XV ECPD INTERNATIONAL CONFERENCE ON MATERIAL HANDLING AND WAREHOUSING
University of Belgrade, Faculty of Mechanical Engineering, Materials Handling Institute; 09.-10. Dec. 1998.

B2 - 12.

SOLVING THE UNCAPACITATED WAREHOUSE LOCATION PROBLEM BY SGA WITH ADD-HEURISTIC

Jozef Kratica
Faculty of Mechanical Engineering, Department of Mathematics, 27. marta 80, Belgrade.
Vladimir Filipović
Dušan Tošić
Faculty of Mathematics, Studentski trg 16, Belgrade.

Abstract: This paper presents a new hybrid approach for solving the Uncapacitated Warehouse Location Problem (UWLP) by Simple Genetic Algorithm (SGA) and Add-Heuristic. The best individual in every generation of SGA is improved by Add-Heuristic, adding or removing one warehouse with maximal reduction of overall cost. The existence of warehouse with adding/removing may reduce overall cost and binary string is changed in the corresponding position. SGA itself produces good results with the reasonable running time, but described hybrid approach improves performance of implementation. The local search nature of Add-Heuristic provides achieving better regions of search for given problem.

Keywords: Simple Genetic Algorithm, Add-Heuristic, Uncapacitated Warehouse Location Problem, NP-hard problems, Combinatorial optimization

1. INTRODUCTION

The location problems have been examined extensively during the past three decades. The review of all the important contributions in that respect is beyond the scope of this paper. There are a lot of survey articles related to this problem and we mention: Dearing (1985); Francis at al. (1983).

The UWLP is a well-known NP-hard combinatorial optimization problem (Krarup and Pruzan, 1983.). This problem is also known as uncapacitated facility location problem or simple plant location problem. The UWLP has been studied previously by many researchers (Krarup and Pruzan, 1983.; Cornuejols, at al., 1990.; Gao, at al., 1994). Although SPLP is NP-hard, in some special cases it is solvable in polynomial time (Grishukhin, 1994).

Some ways for the solving UWLP by simple genetic algorithm (SGA) are proposed in: Kratica at al. (1996); Kratica (1998b).

1.1 Problem formulation

The problem of serving the given set of customers from the chosen warehouses is considered. The objective is to minimize the sum of fixed charges for establishing the warehouses and transportation costs corresponding to the supply of demands.

Consider the set $I = \{1,\ldots, m\}$ of candidate places to locate warehouses, and the set $J = \{1,\ldots, n\}$ of customers. Each warehouse $i \in I$ has a fixed cost f_i. Each customer $j \in J$ has a demand b_j and c_{ij} is the unit transportation cost from warehouse i to customer j. Without loosing a generality we can normalize customer demand to $b_j = 1$.

We should find which warehouses are established and amount supplying from warehouse i to customer j so that the total cost, including both fixed and variable cost, are minimized.

The problem may be formulated mathematically as:

$$\min \left(\sum_{i=1}^{m} \sum_{j=1}^{n} c_{ij} x_{ij} + \sum_{i=1}^{m} f_i y_i \right) \qquad (1)$$

subject to:

$$\sum_{i=1}^{m} x_{ij} = 1, \qquad \text{for every } j \in J; \qquad (2)$$

$$0 \le x_{ij} \le y_i \text{ and } y_i \in \{0,1\}, \quad \text{for every } i \in I \text{ and every } j \in J; \qquad (3)$$

where x_{ij} is the amount supplying from warehouse i to customer j. Array y_i indicates whether warehouse i is established ($y_i = 1$) or not ($y_i = 0$).

Note a set of established warehouses as $E = \{ i \mid y_i = 1 \}$ with cardinality $e = |E|$.

1.2 Genetic algorithms

Genetic Algorithms (GAs) are robust and adaptive methods that may be used to solve the problems related to search and optimization. Detailed bibliography about applying of GAs on the optimization problems is given in: Alander (1995); Osman and Laporte (1996).

GAs work with a population of individuals (usually 10-200), each representing a possible solution for the given problem. To each individual fitness value has assigned according to adaptation of this individual. The population evolves towards better solutions by means of randomized processes of selection, crossover, and mutation.

The selection mechanism favors individuals of better fitness value to reproduce more often than the worse ones when forming a new population. The crossover allows mixing of parental information when it is passed to their descendants. The result of crossover is a structured and randomized exchange of genetic material between solutions, with the possibility that "good" solutions can generate "better" ones.

The mutation changes a bit in the binary string with some small probability p_m. The role of mutation in GAs is restoring lost or unexplored genetic material into the population. It can be used to prevent the premature convergence of GA to suboptimal solutions. The initial population is usually randomly initialized. For detailed information about GA see: Goldberg (1989); Beasley D., at al. (1993). Genetic algorithms were successfully used to approximative solve some NP-hard problems. (De Jong and Spears, 1989.; Khuri, at al., 1994).

Informal description (the rough outline) of GA is given in Figure 1.

```
Input_Data();
Population_Init();
while not Finish() do
   for i:=1 to N_pop do
      p_i: = Objective_Function(i);
   Fitness_Function();
   Selection();
   Crossover();
   Mutation();
Output_Data();
```

Figure 1. Simplest form of GA

2. BASIC GA IMPLEMENTATION

In this section SGA implementation for solving UWLP is concisely described. The detail presentation could be find in: Kratica (1998b). Presented results here are good for test problems with size up to 500×500. In those cases the running time is also reasonably good (see Table 2).

For the given problem binary coding of possible warehouse locations is used. Every bit in the binary string, denotes that particular warehouse is established if its value is 1, while 0 denotes it is not established. For the faster execution the item string is allocated in 32-bit words.

The objective value function is computed depends on number of established warehouses e and threshold e_0. If $e > e_0$ in initialization part, warehouses are indexed in nondecreasing order of transportation costs for every customer. After that the objective value function speedy find first established warehouse, with minimal transportation cost. If $e \le e_0$ previous procedure is not appropriate, and ordinal array of established warehouses is formed. We find the established warehouse with minimal cost searching only the same ordinal array for every customer.

In our case the size of population is 20 individuals. The whole population is replaced in every generation except the best individual. The best individual in the current generation directly goes to the next generation, without selection, crossover and mutation. The initial population is randomly initialized, because the maximal diversity into the population is maintained in that case.

The duplicate strings are discarded from the population, i.e. multiple occurrence of the same string is practically excluded. The duplicate strings in every generation are discarded by setting their fitnesses to zero. This method does not remove duplicate item string physically, but eliminates its occurrence in the next generation. This technique maintains the diversity of genetic material. Discarding of duplicate strings decreases the appearance of dominating individuals in the population, and effectively decreases the possibility of premature

convergence in local optimum. This is a very important factor for successful working GA, particularly in the cases of large size test problems.

Since we solve the minimization problem, all objective values of individuals in the population, are scaled inversely into the interval (0,1). Formula for that scaling is given by.

$$f_i = \frac{\max - p_i}{\max - \min} \qquad (4)$$

In the equation (4) numbers *max* and *min* are maximal and minimal item objective values in the current generation, and p_i is an objective value of item *i*.

GA implementation presented in this paper performs simple roulette selection, one-point crossover and simple mutation. The crossover probability is p_{cross} = 0.85, and mutation with variable rate is used. The formula for the mutation probability is:

$$p_{mut} = \begin{cases} 0.01, & n \le 50 \\ \frac{1}{2n}, & n \ge 100 \end{cases} \qquad (5)$$

As we see from the experimental results presented in Table 2 and Table 3, this mutation rate is good compromise between exploration and exploitation components of GA search, applied to solving UWLP.

Maximal number of iterations is 20 000. If the optimal solution is obtained before 20 000 iterations, execution is stopped, and results are immediately printed. The optimal solution can not be directly recognized by GA, but some indirectly criterions could be used. For example: the best individual is not changed in last M generations. The optimal solution can be used in test-examples if it is available by some other methods.

We also optimize run-time performance of the genetic algorithm by caching. This idea is used to avoid the attempts of computing the same objective value repeatedly many times. Instead of that, objective values are remembered and reused. If the program has computed objective value for a particular item string, and the same string appears again, the cached values are used to avoid computing it many times. For detailed information about caching GA see Kratica (1998a).

3. HYBRID GA IMPLEMENTATION

Add-heuristic appeared as stand-alone method for solving UWLP. In first step one warehouse, giving minimal overall cost, is found and established alone. In every next step one previously non-established warehouse, maximally decreasing overall cost is found and established itself. If such warehouse does not exists, i.e., if all previously non-established warehouses increase overall cost by establishing itself, the algorithm is finished. Current configuration of established warehouses is promoted as final solution. The detailed analysis of various variants of Add-heuristic is presented in Krarup and Pruzan (1983).

The results obtained by Add-heuristic, as stand-alone method are not always satisfactory and because of that Add-heuristic is later adapted for successful hybridization with other methods. One of such adaptation with dual-based algorithms is given in Tcha at. al. (1988).

In this paper Add-heuristic is adapted and hybridized with GA. The modified Add-heuristic is applied in every generation to the best individual in population. Method tries to improve the overall cost by adding or removing one warehouse that maximally decreases overall cost. Candidates for adding are non-established warehouses, and for removing are established warehouses. If the overall cost is improved, new configuration of established warehouses overwrites original genetic string. Add-heuristic can be repeat many times, but we apply it only once to the same individual.

In order to accelerate execution, both established warehouse with minimal cost as well as established warehouse with second minimal transportation cost are memorized. Such information is important in the case when warehouse with minimal transportation cost for current customer is removed. After that the old second minimal transportation cost is directly promoted into the new minimal transportation cost, without additional computation requirement.

4. COMPUTATIONAL RESULTS

Test problems 41-134 and A to C which will be used in this section are taken from the ORLIB (Beasley J.E., 1996). The random problems MO, MP, MQ and MR of larger size, are generated by the author, and these problems are described in detail by Kratica (1998b).

The testing process is performed by PC compatible computer AMD 80486DX5 at 133MHz with 32 MB of memory. The Table 1 contains information about test problem names, number of problems and number of runs for every test problem. Since genetic operators of selection, crossover and mutation are undeterministic, every test problem was running 10 times and we compute an average value.

The results of running hybrid algorithm are summarized in Table 3, while the results obtained by base GA implementation are described in Table 2.

The columns in Tables 2 and 3 contain:

- Names of test problems;
- Average number of generations for finishing GA execution;
- Average execution time in seconds;

- Quality of solutions, i.e., how many solutions are optimal, how many solutions have relative error to optimal solution less than 0.2%, less than 1%, less than 5%, and over 5%.

Table 1. Test problems

Test problem names	Num. of problems	Num. of runs
41-74	13	10
81-104	12	10
111-134	12	10
A - C	3	10
MO1-MO5	5	10
MP1-MP5	5	10
MQ1-MQ5	5	10
MR1-MR5	5	10

Table 2. Results of running base GA

Test problem names	Avg. gener.	Avg. time (s)	Opt.	<0.2 (%)	<1 (%)	<5 (%)	>5 (%)
41-74	51.5	0.228	124	4	1	1	-
81-104	179.0	0.803	120	-	-	-	-
111-134	1 056	5.36	120	-	-	-	-
A-C	10 608	267.5	16	9	5	-	-
MO1-MO5	8 263	64.6	33	10	7	-	-
MP1-MP5	3 120	46.5	47	0	3	-	-
MQ1-MQ5	5 078	116.2	47	1	2	-	-
MR1-MR5	8 728	330.3	40	1	8	1	-

Table 3. Results of running hybrid GA

Test problem names	Avg. gener.	Avg. time (s)	Opt.	<0.2 (%)	<1 (%)	<5 (%)	>5 (%)
41-74	6.9	0.042	130	-	-	-	-
81-104	120.3	0.556	120	-	-	-	-
111-134	930.9	0.89	120	-	-	-	-
A-C	9 989	295	17	8	5	-	-
MO1-MO5	4 811	39.2	41	5	4	-	-
MP1-MP5	2 323	36.4	48	0	2	-	-
MQ1-MQ5	5 508	133.3	44	0	6	-	-
MR1-MR5	8 402	338.2	46	1	3	-	-

The hybrid GA implementation produced better quality of solutions than base GA implementation. For example, the testing MR problem set during 5×10 runs, gives 46 optimal solutions, while the testing of base implementation gives 40 optimal solutions. The exception is the test-problem MQ where base GA application gives better results.

The hybrid GA implementation also creates smaller average number of generations (except test problems MQ). In our testing process, smaller average number of generations corresponded to a better quality of solutions, and vice versa.

The hybrid GA has smaller execution time for test problems 41-134, MO and MP. The difference is especially large for test problems 41-74 and 111-134, where hybrid GA is faster more than 5 times. In the cases that base GA is faster (A-C, MQ and MR), the difference is not significant (less than 10%).

5. CONCLUSIONS

In this paper, the improvement of GA implementation by Add-heuristic for solving UWLP is explored. The GA implementation assisted with Add-Heuristic successfully prevents possibility of premature convergence, and restores lost or unexplored genetic material into the population.

During the testing process, in both cases (base GA and hybrid GA) for every particular test problem at least one-half of optimal solutions is obtained. Both implementations are able to solve large-scale uncapacitated warehouse location problems, with good running time.

According to UWLP benchmark, the use of hybrid GA technique brings some improvements. In general, the average run-time produced by hybrid GA implementation is smaller than the one produced by the base GA implementation. The quality of solutions and average number of generations are also better for hybrid GA implementation.

The research presented in this paper can be continued, and extended in several directions:

- Examination of presented approach to test problems of very large size (over the 1000 warehouses and customers);
- Parallelization of genetic algorithm and implementation to distributed and multiprocessor systems;
- Hybridization of GA (instead Add-heuristic) with some other: dual based improvement, Lagrangean relaxation, Benders decomposition, etc.
- Applying of presented approach to the other related location problems: p-median, capacitated warehouse location, dynamic warehouse location, multi-product uncapacitated warehouse location and data file location/allocation problems.

REFERENCES

Alander, J.T. (1995).: Indexed bibliography of genetic algorithms in operations research, Report 94-1-OR, Department of Information Technology and Production Economics, University of Vassa, Finland. *ftp://ftp.uwasa.fi/cs/report94-1/gaORbib.ps.Z*

Beasley, D., Bull, D.R., Martin, R.R. (1993).: An Overview of Genetic Algorithms, *University Computing*, Vol. 15, pp 170-181.

Beasley, J.E. (1996).: Obtaining test problems via Internet, *Journal of Global Optimization*, Vol. 8, pp 429-433. *http://mscmga.ms.ic.ac.uk/info.html*

Cornuejols, G., Nemhauser, G.L., Wolsey, L.A. (1990).: The uncapacitated facility location problem, In: *Discrete Location Theory*, Eds: P.B. Mirchandani and R.L. Francis, John Wiley & Sons, chap III, pp 120-171.

Dearing, P.M. (1985).: Location problems, *Operations Research Letters*, Vol. 4, pp 95-98.

De Jong, K.E., Spears W.M. (1989).: Using Genetic Algorithms to Solve NP-Complete Problems, In: *Proceedings of the Third International Conference on Genetic Algorithms*, pp 124-132, Morgan Kaufmann, San Mateo, California.

Francis, R.L., McGinnis, L.F., White, J.A. (1983).: Locational analysis, *European Journal of Operational Research*, Vol. 12, pp 220-252.

Gao, L.L., Robinson, E., Powell Jr. (1994).: Uncapacitated facility location: General solution procedure and computational experience, *European Journal of Operational Research*, Vol. 76, No. 3, pp 410-427.

Goldberg, D. (1989).: *Genetic Algorithms in Search, Optimization and Machine Learning*, Addison-Wesley, Reading.

Grishukhin, V.P. (1994).: On polynomial solvability conditions for the simplest plant location problem, In: *Selected topics in discrete mathematics*, Eds: A.K. Kelmans and S. Ivanov, pp 37-46, American Mathematical Society, Providence, RI.

Khuri, S., Back, T., Heitkotter, J. (1994).: An Evolutionary Approach to Combinatorial Optimization Problems, In: *Proceedings of CSC'94*, Phoenix, Arizona.

Krarup, J., Pruzan, P.M. (1983).: The simple plant location problem: Survey and synthesis, *European Journal of Operational Research*, Vol. 12, pp 36-81.

Kratica, J., Filipović, V., Šešum, V., Tošić, D. (1996).: Solving of the Uncapacitated Warehouse Location Problem using a Simple Genetic Algorithm, In: *Proceedings of the XIV International Conference on Material Handling and Warehousing*, Ed. Đ. Zrnić, pp 3.33-3.37, Belgrade.

Kratica, J. (1998a).: Improving Performance of the Genetic Algorithm by Caching, *Accepted for publication in Computers and Artificial Intelligence*, Vol. 15.

Kratica, J. (1998b).: Improvement of Simple Genetic Algorithm for Solving the Uncapacitated Warehouse Location Problem, In: *3nd On-line World Conference on Soft Computing in Engineering Design and Manufacturing - WCS3*, (June 1998). *http://www.bioele.nuee.nagoya-u.ac.jp/WSC3/*

Osman, I.H., Laporte, G. (1996).: Metaheuristic: A bibliorgaphy, *Annals of Operations Research*, Vol. 63, pp 513-623.

Tcha, D.W., Ro, H.B., Yoo, C.B. (1988).: Dual-based Add Heuristic for Uncapacitated Facility Location, *Journal of Operational Research Society*, Vol. 39., No 9.