Acta Agraria Kaposváriensis (2008) Vol 12 No 2, 59-69
Kaposvári Egyetem, Állattudományi Kar, Kaposvár
University of Kaposvár, Faculty of Animal Science, Kaposvár

A memetic Algorithm for the Capacitated Vehicle Routing Problem

I. Borgulya
University of Pécs, Faculty of Business and Economics, Pécs, H-7622 Rákóczi út 80.

ABSTRACT

In this paper we present a new memetic algorithm for the CVRP (Capacitated Vehicle Routing Problem). The new algorithm was developed from our earlier multi-objective algorithm for the vehicle routing problem - selecting and further developing one part of the earlier algorithm. The new algorithm is a steady-state evolutionary algorithm. It uses tournament selection; the descendents are derived from the parents by mutation based on the EVL (Extended Virtual Loser) where the EVL is an explicit collective memory technique. The algorithm is a memetic algorithm and uses five different stochastic 2-opt local searches to improve the descendents. We used some test problems of the Vehicle Routing Data Sets and of Christofides. Comparing the results with other method's results we concluded: in the case of n < 200 costumers we got similar results that was published earlier.
(Keywords: Evolutionary algorithm, explicit collective memory, combinatorial optimization)

ÖSSZEFOGLALÁS

Egy memetikus algoritmus a járatszervezési problémára
Borgulya I.
Pécsi Tudományegyetem, Közgazdaságtudományi Kar, Pécs, 7622 Rákóczi út 80.

A cikkben az egy telephelyes, kapacitással adott járatszervezései problémára (CVRP: Capacitated Vehicle Routing Problem) mutatunk be egy memetikus algoritmust. A megoldáshoz egy korábbi több-célfüggvényes járatszervezési algoritmusunkat használjuk fel, kiemelve és továbbfejlesztve az algoritmusból az egy célfüggvényes járatszervezési problémánál alkalmazható algoritmus részt. Az új algoritmus egy steady-state rendszer, amely tournament szelekciót alkalmaz, az utódokat mutációval generálja a szülőkből, ahol a mutáció egy memória alapú technikán, az EVL (Extended Virtual Loser) technikán alapul. Az algoritmus, mint memetikus algoritmus, az utódok minőségét ötféle sztochasztikus helyi kereső eljárással javítja. Az algoritmust a „Vehicle Routing Data Sets", valamint Christofides néhány tesztfeladatán ellenőriztük. Az eredményeket más módszerekkel is összehasonlítottuk: n < 200 fogyasztó esetén a korábban publikált eredményekhez hasonlót kaptunk.
(Kulcsszavak: Evolúciós algoritmus, explicit kollektív memória, kombinatorikus optimalizálás)

INTRODUCTION

The Vehicle Routing Problem (VRP) is a well-known, often studied problem. Many practical applications exist for various industrial areas (e.g. transport, logistic, workshop problem). One of the simplest versions of the vehicle routing problem is the CVRP. The CVRP is a graph problem that can be described as follows: n customers must be served from a unique depot a quantity q_i of goods ($i = 1, \dots, n$). To deliver those goods, a fleet of vehicles with a capacity C is available. A solution of the CVRP is a collection of tours where each customer is visited only once and the total tour demand is at most C, with the objective f: minimization of the total distance traveled by all the vehicles.

The VRP has been proved NP-hard (*Laporte*, 1992) and applied solution methods range from exact methods to specific heuristics, and meta-heuristics. As exact methods we can use e.g. the branch and bound, and the branch and cut methods (e.g. *Hadjiconstantinou et al*., 1995). As the size of problem gets larger, it is nearly impossible to get a solution. Therefore, different heuristic we can use e.g. the neural network, and as meta-heuristics we can use the simulated annealing, tabu search, evolutionary algorithms, ant colony optimization, particle swarm optimization (e.g. *Sun et al*., 2005; *Van Breedam*, 2001; *Russel et al*., 2005; *Toth et al*., 2003; *Mazzeo et al*.,2004; *Chen et al.,* 2006).

In our earlier work (*Borgulya*, 2008) we presented a new method for a bi-objective CVRP, used a new extended version of an explicit collective memory method, named virtual loser. In the EVL we enabled the virtual loser to handle more discrete values and the values of the variables can be e.g. values of permutations too. In this paper we selected and further developed one part of the earlier multi-objective algorithm; we developed a new evolutionary algorithm (EA) for the CVRP. So:

- We adapted the EVL for the CVRP and developed an EA with two steady-state stages.
- We used tournament selection (instead of truncation selection).
- We used a special mutation operator with two possibility moves: the first is a move based on the EVL, the second is a random move, and
- We used five different stochastic local search procedures to improve the solutions.

We used some benchmark problems of the Vehicle Routing Data Sets and of Christofides and got good results. To compare the results of our algorithm we chose other meta-heuristics, e.g. some versions of the ant colony optimization, tabu search and some EAs. The quality of our results is good, but our algorithm has longer running time than the running times of the best methods.

In addition to this introduction section, this paper is organized into the following sections. Section 2 includes the new EA for the CVRP. In Section 3, we present our computational experience with the new EA and compare our results with other heuristics results. Section 4 contains concluding remarks. The extended virtual loser is described in the Appendix.

THE NEW ALGORITHM

The structure of the algorithm

The new memetic algorithm, named MA, uses a 2-stage algorithm structure. Each stage is a hybrid steady-state EA. The first stage is a quick ”preparatory” stage which is designated to improve the quality of the initial population. In the second stage the descendents are derived from the parents by mutation. In every stages the algorithm uses stochastic 2-opt local searches to improve the solutions.

The main steps of MA:

```
Procedure MA(t, itt, kn, genlimit, opt, optp)
Initial population. Initial values of ECM
/* First stage:
  Do itt times
Selection, local searches, reinsertion.
In every kn-th iteration:
Update of the ECM, Delete of the duplicates element.
  od.
/* second stage:
  Repeat
Do kn times
   Selection, mutation, local searches, reinsertion.
od.
Update of the ECM. Delete of the duplicates element, Restart.
optp= the best individual, opt=f(optp)
  until genlimit <number of generations
end
```

The parameters of the algorithm:

t - the size of the population,
itt - the number of the generation in the first stage.
kn - the algorithm is controlled in every kn*th* generation.
genlimit - a parameter for the stopping condition. The procedure is finished if the number of the generations is more than *genlimit*.

The characteristics of the EAs

The main functions and characteristics of the EAs are the following:

Individuals. An individual is a permutation of costumers and the depot several times. The identification number of the depot is one. Every tour (or route) begins with one (cyclic permutations are considered identical). Each tour belongs to a vehicle and the total tour demand is at most *C*. The total tour demands controlled by constraints: the i^{th} tour has a constraint:

g_i(total tour demand by the i*th* tour - C)≤ 0 ($i = 1, 2, \ldots, k$).

Initial population. The P population is generated randomly but the first individuals (e.g. 30 individuals) are generated in the following specific way. We prepare the nearest neighbor list of each costumer, and we rank the lists according to increasing distance from the costumers. For the first individuals we choose the first costumer randomly and next, we choose the closest costumers one after the other based on the nearest neighbor lists. In the second step, tours are cut and separated in the permutation based on the *C* capacity. (Finally it is possible that there will be vehicles without goods, or there will be a vehicle with excess goods.)

Fitness function. The algorithm uses the objective function *f* and the constraints too for the fitness. Let D(x) be the measure of violation of constraints gi (j=1,2,…,m):

$$D(x) = \left(\sum_{j=1}^{m} \max\{g_j(x), 0\}^2 \right)^{1/2} \qquad (1)$$

(If individual *x* is element of the feasible space, then *D(x)=0*). Let us utilize the value D(x) in the optimum search for characterizing the individual *x* in the following way: *x* is better than individual *y* if *D(x)<D(y)*. In case *D(x)=D(y)* we call *x* better than *y* if *f(x)<f(y)*.

Selection operator. In the first stage the descendants are randomly generated. In the second stage the algorithm uses the tournament selection with parameter *5*.

Mutation operator. In the second stage we apply the mutation used max. 4 moves. All moves are the following: move based on the EVL (see Appendix) or a random move. The algorithm uses three different types of moves: swaps two customers, reverses the sub-tour between two customers, or swaps in two different tours randomly chosen sub-tours.

ECM update. It is periodically updated by using the weakest individuals. In the updating procedure we use 20% of the population (see Appendix).

Local search. In the MA we apply five versions of the 2-opt-local-search algorithm one after the other. The local search versions use different moves by two customers (*Figure 1*):

1. reverses the sub-tour between the two customers,
2. swaps the customers,
3. swaps the final sub-tour parts in two different tours which begin with the two customers,
4. swaps the beginning sub-tour parts in two different tours which terminate with the two customers,
5. moves the second customer after the first customer.

All local searches are stochastic: if they could not improve the solution, they accept the wrong solution with a small probability (e.g. 10^{-5}).

Figure 1

Example for the different local search moves

Original tours: 1, 2, 6, 9, 3, 1, 4, 5, 8, 1, 7, 10.

E.g. two customers: 9, 5.

The moves:

1.	1, 2, 6, **5**, 3, 1, 4, **9**, 8, 1, 7, 10.
2.	1, 2, 6, **5, 4, 1, 3, 9**, 8, 1, 7, 10.
3.	1, 2, 6, **9**, **5**, 3, 1, 4, 8, 1, 7, 10.
4.	1, 2, 6, **5**, **8**, 1, 4, **9**, **3**, 1, 7, 10.
5.	1, **4**, **5**, 3, 1, **2**, **6**, **9**, 8, 1, 7, 10.

1. ábra: Példa a helyi kereső eljárások különböző transzformációira.

Reinsertion. In every stage, the algorithm compares the descendent with the most similar solution (The measure of the similarity of the permutation is based on the Hamming distance). If the descendent is better than the former solution, it is replaced with the descendent. If the number of the individuals is less than the population size, the descendent is inserted to the population (e.g. after restart).

Restart strategy. If no new best individual in the population was found for more than 50 generations, the MA begins the second stage with another population. The individuals excepting the best 30% of the population are deleted.

Stopping criteria. The algorithm is terminated if the number of the generations is more than *genlimit*.

EXPERIMENTAL RESULTS

We tested the MA with some benchmark problems of Christofides (C1, C2, C3, C4, C5, C11, C12) and with some benchmark problems of the Vehicle Routing Data Sets (http://branchandcut.org/ index.htm). The MA was implemented in Visual Basic and ran on Intel Core Duo CPU 2.2 GHz with 2 GB RAM.

Parameter selection

Our experience with the earlier algorithm (*Borgulya*, 2008) made easier to choose the values of the parameters. So the used parameters were the following: $t = 90$, $itt = 50$ and $kn = 10$. The maximum number of the generations was 10000 or 20000 depending on the problem.

Comparative results

The results of the MA we show in *Table 1*. Every test problem was run 20 times, and the table shows average results. In the table we give the problem name (*Problem*), the best known solution (BKS), the found best solution (*Best*), the found worst solution (*worst*), the average relative percentage deviation of the solution from the best known solution (*Avg*) and the average running time in seconds to the best solutions (*time*). We got good results by small and medium size problems. The algorithm managed to find the best known solutions in 19 cases from the 26 cases and there are only 4 test problems where the solution is not within 1.0 percent of the best known solutions.

To compare the results of our algorithm we chose other meta-heuristics, e.g. some versions of the ant colony optimization, genetic algorithm and a search procedure. The selection was difficult, because the methods solved only a special set of the benchmark problems: the problems of Christofides (C1, C2, …, C14) or the benchmark problems of the Vehicle Routing Data Sets. We found only one method that solved both benchmark sets and appropriate dates were for the comparison.

To compare the methods based on the Vehicle Routing Data Sets we chose the genetic algorithm from *Tavares et al*. (2003) (GVR) and a cluster-and-search heuristic from *Ganesh et al*. (2006) (CLOVES).

In *Table 2* we can compare the quality of the different results. The table gives the problem name (*Problem*), the relative percentage deviation of the best found solution from the best known solution (*Best*) and the average relative percentage deviation of the solutions from the best known solution (*Avg*.). By small and medium size problems the quality of MA's results is very good. The MA is better in both error percentages (*Best* and *Avg*.) than the GVR and CLOVES. In the case of n>100 we could not compare the MA's results with GVR's and CLOVES's results, because this results are not published.

To compare the methods based on the benchmark problems of Christofides we chose several methods. The methods are the following: the cluster-and-search heuristic from *Ganesh et al.* (2006) (CLOVES), ant colony optimization variants from *Baker et al*. (2003) (B-AS), from *Lin et al*. (2008) (LACO) and from *Bin et al*. (2008) (IACO); simulated annealing from *Osman* (1993) (O_SA), tabu search from *Osman* (1993) (O_TS), tabu search from *Toth et al*. (2003) (T_TS) and a memetic algorithm from *Prins* (2004) (P_MA). In *Table 3* and *4* we compare the quality of the different results. The *Table 3* gives the error percentages (*Best* and *Avg*.) similar way as the *Table 2*. In this table we compare only the P_MA, CLOVES, IACO and MA methods, because the appropriate date is not available by the other methods. The table shows that P_MA and CLOVES have the best results and the average error of our MA is only at the ant colony optimization variant IACO better.

Table 1

Results of MA on benchmark instances

Problem (1)	**BKS**	**MA**			
		Best (2)	*Worst* (3)	*Average* (4)	*Time* (5)
A32k5	784	784	784	0.00	0.6
A54k7	1167	1167	1167	0.00	61
A60k9	1354	1354	1354	0.00	113
A69k9	1159	1164	1170	0.66	214
A80k10	1763	1763	1782	0.53	382
B57k7	1140	1153	1155	1.14	960
B63k10	1496	1496	1504	0.40	850
B78k10	1221	1221	1223	0.08	73
E76k7	682	682	689	0.47	82
E76k8	735	736	738	0.21	98
E76k10	830	835	841	0.80	597
E76k14	1021	1022	1026	0.29	365
F72k4	237	237	237	0.00	12
F135k7	1162	1162	1187	0.79	1050
M101k10	820	820	820	0.00	24
M121k7	1034	1034	1064	1.10	380
M200k17	1296	1309	1320	1.23	8150
P76k4	593	593	595	0.16	51
P101k4	681	681	685	0.14	131
C1	524.61	524.61	524.61	0.00	26
C2	835.26	835.32	844.10	0.90	715
C3	826.14	826.14	832.93	0.42	97
C3	1028.42	1032.68	1046.60	0.91	1278
C5	1291.65	1342.13	1367.21	5.24	4590
C11	1042.11	1042.11	1042.11	0.00	510
C12	819.56	819.56	819.56	0.00	12

1. táblázat: Az MA eredményei benchmark feladatokon.

Probléma(1), Legjobb(2), Legrossazbb(3), Átlag(4), Idő(5)

Table 2

Comparative results on the Vehicle Routing Data Sets

Problem(1)	MA		GVR		CLOVES	
	Best (2)	*Average* (3)	*Best*	*Average*	*Best*	*Average*
A32k5	0.00	0.00	0.00	0.76	0.00	0.00
A54k7	0.00	0.00	0.00	1.64	0.43	2.91
A60k9	0.00	0.00	0.00	1.76	0.30	3.62
A69k9	0.43	0.66	0.51	1.98	0.35	3.45
A80k10	0.00	0.53	0.79	2.88	0.96	1.30
B57k7	1.14	1.14	0.00	0.10	3.47	7.20
B63k10	0.00	0.40	0.00	3.20	0.94	3.01
B78k10	0.00	0.08	0.16	2.75	3.19	5.41
E76k7	0.00	0.47	0.73	3.35	1.17	1.76
E76k8	0.13	0.21	0.40	2.76	0.41	0.68
E76k10	0.60	0.80	1.32	3.20	4.46	4.46
E76k14	0.09	0.29	0.09	2.20	1.08	1.08
Average	0.19	0.38	0.33	2.21	1.39	2.90

2. táblázat: Összehasonlító eredmények a Vehicle Routing Data Sets példáin

Probléma(1), Legjobb(2), Átlag(3)

Table 3

Comparative results on the problems of Christofides

Problem (1)	P_MA		CLOVES		IACO		MA	
	Best (2)	*Avg* (3)	*Best*	*Avg*	*Best*	*Avg*	*Best*	*Avg*
C1	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C2	0.00	0.00	0.00	0.00	0.00	1.62	0.00	0.90
C3	0.00	0.00	0.00	0.00	0.47	2.20	0.00	0.42
C4	0.19	0.31	0.11	0.20	0.00	1.37	0.41	0.91
C5	0.38	0.68	0.57	1.00	1.08	2.34	3.91	5.24
C11	0.00	0.00	0.00	0.00	0.00	0.57	0.00	0.00
C12	0.00	0.00	0.00	0.00	0.00	0.46	0.00	0.00
Average	0.08	0.14	0.10	0.17	0.22	1.22	0.50	1.06

3 . táblázat: Összehasonlító eredmények Christofides problémáinál

Probléma(1), Legjobb(2), Átlag(3)

In *Table 4* we compare only the error percentages of the found best solutions on the benchmark problems of Christofides. By this comparison we found again that the P_MA and CLOVES methods are the best, and the results of our MA are similar with the results of T_TS and LACO.

Table 4

The best results of some methods on the problems of Christofides

Problem (1)	B_AS	O_SA	O_TS	T_TS	P_MA	LACO	IACO	MA	CLOVES
C1	0.00	0.65	0.00	0.00	0.00	0.00	0.00	0.00	0.00
C2	1.08	0.40	1.05	0.06	0.00	0.00	0.00	0.00	0.00
C3	0.75	0.37	1.44	0.05	0.0	0.00	0.47	0.00	0.00
C4	3.22	2.88	1.55	0.46	0.19	1.00	0.00	0.41	0.11
C5	4.03	6.55	3.31	2.07	0.38	1.64	1.08	3.91	0.57
C11	2.22	12.85	0.09	0.07	0.00	0.32	0.00	0.00	0.00
C12	0.00	0.79	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Average (2)	1.61	3.49	1.06	0.47	0.08	0.42	0.22	0.50	0.10

4. Táblázat: Néhány módszer legjobb eredménye Christofides problémáinál

Probléma(1), Átlag(2)

At the end the comparison of the running times was encumbered by the use of various programming languages, operating systems and computers. For comparison we chose the P_MA, LACO, IACO and MA methods and we compared the running time based only on the CPU speed (*Table 5*). This is a draft comparison, but we can see that IACO has the shortest running time and our MA has 10-15 time larger running times than the IACO's running time.

Table 5

Average running times in CPU seconds

Problem (1)	P_MA	LACO	IACO	MA
	(1GHz)	(2.8GHz)	(1GHz)	(2.2GHz)
C1	0.50	38.14	2	26
C2	46.36	118.27	11	715
C3	27.63	293.25	30	97
C4	330.11	701.38	211	1278
C5	1146.52	1844.34	677	4590
C11	17.85	332.77	61	510
C12	2.70	316.02	31	12
Average (2)	225	521	146	1033

5. táblázat: Átlagos futásidők CPU másodpercben.

Probléma(1), Átlag(2)

We can conclude based on the comparison that our MA is the best method on the Vehicle Routing Data Sets and it is the fourth best method on the benchmark problems of Christofides. In the case of CLOVES we can compare the CLOVES and MA on both benchmark sets. Though CLOVES is one of the best methods on the benchmark problems of Christofides, the MA has significantly more accuracy on the Vehicle Routing Data; so we can say that the MA is better than the CLOVES method.

To improve the running time and to reach a faster convergence, we try to improve the algorithm in the future. We will analyze the local search technique, and will try to use other procedures generating the initial solutions.

CONCLUSIONS

In this paper we presented a new EA for the CVRP. We adapted an explicit collective memory method, the extended virtual loser for the CVRP and developed an EA with a new mutation and local search technique. The results show that our algorithm has good quality and has better results as one of the best methods on small and medium size problems.

As future research, we want to improve the effect and the speed of the local search, we want to use other appropriate procedure for the initial solutions and we try to use this extended virtual loser technique by other versions of the vehicle routing problem.

ACKNOWLEDGEMENTS

This research has been supported by the Hungarian National Research Foundation OTKA, under K 68137.

APPENDIX

The principle of the EVL is the following (*Borgulya*, 2006, 2008). Let's consider a generic EA, and suppose that each variable of the individual can have *m* discrete values. (We present only this simple version. If the numbers of the discrete values or the discrete values aren't the same for every variable we can easily modify the following formulas.) Let us notice ECM an *n x m* matrix that stores the relative frequency of the different values of the variables. This matrix is updated through the search procedure using a few of the worst performing individuals.

Let ECM_{ij}^{gen} be the collected relative frequency of the i*th* values on the j*th* position (variable) until the gen*th* generation. We can update the elements of the ECM matrix:

$$ECM_{ij}^{gen+1} = (1-\alpha)ECM_{ij}^{gen} + \alpha \Delta ECM_{ij} \; (e.g.\alpha = 0.2) \tag{2}$$

where ΔECM_{ij} is the relative frequency of the i^{th} value on the position j^{th} based on the worse individuals of the gen^{th} generation and α denotes some relaxation factor.
For the probability of mutating the j*th* variable in individual X we can use the

$$p_j = 1 - \left| \frac{ECM_{x_j j}^{gen}}{\sum_{k=1}^{n} ECM_{kj}^{gen}} - a_j \right| \tag{3}$$

formula, where B is one of the best individuals and *If* $X_j = B_j$ *then* $a_j = 1$ *else* $a_j = 0$.

For the CVRP, the mutation based on the EVL is the following. Let X be a descendant and let B be one of the best individuals. We choose randomly the *jth* and j+1*th* positions (X_j, X_{j+1}), search a better customer for the j+1*th* position. Let U notice the set of the closest customers of X_j (e.g. the first 40 closest customers). We rank the customers increasing based on the distance from customer X_j and select the first $X_z \in U$ customer from the queue with p_{j+1} probability, where

$$p_{j+1} = 1 - \left| \frac{ECM^{gen}_{x_z j+1}}{\sum_{k=1}^{n} ECM^{gen}_{k,j+1}} - a_{j+1} \right| \tag{4}$$

After that e.g. we swap the values of X_{j+1} and X_z.

ECM update

In every kn*th* generation the ECM is updated by using the weakest individuals. In the updating procedure we use 20% of the population.

We observed that the use of the ECM matrix is insufficiently efficient after 50 - 100 generations, the convergence is slow. So we applied a restart strategy for the ECM. After every 20- 50 generation we delete the value of the ECM, and we begin the ECM update with empty matrix.

REFERENCES

Baker, B.M., Ayechew, M.A. (2003): A Genetic Algorithm for the Vehicle Routing Problem. Computers &Operations Research 30. 787-800. p.

Bin, Y., Zhongzhen, Y., Baozhen, Y. (2008): An Improved Ant Colony Optimization for Vehicle Routing Problem. European Journal of Operational Research (In print).

Borgulya, I. (2006): An Evolutionary Algorithm for the biobjective QAP. In: Reusch, B. (ed). Computational Intelligence, Theory and Applications. Advances in Soft Computing, Springer series. 577-586. p.

Borgulya, I. (2008): An Algorithm for the Capacitated Vehicle Routing Problem with Route Balancing. Central European Journal of Operation Research (In print)

Chen, A., Yang, G., Wu, Z. (2006): Hybrid discrete particle swarm optimization algorithm for capacitated vehicle routing problem. J Zhejiang Univ. SCIENCE A 7. 4. 607-614. p.

Ganesh, K., Narendran, T.T. (2007): CLOVES: A cluster-and-search heuristic to solve the vehicle routing problem with delivery and pick-up. European Journal of Operational Research 178. 3. 699-717. p.

Hadjiconstantinou. E., Christofides, N. (1995): A new exact algorithm for the vehicle muting problem based on q-paths and k-shortest paths relaxations. Annals of Operations Research 61, 21-43. p.

Laporte, G. (1992): The vehicle routing problem: an overview of exact and approximate algorithms. European Journal of Operational Research 59(3), 345-358. p.

Lin, S-W., Lee, Z-J., Ying, K-C., Lee, C-Y. (2008): Applying hybrid meta-heuristics for capacitated vehicle routing problem. Expert Systems with Applications. doi:10.1016/j.eswa.2007.11.060

Mazzeo, S., Loiseau, I. (2004): An Ant Colony Algorithm for the Capacitated Vehicle Routing. Electronic Notes in Discrete Mathematics 18. 181–186. p.

Osman, I.H. (1993): Metastrategy Simulated Annealing and Tabu Search Algorithms for the Vehicle Routing Problem. *Ann Oper Res,* 41. 421-51. p.

Prins, C. (2004): A simple and effective evolutionary algorithm for the vehicle routing problem. Computers & Operations Research 31. 1985-2002. p.

Russel, M., Lamont, G.B. (2005): A Genetic Algorithm for Unmanned Aerial Vehicle Rouiting. GECCO'05 Washington, DC, USA, ACM. 1523-1530. p.

Sun, H., Xie, J., Xue, Y. (2005): A Sweep-Based TCNN Algorithm for Capacity Vehicle Routing Problem. Springer-Verlag Berlin Heidelberg, LNCS 3496, 756.761.

Tavares, J., Pereira, F.B., Machado, P., Costa, E. (2003): On the Influence of GVR in Vehicle Routing. Proc. of the 2003 ACM Symposium On Applied Computing, Melbourne, Florida USA, 753-758. p.

Toth, P., Vigo, D. (2003): The granular tabu search and its application to the vehicle-routing problem. INFORMS Journal on Computing, 15, 333–346.

Van Breedam, A. (2001): Comparing descent heuristics and metaheuristics for the vehicle routing problem. Computers & Operations Research 28. 289-315. p

Corresponding author (*Levelezési cím*):

István Borgulya
University of Pécs, Faculty of Business and Economics
H-7622 Pécs, Rákóczi ut 80.
Pécsi Tudományegyetem, Közgazdaságtudományi Kar
7622 Pécs, Rákóczi ut 80.
Tel.: 36-72-501-599, Fax: 36-72-501-553
e-mail: borgulya@ktk.pte.hu