GA Inspired Heuristic for Uncapacitated Single Allocation Hub Location Problem

Vladimir Filipović, Jozef Kratica, Dušan Tošić, and Djordje Dugošija *

Abstract In this article, the results achieved by applying GA-inspired heuristic on Uncapacitated Single Allocation Hub Location Problem (USAHLP) are discussed. Encoding scheme with two parts is implemented, with appropriate objective functions and modified genetic operators. The article presents several computational tests which have been conducted with ORLIB instances. Procedures described in related work round distance matrix elements to few digits, so rounding error is significant. Due to this fact, we developed exact total enumeration method for solving subproblem with fixed hubs, named Hub Median Single Allocation Problem (HMSAP). Computational tests demonstrate that GA-inspired heuristic reach all best solutions for USAHLP that are previously obtained and verified branch-and-bound method for HMSAP. Proposed heuristic successfully solved some instances that were unsolved before.

Keywords: Genetic algorithms, Evolutionary computations, Hub location, USAHLP, HMSAP.

Vladimir Filipović · Dušan Tošić · Djordje Dugošija
University of Belgrade, Faculty of Mathematics, Studentski trg 16/IV, 11 000 Belgrade, Serbia, e-mail: vladaf@matf.bg.ac.yu, tdusan@mi.sanu.ac.yu, dugosija@matf.bg.ac.yu

Jozef Kratica
Mathematical Institute, Serbian Academy of Sciences and Arts, Kneza Mihajla 36/III, 11 001 Belgrade, Serbia, e-mail: jkratica@mi.sanu.ac.yu

* This research was partially supported by Serbian Ministry of Science under the grant no. 144007.

1 Introduction

The past four decades have witnessed an explosive growth in the field of network-based facility location modelling. The multitude of applications in practice is a major reason for the great interest in that field. Computer and telecommunication networks, DHL-like services and postal networks, as well as transport systems can be analyzed as a hub network. All those systems contain a set of facilities (locations) that interact with each other, and with given distance and transportation cost. Instead of serving every user from its assigned facility with a direct link, hub network allows the transportation via specified hub facilities. Hubs serve as consolidation and connection points between two locations. Each node is allocated to one or more hubs and the flow from one node to another is realized via one or more hub facilities. Using switching points in the network and increasing transportation between them the capacity of the network can be used more efficiently. This strategy also provides lower transportation cost per unit.

There are various model formulations proposed for the problem of choosing subset of hubs in the given network. They involve capacity restrictions on the hubs, fixed cost, predetermined number of hubs and other aspects. Two allocation schemes in the network can be assumed: single allocation and multiple allocation concept.

In the single allocation hub location problem each node must be assigned to exactly one hub node so that all the transport from (to) each node goes only through its hub. Multiple allocation scheme allows each facility to communicate with more than one hub node. If the number of switching centers is fixed to p, we are dealing with p-hub problems. Capacitated versions of hub problems also exist in the literature, but the nature of capacities is different. The flows between hubs or between hubs and non-hubs can be limited. There are also variants of capacitated hub problems that consider limits on the flow into the hub node, through the node or fixed costs on hubs. One review of hub location problems and their classification can be found in [5, 6].

2 Mathematical formulation

Consider a set $I = \{1, ..., n\}$ of n distinct nodes in the network, where each node denotes to origin/destination or potential hub location. The distance from node i to node j is C_{ij}, and triangle inequality may be assumed [6]. The transportation demand from location i to j is denoted with W_{ij}. Variable $X_{ik} = 1$ if the node i is allocated to hub established at node k. Therefore, $X_{jj} = 1 \Leftrightarrow j$ is hub. Otherwise, if node i is not allocated to hub at node k, variable $X_{ik} = 0$.

Each path from source 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 the destination location. In this, single allocation hub problem, is assumed that the flow from certain node involving only one hub node in all transportation. Parameters χ and δ denote unit costs for collection (communication from origin to the first hub) and distribution (communication from last hub to destination), while α represents transfer cost among hubs. The objective is location hub facilities to minimize the total flow cost. The fixed cost for establishing hub node j is denoted with f_j. Using the notation mentioned above, the problem can be written as:

$$min \sum_{i,j,k,l \in N} W_{ij}(\chi C_{ik} X_{ik} + \alpha C_{kl} X_{ik} X_{jl} + \delta C_{jl} X_{jl}) + \sum_{j} X_{jj} f_j \quad (1)$$

subject to

$$\sum_{k} X_{ik} = 1, \quad for\ every\ i \quad (2)$$

$$X_{kk} - X_{ik} \geq 0 \quad for\ every\ i,\ k \quad (3)$$

$$X_{ik} \in \{0, 1\} \quad for\ every\ i,\ k \quad (4)$$

The objective function (1) minimizes the sum of the origin-hub, hub-hub and hub- destination flow costs multiplied with χ, α and δ factors respectively. Equation (2) forces single allocation scheme - each node is assigned to exactly one hub, and equation (3) allows that particular node can be assigned only to established hub.

3 Previous work

Several methods for solving this problem are described in the literature [2, 3]. Due to the fact that this problem is NP hard, it is shown that its subproblem Hub Median Single Allocation Problem - HMSAP is NP hard [6], many authors recognized that good results can be obtained by applying evolutionary inspired solving strategies. In paper [1] several variants of hybridization Genetic algorithm and Tabu search are proposed. Obtained results are presented on CAB problem instances (from ORLIB, described in [4]). In paper, there isn't any result obtained by applying proposed hybrid algorithms on AP ORLIB instances.

Paper [11] proposed more advanced GA method for solving USAHLP. Proposed method uses more efficient representation, better initialization (initial number of hubs in item is set with more realism) and advanced crossover operator that is well suited to the problem domain. However, in this paper (it was also the case with previous methods) proposed GA method uses the

simplest selection operator - proportional selection. Authors in [11] publish obtained results for both CAB and AP instances.

Paper [7] describes SATLUHLP heuristic that solves USAHLP. Heuristic SATLUHLP is hybrid of Simulated annealing and Tabu search. That heuristic is divided into three levels: the first level is to determine the number of hubs; the second level is to select the hub locations for a given number of hubs; and the third level is to allocate the non-hubs to the chosen hubs. Presented results are obtained by testing on ORLIB instances (CAB and AP) and those results are compared with [11].

4 Proposed GA heuristic method

Genetic algorithms (GAs) are problem-solving metaheuristic methods rooted in the mechanisms of evolution and natural genetics. The main idea was introduced by Holland [9], and in the last three decades GAs have emerged as effective, robust optimization and search methods.

4.1 Representation

Each gene of individual represents one node. Particular gene contains of two parts. The first part is 1 or 0 - it indicates if hub is established at corresponding node, or not. Second part contains number from set $\{0, 1, \ldots, n-1\}$. That number specifies which hub is assigned to fixed non-hub node. Naturally, every hub is assigned to itself. For instance, if non-hub node is assigned to the closest hub, then there will be 0 in the second part. Furthermore, if non-hub node is assigned to hub that is more distant to node than closest hub, but less distant than any other hub, there will be number 1 in second part of genetic code, etc.

The first part of genetic code is generated in random manner. Due to the fact that less distant hubs should be more often selected during generation, it is preferable that second part of genetic code contains large number of zeros. To accomplish that, probability that the first bit in each gene will be set to 1 is $1.0/n$. Bits that follows will have probability to be set to one equals to the half of its predecessor probability - e.g. $0.5/n$, $0.25/n$, $0.125/n$, ... respectively.

4.2 Objective function

Fitness of the individual is calculated according to following procedure:

- First part of each gene gives indexes of established hubs.
- After set of established hubs is obtained, array of established hubs will be sorted (for each non-hub node) in ascending order, according to distance to that specific non-hub node.
- Element that corresponds to specific non-hub node is extracted from second part of every gene. If extracted element has value r (r = 0, 1, ..., $n-1$), then r-th element of (adequately) sorted array will be index of the hub which is specific node assigned to.
- Now, objective value (and fitness of individual) is obtained simply by summing distances source-hub, hub-hub and hub-destination, multiplied with load and with corresponding parameters χ , α and δ.

Sorting of established hubs array according to distance, for each individual, takes part in every generation and that requires the processor's extra work. However, the obtained results confirm our estimate that the processor's extra work has very little influence on overall time of algorithm execution.

4.3 Genetic operators

Genetic operators are designed in following way:

- GA uses FGTS [8] as selection operator. Parameter F_{tour}, that governs selection method is not changed during execution of GA, and its value is 5.4. That value is experimentally obtained. Moreover, FGTS selection with $F_{tour} = 5.4$ behave very well in solving some similar problems.
- After selection, one-position crossover operator has been applied. Probability of crossover is 0.85, which means that about 85% individuals in population will produce offsprings, but in approximately 15% cases crossover will not take part and offsprings will be identical to its parents. Crossover point is chosen on the gene boundary. Therefore, there is no gene splitting.
- Evolutionary method uses simple mutation, which pseudo randomly changes one bit in both parts of every gene. Mutation levels are different in different parts. The first bit in every gene mutates with probability $0.6/n$. The second bit in each gene mutates with probability $0.3/n$ and subsequent bits mutate with probability that is half of its predecessor mutation probability ($0.15/n$, $0.075/n$, $0.0375/n$, $0.01875/n$, etc.).
  During GA execution, sometimes happens that all individuals in population have same bit at specified position. Such bits are known as frozen bits. If number of frozen bits is l, then search space becomes 2^l times smaller and probability of premature convergence quickly rises. Selection and crossover can not change frozen bit. Probability of classic mutation is often too small to successfully restore lost subregions in search space. On the other side, if probability of classic mutation is significant, GA pretty much behave like pure random search procedure. Therefore, mutation probability will be in-

creased only for frozen bits. In this GA, probability of mutation for frozen bits in first part is two and half times higher than probability for non-frozen bits, it is $1.5/n$. Probability of mutation for frozen bits in second part is one and half times higher comparing to non-frozen counterparts, so it is $0.225/n$, $0.1125/n$, $0.055625/n$, etc. Reason for lower mutation probabilities for bits in second part is importance that second part mainly contains zeros. In section that describes representation is already highlighted that zero in second part represents the nearest hub to specific non-hub node. Obtained experimental results also justifies probability setting that is described.

4.4 Other GA aspects

There are many aspects (beside representation, objective function and genetic operators) that have significant influence on GA performance. Most important among them are:

- Population has 150 individuals. Number of individuals does not increase nor decrease during GA execution.
- GA uses steady-state replacement policy and elitist strategy - 100 best fitted individuals (e.g. elite individuals) are directly transferred into new generation and its fitness remains the same and should not be recalculated.
- Duplicate individuals are removed in every generation during GA execution. This is accomplished by setting fitness value of duplicated individual to zero, so that individual won't be selected to pass into new generation during selection phase. On that way, genetic diversity is preserved and premature convergence has very small probability.
- Sometimes, during GA execution it happens that individuals with the same fitness value but different genetic code dominate the population. If genetic codes of such dominating individuals are similar, it can bound GA execution to some local extremum. In order to avoid similar situations, we decided to limit the number of individuals with the same fitness and different genetic code. In the current implementation, that number is 40.
- GA execution is stopped after 1000 generations when larger instances are solved, or after 500 generations on small size USAHLP instances. Algorithm is also stopped if best individual does not improve its value during 200 generations.
- Furthermore, performance of GA is improved by cashing GA [10] and cash size is 5000.
- Previously described representation, initialization, selection and mutation prevent creation of incorrect individuals, so there is no need for some special correction.

5 Computational results

Algorithms are tested on ORLIB instance set, taken from [4].

CAB (Civil Aeronautics Board) data set is based on information about civil air traffic among USA cities. It contains 60 instances, with up to 25 nodes and up to 4 hubs. In this instances is assumed that unit costs for collection and distribution (χ and δ) is 1. Results of the proposed GA implementation (just like implementations described in [7, 11]) obtain optimal solution for all instances, with extremely short execution times. Therefore, results on CAB instances are omitted from this paper and can be downloaded from `http://www.matf.bg.ac.yu/~vladaf/Science/USAHLP/cab.txt` .

Data for AP (Australian Post) set are obtained from Australian Post System. AP contains up to 200 nodes that represent postal areas. Smaller size AP instances are obtained by aggregation of the basic, large, data set. Distances among cities fulfill triangle inequality, but load is not symmetric at all. AP also includes fixed price for hub establishment. Suffix "L" in instance name will indicate that fixed costs are light, and suffix "T" will indicate heavy fixed costs. Larger AP instances, that are significantly larger and therefore more difficult, make that algorithm executes for longer time. Those instances will more likely give us better look on overall behavior of algorithm.

However, a new problem arises there: results (e.g. obtained solutions) that are described in paper [11] are sometimes significantly different to solutions that are obtained by proposed GA method. In direct, personal communication, we asked the author to help us to determine possible reasons for the observed differences. In his answer, Topcuoglu speculates that the reason for this is an accumulated rounding error, because he rounded the distance matrix to three decimal places. Results that are published in [7] are not completely identical to results that gives proposed GA method, but difference is much smaller comparing to [11], since distances are rounded up to six decimal places.

In order to completely clear up dilemmas, we decided to obtained exact solution of USAHLP subproblem, called Hub Median Single Allocation Problem - HMSAP. HMSAP problem is similar to USAHLP, but hubs are fixed. In other words, HMSAP should make assignment of hubs to non-hub nodes so overall traffic cost is minimal. Once when we get set of established hubs (note that all algorithms in comparison got the same set of established hubs), we obtained hub assignment by solving HMSAP problem with classical enumeration algorithm. It is widely known that such algorithm guaranties optimality of obtained solution.

Algorithms are executed on the computer with AMD Sempron 2.3+ processor, which works at 1578 MHz clock and have 256 MB memory. During experiments, computer works on UNIX (Knoppix 3.7) operating system. There were activated all C compiler optimizations during compiling, including AMD processor optimization. Proposed GA was executed 20 times for each problem instance.

Table 1 shows experimental results that are obtained by proposed GA method, results obtained by HMSAP and results presented in [7, 11]. First column identifies AP instance that is solved. Best solution obtained by GA is presented in column *GA.best*. Column t contains average time (expressed in seconds) that GA needs to obtain best solution, and column t_{tot} contains average time (also expressed in seconds) for finishing GA. In average, GA finishes after **gen** generations.

Quality of obtained solution is quantified as average gap (denoted as *avg.gap* and expressed in percents) and it is calculated by following formula: $avg.gap = \frac{1}{20}\sum_{i=1}^{20} gap_i$, where gap_i represent gap that is obtained during i-th execution of GA on specific instance. Gap is calculated in respect to optimal solution *Opt.sol* (if it is already known): $gap_i = \frac{sol_i - Opt.sol}{Opt.sol}100$. In cases where optimal solution is not known in advance, gap is calculated in respect to best found solution *Best.sol*: $gap_i = \frac{sol_i - Best.sol}{Best.sol}100$. Tables also contain standard deviation of the gap (denoted as σ) and it is calculated on the following way: $\sigma = \frac{1}{20}\sqrt{\sum_{i=1}^{20}(gap_i - avg.gap)^2}$.

Column **Hubs** gives information about established hubs. Column **HMSAP Enu** contains information about obtained exact solution of the subproblem when hubs are fixed and next column contains time that enumeration algorithm spent in order to obtain solution. Columns **Topcu.** and **Chen.** contain Topcuoglu's and Cheng's results.

If there is abbreviation "n.t." in table cell, it means that problem instance is not tested. Abbreviation "n.n" means that solving was not necessary - for instance if there is only one established hub, assignment is trivial and it is not necessary to solve HMSAP problem. Abbreviation "n.f" means that algorithm did not finished its work, and "time" indicates that execution lasted more than one day.

All the cells in Table 1, where differences are so significant that can not be easily explained only by accumulation of rounding error, are bolded. There is also some chance that those differences are generated by some differences in downloaded AP problem instances, or by inadequate aggregation.

Papers [7, 11] present results only for AP instances with $\chi = 3$, $\alpha = 0.75$ and $\delta = 2$, so Table 1 contains data for direct comparison among proposed GA, Topcuoglu's algorithm and Cheng's algorithm. Results of proposed GA and HMSAP Enu algorithm on AP instances with different values for χ, α and δ can be downloaded from address `http://www.matf.bg.ac.yu/~vladaf/Science/USAHLP/ap.txt` .

Data in Table 1 indicate that, whenever exact enumeration algorithm obtain the solution, proposed GA also obtained the same solution. In some cases, for example 120T instance, when number of established hubs is small, HMSAP Enu finishes its work quicker than GA, but HMSAP Enu solves only subproblem with fixed established hubs. Furthermore, we can notice

Table 1 Results for comparison GA and HMSAP Enu on AP instances with $\chi = 3$, $\alpha = 0.75$, $\delta = 2$

Inst.	$GA.best$	$t[s]$	$t_{tot}[s]$	**gen**	$avg.gap[\%]$	$\sigma[\%]$	**Hubs**	**HMSAP Enu**	$t[s]$	**Topcu.**	**Chen.**
10L	224250.055	0.009	0.101	217	0.000	0.000	3,4,7	224250.055	0.04	224249.82	224250.06
10T	263399.943	0.020	0.113	249	0.000	0.000	4,5,10	**263399.943**	0.07	**263402.13**	263399.95
20L	234690.963	0.016	0.206	216	0.000	0.000	7,14	234690.963	0.11	234690.11	234690.96
20T	271128.176	0.029	0.213	229	0.909	1.271	7,19	**271128.176**	0.11	**263402.13**	271128.18
25L	236650.627	0.035	0.275	228	0.000	0.000	8,18	**236650.627**	0.20	**236649.69**	236650.63
25T	295667.835	0.004	0.233	201	0.000	0.000	13	n.t.	n.n.	**295670.39**	295667.84
40L	240986.233	0.101	0.554	244	0.221	0.235	14,28	**240986.233**	1.90	**240985.51**	**240986.24**
40T	293164.836	0.069	0.500	231	0.000	0.000	19	n.t.	n.n.	**293163.38**	293164.83
50L	237421.992	0.298	0.904	298	0.327	0.813	15,36	**237421.992**	4.16	**237420.69**	237421.99
50T	300420.993	0.008	0.592	201	0.000	0.000	24	n.t.	n.n.	**300420.87**	**300420.98**
60L	228007.900	0.415	1.205	306	0.546	0.919	18,41	228007.900	7.93	n.t.	n.t.
60T	246285.034	0.231	1.016	258	0.356	1.593	19,41	246285.034	7.54	n.t.	n.t.
70L	233154.289	0.451	1.489	286	0.000	0.000	19,52	233154.289	9.84	n.t.	n.t.
70T	252882.633	0.360	1.397	269	0.000	0.000	19,52	252882.633	9.88	n.t.	n.t.
80L	229240.373	1.143	2.397	383	1.143	1.286	22,55	229240.373	21.34	n.t.	n.t.
80T	274921.572	0.633	1.818	300	0.249	0.765	5,41,52	274921.572	4455	n.t.	n.t.
90L	231236.235	0.919	2.463	319	0.841	0.865	26,82	231236.235	87.23	n.t.	n.t.
90T	280755.459	0.437	1.934	257	0.133	0.395	5,41	280755.459	16.64	n.t.	n.t.
100L	238016.277	1.382	3.221	349	0.381	0.757	29,73	**238016.277**	69.69	**238017.53**	**238015.38**
100T	**305097.949**	0.365	2.180	239	0.000	0.000	52	**n.t.**	n.n.	**305101.07**	**305096.76**
110L	222704.770	3.025	5.205	478	1.430	1.611	32,77	222704.770	7 045	n.t.	n.t.
110T	227934.627	2.604	4.761	438	4.846	5.774	32,77	227934.627	6 718	n.t.	n.t.
120L	225801.362	2.304	4.775	384	0.392	0.778	32,85	225801.362	2 896	n.t.	n.t.
120T	232460.818	3.440	5.913	475	1.741	2.972	32,85	232460.818	2 934	n.t.	n.t.
130L	227884.626	3.563	6.661	428	1.098	1.037	36,88	n.f.	time	n.t.	n.t.
130T	234935.968	3.108	6.181	399	0.398	0.459	36,88	n.f.	time	n.t.	n.t.
200L	**233802.976**	11.521	19.630	482	0.398	0.815	43,148	**n.f.**	time	**228944.77**	**228944.18**
200T	**272188.113**	10.981	19.221	463	0.326	0.215	54,122	n.f.	time	**233537.93**	**233537.33**

four cases where execution of enumeration algorithm lasted extremely long, but GA implementation obtained solution during very small amount of time.

GA implementation is also comparable in terms of running time with Topcuoglu and Chen methods. Note that running time of the GA increases at smaller rate than in [7, 11]. For example, for n=200, GA running time is approximately 20 seconds, while Topcuogly is 3000 seconds and Chen is 180 seconds.

6 Conclusions

In this article, we introduced a GA-inspired heuristic that solves the US-AHLP by simultaneously finding the number of hubs, the location of hubs, and the assignment of nodes to the hubs. The assignment part (HMSAP) is

successfully verified by the results of enumeration method for all cases where exact HMSAP solution can be obtained in reasonable time.

In proposed method, two-part encoding of individuals and appropriate objective functions are used. Arranging located hubs in non-decreasing order of their distances from each non-hub node directs GA to promising search regions. We have used the idea of frozen bits to increase the diversity of genetic material by mutation. The caching technique additionally improves the computational performance of GA.

Extensive computational experiments indicate that the proposed method is very powerful and that the medium-size and large-size USAHLP instances can be solved within a twenty seconds of computing time for sizes attaining 200 nodes. Such results imply that the GA may provide an efficient metaheuristic for real world USAHLP and related problems.

Hence, our future work could also concentrate on the speed-up of the algorithm by taking advantage of parallel computation and on GA hybridization with exact methods.

References

1. Abdinnour-Helm, S.: A hybrid heuristic for the uncapacitated hub location problem. European Journal of Operational Research 106, 489-499 (1998)
2. Abdinnour-Helm, S., Venkataramanan, M.A.: Solution Approaches to Hub Location Problems. Annals of Operations Research 78, 31-50 (1998)
3. Aykin, T.: Networking Policies for Hub-and-spoke Systems with Application to the Air Transportation System. Transportation Science 29, 201-221 (1995)
4. Beasley, J.E.: Obtaining test problems via internet. Journal of Global Optimization 8, 429-433 (1996) `http://mscmga.ms.ic.ac.uk/info.html` `http://www.brunel.ac.uk/depts/ma/research/jeb/orlib`
5. Campbell, J.F.: Hub Location and the p-hub Median Problem. Operations Research 44(6), 923-935 (1996)
6. Campbell, J.F., Ernst, A., Krishnamoorthy, M.: Hub Location Problems. In: Hamacher, H., Drezner, Z. (eds.) Location Theory: Applications and Theory, pp. 373-407. Springer-Verlag, Berlin-Heidelberg (2002)
7. Chen, F.H.: A hybrid heuristic for the uncapacitated single allocation hub location problem. OMEGA - The International Journal of Management Science 35, 211-220 (2007)
8. Filipović, V.: Fine-grained tournament selection operator in genetic algorithms. Computing and Informatics 22, 143-161 (2003)
9. Holland, J.: Adaptation in Natural and Artificial Systems. The University of Michigan Press (1975).
10. Kratica, J.: Improving performances of the genetic algorithm by caching. Computers and Artificial Intelligence 18, 271-283 (1999)
11. Topcuoglu, H., Court, F., Ermis, M., Yilmaz, G.: Solving the uncapacitated hub location problem using genetic algorithms. Computers & Operations Research 32, 967-984 (2005)