 that the proposed GA method quickly reaches all previously known optimal solutions on CAB and smaller AP instances. For the CAB data set, the optimal solution was detected in $t[s] \leq 0.048$, while the total running time $t_{tot}[s] \leq 0.161$ seconds. For AP data set, the CPU times were $t[s] \leq 0.571$ and $t_{tot}[s] \leq 1.282$ seconds. On average, instead of making 25 000 calls of the objective function, between 82.1% and 97.2% of the values from the cache-queue table were re-used while solving CAB instances, and between 65% and 98.5% while solving smaller AP instances (see *cache*[%] columns). Table 3 provides results of the proposed GA approach for 42 large AP instances with $40 \leq n \leq 200$ nodes and $2 \leq p \leq 20$ hubs. For only 9 large AP instances the optimal solution is known from the literature, and for the remaining 33 instances no optimal or any other solution is given in the literature so far. As it can be seen from the Table 3, the GA reaches all optimal solutions, but also provides results on the unsolved AP instances in a reasonable computational time. For the largest AP instance with $n = 200, p = 20$, the best GA solution was found in $t[s] = 1935.840$ seconds, while the total running time was $t_{tot}[s] = 2425.588$. The values stored in the cache table provided between 42.3% and 96.1% of savings, instead of 250 000 calculations of the objective function.

The detailed comparisons of the proposed GA with the best-known heuristic and exact methods for solving the UMApHMP are presented in Tables 4-6. The best GA results on the CAB and AP data sets were compared with the results obtained by the **Shortest-Path Based Heuristic *(SPBH)***, **Explicit Enumeration Heuristic *(EEH)*** and **Exact Shortest Path Based Branch-and-Bound Algorithm *(SPBnB)***, which were proposed in Ernst and Krishnamoorthy (1998b) and tested on DEC 3000/700 (200MHz alpha chip).

Table 1. GA results on CAB instances

$n\ p\ \alpha$	OPT_{sol}	GA_{best}	$t[s]$	$t_{tot}[s]$	gen	$gap[\%]$	$\sigma[\%]$	$eval$	$cache[\%]$
20 2 0.2	972.251	opt	0.006	0.046	209	0.0	0.0	296	97.2
20 2 0.4	1013.358	opt	0.004	0.045	210	0.0	0.0	296	97.2
20 2 0.6	1046.895	opt	0.006	0.046	210	0.0	0.0	296	97.2
20 2 0.8	1075.301	opt	0.003	0.044	201	0.0	0.0	297	97.1
20 2 1.0	1090.628	opt	0.004	0.045	204	0.0	0.0	296	97.1
20 3 0.2	712.090	opt	0.013	0.066	213	0.0	0.0	950	91.2
20 3 0.4	803.810	opt	0.014	0.067	213	0.0	0.0	938	91.3
20 3 0.6	884.636	opt	0.016	0.068	215	0.0	0.0	944	91.3
20 3 0.8	948.415	opt	0.009	0.064	208	0.0	0.0	949	91.0
20 3 1.0	975.532	opt	0.013	0.068	209	0.0	0.0	948	91.1
20 4 0.2	568.505	opt	0.027	0.101	226	0.0	0.0	1656	85.5
20 4 0.4	694.557	opt	0.018	0.099	210	0.0	0.0	1603	85.0
20 4 0.6	788.594	opt	0.024	0.102	215	0.0	0.0	1596	85.4
20 4 0.8	870.076	opt	0.022	0.102	215	0.0	0.0	1595	85.4
20 4 1.0	934.083	opt	0.023	0.103	216	0.0	0.0	1586	85.5
25 2 0.2	996.022	opt	0.003	0.053	201	0.0	0.0	410	96.0
25 2 0.4	1072.489	opt	0.003	0.052	201	0.0	0.0	407	96.0
25 2 0.6	1137.081	opt	0.002	0.054	201	0.0	0.0	409	96.0
25 2 0.8	1180.020	opt	0.003	0.054	201	0.0	0.0	410	96.0
25 2 1.0	1206.620	opt	0.003	0.053	201	0.0	0.0	410	96.0
25 3 0.2	752.907	opt	0.022	0.095	218	0.0	0.0	1237	88.8
25 3 0.4	859.636	opt	0.017	0.093	209	0.0	0.0	1236	88.4
25 3 0.6	949.230	opt	0.017	0.094	209	0.0	0.0	1246	88.3
25 3 0.8	1020.037	opt	0.017	0.095	209	0.0	0.0	1249	88.2
25 3 1.0	1062.144	opt	0.021	0.099	213	0.0	0.0	1250	88.4
25 4 0.2	618.483	opt	0.048	0.153	233	0.0	0.0	2028	82.8
25 4 0.4	754.489	opt	0.045	0.153	228	0.0	0.0	1982	82.9
25 4 0.6	866.445	opt	0.023	0.145	209	0.0	0.0	1892	82.2
25 4 0.8	951.755	opt	0.027	0.152	210	0.0	0.0	1910	82.1
25 4 1.0	1006.657	opt	0.029	0.161	210	0.0	0.0	2021	81.1

Table 2. GA results on AP instances

n p	OPT_{sol}	GA_{best}	$t[s]$	$ttot[s]$	gen	$gap[\%]$	$\sigma[\%]$	$eval$	$cache[\%]$
10 2	163603.94	opt	0.001	0.037	201	0.000	0.000	156	98.5
10 3	131581.79	opt	0.001	0.038	201	0.000	0.000	268	97.4
10 4	107354.73	opt	0.004	0.040	204	0.000	0.000	351	96.6
10 5	86028.88	opt	0.003	0.042	201	0.000	0.000	384	96.2
10 6	72427.73	opt	0.002	0.042	201	0.000	0.000	341	96.6
10 7	63466.81	opt	0.002	0.041	202	0.000	0.000	273	97.3
10 8	54628.75	opt	0.002	0.041	202	0.000	0.000	196	98.1
20 2	168599.79	opt	0.004	0.045	201	0.000	0.000	297	97.1
20 3	148048.30	opt	0.012	0.065	210	0.000	0.000	883	91.7
20 4	131665.43	opt	0.017	0.091	213	0.000	0.000	1461	86.5
20 5	118934.97	opt	0.020	0.119	210	0.000	0.000	1809	83.0
20 6	107005.85	opt	0.045	0.161	226	0.000	0.000	2239	80.4
20 7	97697.75	opt	0.031	0.184	209	0.000	0.000	2301	78.4
20 8	91454.83	opt	0.060	0.211	227	0.000	0.000	2313	79.8
25 2	171298.10	opt	0.003	0.051	201	0.000	0.000	411	96.0
25 3	151080.66	opt	0.016	0.088	209	0.000	0.000	1130	89.3
25 4	135638.58	opt	0.028	0.139	212	0.000	0.000	1851	82.8
25 5	120581.99	opt	0.051	0.208	223	0.000	0.000	2464	78.2
25 6	110835.82	opt	0.094	0.277	246	0.000	0.000	2913	76.5
25 7	103880.23	opt	0.139	0.374	257	0.000	0.000	3461	73.2
25 8	97795.59	opt	0.155	0.453	252	0.000	0.000	3750	70.5
40 2	173415.96	opt	0.025	0.102	211	0.000	0.000	783	92.7
40 3	155458.61	opt	0.073	0.245	226	0.000	0.000	2062	82.0
40 4	140682.74	opt	0.131	0.452	234	0.000	0.000	3265	72.5
40 5	130384.74	opt	0.358	0.788	299	0.024	0.060	4667	68.8
50 2	174390.03	opt	0.061	0.173	228	0.000	0.000	1078	90.6
50 3	156014.72	opt	0.240	0.511	288	0.000	0.000	3069	78.6
50 4	141153.38	opt	0.384	0.885	285	0.000	0.000	4419	69.0
50 5	129412.60	opt	0.571	1.282	301	0.015	0.036	5280	65.0

Table 3. GA results on large AP instances

n p	OPT_{sol}	GA_{best}	$t[s]$	$t_{tot}[s]$	gen	$gap[\%]$	$\sigma[\%]$	$eval$	$cache[\%]$
40 6	122171.26	opt	0.247	4.834	2039	0.000	0.000	23349	77.1
40 7	-	116036.38	0.467	6.002	2086	0.000	0.000	25307	75.8
40 8	-	109971.92	0.579	7.655	2085	0.000	0.000	28348	72.9
40 9	-	104212.42	0.884	9.010	2127	0.000	0.000	29598	72.2
40 10	-	99452.67	0.779	9.491	2085	0.000	0.000	27863	73.3
50 6	121671.76	opt	1.537	9.150	2284	0.000	0.000	31036	72.9
50 7	-	115911.64	4.872	15.725	2851	0.000	0.000	46503	67.4
50 8	-	109926.60	4.294	17.188	2591	0.000	0.000	44043	66.0
50 9	-	104968.27	2.869	17.252	2298	0.000	0.000	38930	66.2
50 10	100508.95	opt	4.333	21.136	2412	0.000	0.000	42743	64.7
50 11	-	96186.22	5.294	24.675	2454	0.000	0.000	44999	63.4
50 12	-	93171.96	3.714	23.870	2267	0.000	0.000	39458	65.2
50 13	-	90409.79	4.255	27.221	2281	0.000	0.000	41079	64.1
50 14	-	87654.61	3.972	29.098	2238	0.000	0.000	40315	64.1
50 15	-	85032.89	7.463	35.493	2456	0.000	0.000	45615	62.9
50 20	-	73490.33	2.824	39.859	2094	0.000	0.000	38133	63.6
100 2	176245.38	opt	0.639	2.736	2089	0.000	0.000	4088	96.1
100 3	157869.93	opt	2.195	13.227	2207	0.000	0.000	21017	81.0
100 4	143004.31	opt	9.007	32.848	2652	0.000	0.000	44346	66.6
100 5	133482.57	opt	20.067	54.389	3097	0.000	0.000	60475	60.9
100 6	-	126107.56	58.421	99.973	4350	0.000	0.000	94424	56.6
100 7	-	120165.15	45.945	100.118	3553	0.011	0.024	80659	54.6
100 8	-	114295.92	77.750	125.793	3891	0.228	0.355	87852	54.8
100 9	-	109448.87	54.651	126.037	3409	0.002	0.005	77693	54.6
100 10	-	104794.05	63.355	146.263	3421	0.001	0.002	79849	53.4
100 15	-	88882.05	150.193	270.956	4004	0.093	0.162	93755	53.1
100 20	-	79191.02	195.747	377.160	3828	0.139	0.152	96737	49.5
200 2	178093.99	opt	8.123	35.686	2129	0.000	0.000	10048	90.6
200 3	159725.11	opt	43.393	174.900	2520	0.000	0.000	40939	67.6
200 4	-	144508.20	172.663	376.815	3585	0.001	0.002	78983	56.0
200 5	-	136761.83	357.326	562.245	4231	0.096	0.092	103391	51.2
200 6	-	129560.60	393.868	681.338	4281	0.046	0.062	111529	47.9
200 7	-	123609.44	460.543	766.016	4219	0.051	0.070	112515	46.7
200 8	-	117709.98	566.177	879.377	4237	0.213	0.189	115253	45.8
200 9	-	112380.66	869.886	1096.180	4809	0.066	0.146	131684	45.3
200 10	-	107846.82	847.216	1157.049	4591	0.090	0.189	127817	44.4
200 15	-	92669.64	1246.186	1750.105	4699	0.397	0.275	135060	42.6
200 20	-	83385.94	1935.840	2425.588	4924	0.169	0.232	142223	42.3

Table 4. Comparisons on CAB instances

				GA			SPBH		EEH		SPBnB	
n	p	α	Opt_{sol}	best	$gap[\%]$	$t_{tot}[s]$	$gap[\%]$	$t_{[}s]$	$gap[\%]$	$t[s]$	nodes	$t[s]$
20	2	0.2	972.251	opt	0.0	0.046	0.0	0.04	0.0	0.05	103	0.11
20	2	0.4	1013.358	opt	0.0	0.045	0.0	0.04	0.0	0.05	83	0.11
20	2	0.6	1046.895	opt	0.0	0.046	0.0	0.04	0.0	0.05	76	0.11
20	2	0.8	1075.301	opt	0.0	0.044	0.0	0.04	0.0	0.05	83	0.11
20	2	1.0	1090.628	opt	0.0	0.045	0.0	0.04	0.0	0.05	90	0.11
20	3	0.2	712.090	opt	0.0	0.066	0.0	0.09	0.0	0.43	102	0.22
20	3	0.4	803.810	opt	0.0	0.067	0.0	0.09	0.0	0.43	92	0.22
20	3	0.6	884.636	opt	0.0	0.068	0.0	0.09	0.0	0.43	116	0.25
20	3	0.8	948.415	opt	0.0	0.064	0.0	0.10	0.0	0.43	138	0.27
20	3	1.0	975.532	opt	0.0	0.068	0.0	0.09	0.0	0.43	92	0.23
20	4	0.2	568.505	opt	0.0	0.101	0.0	0.20	0.0	2.43	161	0.40
20	4	0.4	694.557	opt	0.0	0.099	0.0	0.19	0.0	2.41	177	0.44
20	4	0.6	788.594	opt	0.0	0.102	0.0	0.19	0.0	2.43	184	0.46
20	4	0.8	870.076	opt	0.0	0.102	0.0	0.18	0.0	2.40	190	0.49
20	4	1.0	934.083	opt	0.0	0.103	0.0	0.18	0.0	2.46	261	0.57
25	2	0.2	996.022	opt	0.0	0.053	0.0	0.08	0.0	0.12	76	0.16
25	2	0.4	1072.489	opt	0.0	0.052	0.0	0.07	0.0	0.12	73	0.18
25	2	0.6	1137.081	opt	0.0	0.054	0.0	0.07	0.0	0.12	77	0.21
25	2	0.8	1180.020	opt	0.0	0.054	0.0	0.08	0.0	0.12	87	0.20
25	2	1.0	1206.620	opt	0.0	0.053	0.0	0.08	0.0	0.12	93	0.20
25	3	0.2	752.907	opt	0.0	0.095	0.0	0.18	0.0	1.28	148	0.45
25	3	0.4	859.636	opt	0.0	0.093	0.0	0.19	0.0	1.30	151	0.49
25	3	0.6	949.230	opt	0.0	0.094	0.0	0.19	0.0	1.32	152	0.49
25	3	0.8	1020.037	opt	0.0	0.095	0.0	0.19	0.0	1.29	177	0.55
25	3	1.0	1062.144	opt	0.0	0.099	0.0	0.20	0.0	1.29	176	0.52
25	4	0.2	618.483	opt	0.0	0.153	0.0	0.45	0.0	9.17	257	1.03
25	4	0.4	754.489	opt	0.0	0.153	0.0	0.47	0.0	9.29	304	1.14
25	4	0.6	866.445	opt	0.0	0.145	0.0	0.42	0.0	9.18	386	1.14
25	4	0.8	951.755	opt	0.0	0.152	0.0	0.49	0.0	9.11	399	1.44
25	4	1.0	1006.657	opt	0.0	0.161	0.0	0.44	0.0	9.14	364	1.30

Table 5. Comparisons on AP instances

			GA			SPBH		SPBnB	
n	p	Opt_{sol}	best	$gap[\%]$	$t_{tot}[s]$	$gap[\%]$	$t[s]$	nodes	$t[s]$
10	2	163603.94	opt	0.000	0.037	0.00	0.00	23	0.01
10	3	131581.79	opt	0.000	0.038	0.00	0.01	56	0.02
10	4	107354.73	opt	0.000	0.040	0.00	0.02	92	0.03
10	5	86028.88	opt	0.000	0.042	0.00	0.02	92	0.04
20	2	168599.79	opt	0.000	0.045	0.00	0.04	38	0.08
20	3	148048.30	opt	0.000	0.065	0.00	0.09	160	0.25
20	4	131665.43	opt	0.000	0.091	0.00	0.19	456	0.70
20	5	118934.97	opt	0.000	0.119	0.00	0.33	889	1.39
25	2	171298.10	opt	0.000	0.051	0.00	0.08	45	0.13
25	3	151080.66	opt	0.000	0.088	0.00	0.18	213	0.51
25	4	135638.58	opt	0.000	0.139	0.77	0.40	708	1.68
25	5	120581.99	opt	0.000	0.208	0.00	0.67	1053	3.15
40	2	173415.96	opt	0.000	0.102	0.00	0.44	91	0.76
40	3	155458.61	opt	0.000	0.245	0.00	1.11	521	3.56
40	4	140682.74	opt	0.000	0.452	0.00	2.52	1869	13.82
40	5	130384.74	opt	0.024	0.788	0.00	4.43	6310	50.72
50	2	174390.03	opt	0.000	0.173	0.00	1.24	119	2.00
50	3	156014.72	opt	0.000	0.511	0.00	3.37	765	8.91
50	4	141153.38	opt	0.000	0.885	0.00	7.06	3183	40.14
50	5	129412.60	opt	0.015	1.282	0.00	10.95	10187	143.48

Table 6. Comparisons on large AP instances

			GA			SPBnB	
n	p	Opt_{sol}	best	$gap[\%]$	$t_{tot}[s]$	nodes	$t[s]$
40	6	122171.26	opt	0.000	4.834	23	3242.15
40	7	-	116036.38	0.000	6.002	-	-
40	8	-	109971.92	0.000	7.655	-	-
40	9	-	104212.42	0.000	9.010	-	-
40	10	-	99452.67	0.000	9.491	-	-
50	6	121671.76	opt	0.000	9.150	60	18472.84
50	7	-	115911.64	0.000	15.725	-	-
50	8	-	109926.60	0.000	17.188	-	-
50	9	-	104968.27	0.000	17.252	-	-
50	10	100508.95	opt	0.000	21.136	2125737	57243.30
50	11	-	96186.22	0.000	24.675	-	-
50	12	-	93171.96	0.000	23.870	-	-
50	13	-	90409.79	0.000	27.221	-	-
50	14	-	87654.61	0.000	29.098	-	-
50	15	-	85032.89	0.000	35.493	-	-
50	20	-	73490.33	0.000	39.859	-	-
100	2	176245.38	opt	0.000	2.736	400	25.54
100	3	157869.93	opt	0.000	13.227	3198	162.63
100	4	143004.31	opt	0.000	32.848	25780	1097.19
100	5	133482.57	opt	0.000	54.389	153266	7687.75
100	6	-	126107.56	0.000	99.973	-	-
100	7	-	120165.15	0.011	100.118	-	-
100	8	-	114295.92	0.228	125.793	-	-
100	9	-	109448.87	0.002	126.037	-	-
100	10	-	104794.05	0.001	146.263	-	-
100	15	-	88882.05	0.093	270.956	-	-
100	20	-	79191.02	0.139	377.160	-	-
200	2	178093.99	opt	0.000	35.686	1432	384.67
200	3	159725.11	opt	0.000	174.900	25349	3636.64
200	4	-	144508.20	0.001	376.815	-	-
200	5	-	136761.83	0.096	562.245	-	-
200	6	-	129560.60	0.046	681.338	-	-
200	7	-	123609.44	0.051	766.016	-	-
200	8	-	117709.98	0.213	879.377	-	-
200	9	-	112380.66	0.066	1096.180	-	-
200	10	-	107846.82	0.090	1157.049	-	-
200	15	-	92669.64	0.397	1750.105	-	-
200	20	-	83385.94	0.169	2425.588	-	-

The proposed GA and SPBH, EEH and SPBnB methods were not tested on the same platform, so exact comparisons can not be carried out. According to the SPEC-fp95 SPEC-fp2000 benchmarks (www.spec.org), computers AMD at 1.33GHz and DEC 3000/700 have average (base) speedup values 29.4 and 5.71 respectively. In order to provide some descriptive comparisons of CPU times, we observe $t[s]$ and $t_{tot}[s]$ times of the GA multiplied by $29.4/5.71 = 5.2$ factor.

As it can be seen from Table 4, all three heuristic methods obtain optimal solutions on CAB instances. The total running times of the GA (multiplied by 5.2 factor) and the SPBH are similar, while the the exact SPBnB is slightly slower. The EEH is several times slower in comparison with other three methods.

The results of the EEH on smaller size AP instances were not presented in Ernst and Krishnamoorthy (1998b), so in Table 5 only the comparisons of the GA, SPBH and SPBnB results are given. The SPBH method did not reach optimal solution on AP $n = 25, p = 5$ (the average gap is 0.77%), while the GA reached optimal solutions in all cases. The running time of the GA (multiplied by 5.2 factor) is similar or slightly slower in comparison with SPBH. The SPBnB method is significantly slower in comparison with both GA and SPBH. For example, on AP instance $n = 50, p = 5$, the SPBnB gives optimal solution in $t[s] = 143.48$ seconds, the SPBH in $t[s] = 10.95$ and the GA in $t_{tot}[s] * 5.2 = 1.282 * 5.2 \approx 6.66$ seconds.

In Table 6, the comparisons of the GA and SPBnB method are presented, since the results of the SPBH and EEH were not reported in Ernst and Krishnamoorthy (1998b). The proposed GA reached all optimal solutions previously obtained by exact SPBnB method (on 9 out of 42 large AP instances). Comparing the values in column $t_{tot}[s]$ (multiplied by 5.2 factor) and column $t[s]$, it can be seen that the GA concept reached optimal solution in several times shorter CPU time compared to the SPBnB. For example, for the largest AP instance $n = 200, p = 3$ that was solved to optimality, the CPU times of the SPBnB and the GA are $t[s] = 3636.64$ and $t_{tot}[s] * 5.2 = 174.9 * 5.2 \approx 909.48$ seconds, respectively. For the remaining 33 large AP instances that could not be solved by the SPBnB or any other method up to now, the total running time of the proposed GA is reasonably short ($t_{tot}[s] \leq 41$ min).

6. Conclusions

In this paper a genetic algorithm based on the binary encoding for the UMApHMP is proposed. The initial population is randomly generated with p/n probability of generating ones in the genes. Unfeasible individuals in the initial population are corrected to be feasible. Shortest-path objective function is used in this GA approach. New genetic operators, adopted to the problem are constructed. They keep the feasibility of individuals by preserving exactly p ones in their genetic codes. By applying mutation with frozen bits, and by limiting the number of individuals with the same objective function and different genetic codes, the diversibility of genetic material is considerably increased. Implemented caching

GA technique improves the running time significantly.

The algorithm reaches all best (optimal) solutions known from the literature. The GA also gives results on the challenging AP instances unsolved before. Because of these characteristics, the proposed GA approach is a valuable addition to the repertoire of algorithms for solving hub location problems.

Future work will be directed to: parallel implementation, incorporation of some problem-dependent local search heuristics and solving similar hub location problems.

Acknowledgement

This work is partially supported by the Serbian Ministry of Science under the grant 144007. The author is grateful to Jozef Kratica for his useful comments and suggestions on a draft version of this paper.

References

ABADINOUR-HELM, S. (1998) A Hybrid Heuristic for the Uncapacitated Hub Location Problem. *European Journal of Operational Research* **106**, 489–499.

ABADINOUR-HELM, S. (2001) Using simulated annealing to solve the p-hub median problem. *International Journal of Physical Distribution and Logistics Management* **31** (3), 203–220.

ABADINOUR-HELM, S. and VENKATARAMANAN, M.A. (1998) Solution Approaches to Hub Location Problems. *Annals of Operations Research* **78**, 31–50.

AHUJA, R., MAGNANTI, T. and ORLIN, J. (1993) *Network Flows: Theory, Algorithms and Applications.* Prentice-Hall, New York.

ALUMUR, S. and KARA, B.Y. (2008) Network hub location problems: The state of the art. *European Journal of Operational Research* **190** (1), 1–21.

AYKIN, T. (1995) Networking Policies for Hub-and-spoke Systems with Application to the Air Transportation System. *Transportation Science* **29**, 201–221.

BÄCK, T., FOGEL, D.B. and MICHALEWICZ, Z. (2000) *Evolutionary Computation 1: Basic Algorithms and Operators.* Institute of Physics Publishing, Bristol-Philadelphia.

BÄCK, T., FOGEL, D.B. and MICHALEWICZ, Z. (2000) *Evolutionary Computation 2: Advanced Algorithms and Operators.* Institute of Physics Publishing, Bristol-Philadelphia.

BEASLEY, J.E. (1996) Obtaining Test Problems via Internet. *Journal of Global Optimization* **8**, 429–433. *http://mscmga.ms.ic.ac.uk/jeb/orlib/info.html*

BOLAND, N., KRISHNAMOORTHY, M., ERNST, A.T. and EBERY, J. (2004) Preprocessing and Cutting for Multiple Allocation Hub Location Problems. *European Journal of Operational Research* **155**, 638–653.

Bryan, D.L. (1998) Extensions to the hub location problem: Formulations and numerical examples. *Geographical Analysis* **30**, 315–330.

Bryan, D.L. and O'Kelly, M.E. (1999) Hub-and-spoke networks in air transportation: An analytical review. *Journal of Regional Science* **39**, 275–295.

Campbell, J.F. (1992) Location and allocation for distribution systems with transshipments and transportation economies of scale. *Annals of Operations Research* **40**, 77–99.

Campbell, J.F. (1994) Integer programming formulations of discrete hub location problems. *European Journal of Operational Research* **72**, 387–405.

Campbell, J.F. (1996) Hub Location and the p-hub Median Problem. *Operations Research* **44** (6), 923–935.

Campbell, J.F., Ernst, A. and Krishnamoorthy, M. (2002) Hub Location Problems. In: H. Hamacher and Z. Drezner, eds., *Location Theory: Applications and Theory*. Springer-Verlag, Berlin-Heidelberg, 373–407.

Campbell, J.F., Stiehr, G., Ernst, A.T. and Krishnamoorthy, M. (2003) Solving hub arc location problems on a cluster of workstations. *Parallel Computing* **29**, 555–574.

Campbell, J.F., Ernst, A. and Krishnamoorthy, M. (2005a) Hub Arc Location Problems: Part I - Introduction and Results. *Management Science* **51**, 1540–1555.

Campbell, J.F., Ernst, A. and Krishnamoorthy, M. (2005b) Hub Arc Location Problems: Part II - Formulations and Optimal Algorithms. *Management Science* **51**, 1556–1571.

Ebery, J. (2001) Solving large single allocation p-hub problems with two or three hubs. *European Journal of Operational Research* **128** (2), 447–458.

Elhedhli, S. and Hu, F.X. (2005) Hub-and-spoke network design with congestion. *Computers and Operations Research* **32**, 1615–1632.

Ernst A.T. and Krishnamoorthy, M. (1996) Efficient algorithms for the uncapacitated single allocation p-hub median problem. *Location Science* **4** (3), 139–154.

Ernst A.T. and Krishnamoorthy, M. (1998a) Exact and Heuristic Algorithms for the Uncapacitated Multiple Allocation p-hub Median Problem. *European Journal of Operational Research* **104**, 100–112.

Ernst A.T. and Krishnamoorthy, M. (1998b) An exact solution approach based on shortest-paths for p-hub median problems. *Informs Journal on Computing* **10** (2), 149–162.

Ernst, A.T., Jiang, H. and Krishnamoorthy, M. (2005) Reformulations and computational results for uncapacitated single and multiple allocation hub covering problems. Unpublished Report, CSIRO Mathematical and Information Sciences, Australia.

Filipović, V. (1998) Proposal for Improvement of Tournament Selection Operator in Genetic Algorithms (in Serbian). Master thesis, Faculty of Mathematics, University of Belgrade.

Filipović, V. (2003) Fine-Grained Tournament Selection Operator in Genetic

Algorithms. *Computing and Informatics* **22** (2), 143–161.

FILIPOVIĆ, V. (2006) *Selection and Migration Operators and WEB Services in Paralel Evolutionary Algorithms* (in Serbian). PhD thesis, Faculty of Mathematics, University of Belgrade.

HOLLAND, J.H. (1975) *Adaptation in Natural and Artificial Systems.* The University of Michigan Press, Ann Arbor.

HORNER, M.W. and O'KELLY, M.E. (2001) Embedding economies of scale concepts for hub network design. *Journal of Transport Geography* **9** (4), 255–265.

KARA, B.Y. (1999) *Modeling and analysis of issues in hub location problems.* Ph.D. Thesis, Bilkent University, Industrial Engineering Department, Bilkent, Ankara, Turkey.

KARA, B.Y. and TANSEL, B.C. (1999) On the single-assignment p-hub covering problem. Technical report, Department of Industrial Engineering, Bilkent University, Bilkent 06533, Ankara, Turkey.

KARA, B.Y. and TANSEL, B.C. (2003) The single-assignment hub covering problem: Models and linearizations. *Journal of the Operational Research Society* **54**, 59–64.

KIMMS, A. (2005) Economies of scale in hub-and-spoke network design models: We have it all wrong. Technical Report, Technische Universitat Bergakademie Freiberg, Germany.

KLINCEWICZ, J.G. (1991) Heuristics for the p-hub location problem. *European Journal of Operational Research* **53**, 25–37.

KLINCEWICZ, J.G. (1992) Avoiding local optima in the p-hub location problem using tabu search and GRASP. *Annals of Operations Research* **40**, 283–302.

KRATICA, J. (1999) Improving Performances of the Genetic Algorithm by Caching. *Computers and Artificial Intelligence* **18** (3), 271–283.

KRATICA, J. (2000) *Parallelization of Genetic Algorithms for Solving Some NP-complete Problems* (in Serbian). PhD thesis, Faculty of Mathematics, University of Belgrade.

KRATICA, J., TOŠIĆ, D., FILIPOVIĆ, V. and LJUBIĆ, I. (2001) Solving the Simple Plant Location Problem by Genetic Algorithm. *RAIRO Operations Research* **35** (1), 127–142.

KRATICA, J. and STANIMIROVIĆ, Z. (2006) Solving the Uncapacitated Multiple Allocation p-Hub Center Problem by Genetic Algorithm. *Asia-Pacific Journal of Operational Research* **23** (4), 425–437.

KRATICA, J., STANIMIROVIĆ, Z., TOŠIĆ, D. and FILIPOVIĆ, V. (2006) Two Genetic Algorithms for Solving the Uncapacitated Single Allocation p-Hub Median Problem. *European Journal of Operational Research* **182**, 15–28.

LJUBIĆ, I. (2004) *Exact and Memetic Algorithms for Two Network Design Problems.* PhD thesis, Institute of Computer Graphics, Vienna University of Technology.

Marianov, V., Serra, D. and ReVelle, C. (1999) Location of hubs in a competitive environment. *European Journal of Operational Research* **114**, 363–371.

Nickel, S., Schobel, A. and Sonneborn, T. (2001) Chapter 1: Hub location problems in urban traffic networks. In: J. Niittymaki and M. Pursula, eds., *Mathematics Methods and Optimization in Transportation Systems*, Kluwer Academic Publishers.

O'Kelly, M.E., Skorin-Kapov, D. and Skorin-Kapov, J. (1995) Lower bounds for the hub location problem. *Management Science* **41**, (4) 713–721.

O'Kelly, M.E. (1987) A quadratic integer program for the location of interacting hub facilities. *European Journal ofOperational Research* **32**, 393–404.

O'Kelly, M.E. (1998) A geographer's analysis of hub-and-spoke networks. *Journal of Transport Geography* **6** (3), 171–186.

O'Kelly, M.E. and Bryan, D.L. (1998) Hub location with flow economies of scale. *Transportation Research B* **32** (8), 605—616.

Pirkul, H. and Schilling, D.A. (1998) An efficient procedure for designing single allocation hub and spoke systems. *Management Science* **44** (12), 235–242.

Podnar, H., Skorin-Kapov, J. and Skorin-Kapov, D. (2002) Network cost minimization using threshold-based discounting. *European Journal of Operational Research* **137**, 371–386.

Podnar, H. and Skorin-Kapov, J. (2003) Genetic algorithm for network cost minimization using threshold based discounting. *Journal of Applied Mathematics and Decision Sciences* **7**, 207–228.

Raidl, G.R. and Ljubić, I. (2002) Evolutionary Local search for the Edge-Biconnectivity Augmentation Problem. *Information Processing Letters* **82** (1), 39–45.

Sasaki, M., Suzuki, A. and Drezner, Z. (1999) On the selection of hub airports for the airline hub-and-spoke system. *Computers and OR* **26**, 1411–1422.

Skorin-Kapov, D. (2001) On cost allocation in hub-like network. *Annals of Operations Research* **106**, 63–78.

Skorin-Kapov, D. and Skorin-Kapov, J. (1994) On tabu search for the location of interacting hub facilities. *European Journal of Operational Research* **73**, 502–509.

Skorin-Kapov, D. and Skorin-Kapov, J. (2005) Threshold based discounting networks: The cost allocation provided by the nucleolus. *European Journal of Operational Research* **166**, 154–159.

Skorin-Kapov, D., Skorin-Kapov, J. and O'Kelly, M. (1996) Tight linear programming relaxations of uncapacitated p-hub median problems. *European Journal of Operational Research* **94**, 582–593.

Sohn, J. and Park, S. (1997) A linear program for the two-hub location

problem. *European Journal of Operational Research* **100** (3), 617–622.

Sohn, J. and Park, S. (2000) The single allocation problem in the interacting three-hub network. *Networks* **35** (1), 17–25.

Topcuoglu, H., Corut, F., Ermis, M. and Yilmaz, G. (2005) Solving the uncapacitated hub location problem using genetic algorithms. *Computers and OR* **32** (4), 967–984.

Wagner, B. (2004a) A note on the latest arrival hub location problem. *Management Science* **50** (12), 1751–1752.

Wagner, B. (2004b) Model formulations for hub covering problems. Working paper, Institute of Operations Research, Darmstadt University of Technology.

Yaman, H. (2005) Polyhedral analysis for the uncapacitated hub location problem with modular arc capacities. *SIAM Journal on Discrete Mathematics* **19** (2), 501–522.