Journal of Computing and Information Technology - CIT 20, 2012, 1, 51–58
doi:10.2498/cit.1002019

An Evolutionary Algorithm Based on Repeated Mutations for Solving the Capacitated Vehicle Routing Problem

Krunoslav Puljić

Department of Mathematics, University of Zagreb, Croatia

An evolutionary algorithm for solving the capacitated vehicle routing problem is described. The algorithm employs repeated mutations in a manner similar to local search. Experiments are presented, where the algorithm has been implemented and tested on some benchmark problem instances.

Keywords: capacitated vehicle routing problem, evolutionary algorithms, repeated mutations, local search, experiments

1. Introduction

The *capacitated vehicle routing problem* (CVRP) is an interesting combinatorial optimization task, which occurs frequently in real-world applications [28]. The problem deals with scheduling a fleet of vehicles to distribute goods between depots and customers. A set of routes for vehicles should be determined, which are in some sense optimal, e.g. the shortest or the cheapest. Certain constraints should be taken into account, such as customer demands and vehicle capacities.

Evolutionary algorithms (EAs) are a popular metaheuristic which tries to solve optimization problems by imitating processes observed in nature [17]. An EA maintains a population of chromosomes where each of them encodes a solution to a particular problem instance. The evolution of those chromosomes takes place through the application of operators and procedures that mimic natural phenomena, such as reproduction, mutation, survival of the fittest, etc.

The CVRP is known to be computationally extremely hard. First of all, it belongs to the class of NP-hard problems [21]. Moreover, it is in fact a generalization and combination of the traveling salesman problem (TSP) and the bin-packing problem (BPP). Consequently, the CVRP turns out to be even harder than the already NP-hard TSP or BPP. We can expect that very small instances of the CVRP can be solved to optimality, but large instances can be solved only approximately. Thus it makes sense to consider applications of metaheuristics, such as EAs, to the CVRP.

In recent years, there have been many attempts to solve the CVRP and similar problems by EAs. Some of the obtained performance results are reported for instance in [1, 4, 7, 11, 13, 15, 19, 22, 24, 29, 30]. General impression is that a pure evolutionary approach is not yet competitive on the CVRP or its variants compared to the other metaheuristics, particularly tabu search [28]. It seems that the presently used chromosomes and evolutionary transformations are not able to capture the full essence of the problem itself. Therefore, many authors have proposed hybrid algorithms [2, 5, 8, 10, 12, 16, 18, 20, 25, 26, 27], where the performance of an EA has been improved by replacing its mutation operator by a traditional *local-search* procedure [3, 6, 21].

The aim of this paper is to present yet another EA for solving the CVRP. Our algorithm is purely evolutionary in the sense that it uses only "genetic" operators for altering chromosomes. The novelty of our approach is something we call *repeated mutations*. Namely, the genetic mutation operator is evaluated many times to

produce similar effects as local search in hybrid algorithms. The paper is organized as follows. Section 2 gives preliminaries about the CVRP and EAs. Section 3 describes some useful building blocks of EAs for solving the CVRP, which have originally been introduced by other authors. Section 4 explains how those building blocks have been arranged in a different way to form our algorithm. Section 5 reports on the results of experiments, where the algorithm has been implemented and tested on a well known library of benchmark problem instances. The final Section 6 gives a conclusion.

2. Preliminaries

The CVRP may be described as the following graph problem. Let $G = (V, A)$ be a complete directed graph, where $V = \{0, 1, 2, \ldots, n\}$ is the vertex set and A is the arc set. Vertices $i = 1, 2, \ldots, n$ correspond to the customers, and vertex 0 corresponds to the depot. A nonnegative cost c_{ij} is assigned to each arc $(i, j) \in A$, and it represents the travel cost spent to go from vertex i to vertex j. Each customer vertex i is associated with a nonnegative demand d_i to be delivered, and the depot 0 has a fictitious demand $d_0 = 0$. A set of K identical vehicles, each with the capacity C, is available at the depot. The CVRP consists of finding a collection of $\leq K$ elementary cycles in G with minimum total cost, such that:

- each cycle visits the depot vertex 0,
- each customer vertex $i = 1, 2, \ldots, n$ is visited by exactly one cycle,
- the sum of the demands d_i of the vertices visited by a cycle does not extend the vehicle capacity C.

By an elementary cycle in G we mean a circular path that does not traverse any vertex more than once. The total cost of a collection of cycles is defined as the sum of the costs c_{ij} of all involved arcs. Obviously, the solution to a CVRP instance specifies an optimal schedule for the vehicles delivering goods from the depot to the customers, so that the demand of each customer is satisfied and no vehicle is overloaded. Each cycle in the solution corresponds to a vehicle route.

In many benchmark problem instances, the vertices from V are associated with points of the plane having given coordinates, and the cost c_{ij} for each arc $(i, j) \in A$ is defined as the Euclidean distance between the two involved points. Such instances belong to a more restricted problem called the *Euclidean symmetric* CVRP (ESCVRP).

The general structure of an evolutionary algorithm is shown in Figure 1. Thus an evolutionary algorithm for solving an optimization problem is a randomized procedure which maintains a population (set) of so-called *chromosomes*. Each chromosome represents a potential solution to the given problem instance, and it is implemented by some data structure. The population is iteratively changed, thus giving a series of population versions which are called generations. It is expected that the best chromosome in the last generation represents a near-optimum solution.

```
void Evolution( ) {
  t = 0;
  initialize P[t];
  evaluate P[t];
  while (!termination_condition) {
    t = t+1;
    select P[t] from P[t-1];
    alter P[t];
    evaluate P[t];
  }
}
```

Figure 1. Structure of an evolutionary algorithm.

The initial population `P[0]` is created by using the `initialize` step. Each chromosome is evaluated during the `evaluate` step to give some measure of its "fitness". The fitness measure is related to the objective function of the original optimization problem. A new generation `P[t]` is created in the `select` step, by choosing the more fit chromosomes from `P[t-1]`. During the `alter` step, some members of `P[t]` undergo transformations by means of "genetic" operators to form new chromosomes. After some number of iterations the algorithm hopefully converges, i.e. it satisfies the `termination_condition`.

There are unary genetic operators, called *mutations*, which create new chromosomes (mutants) by a small random change in a single chromosome. There are also higher order operators called *crossovers*, which create new chromosomes (children) by combining parts from several (usually two) chromosomes (parents). More fit chromosomes should have more chance to take part in crossovers.

3. Basic Components of the Algorithm

In our algorithm, a chromosome is built as proposed in [14], thus it is a list of integers representing a *permutation* of customer vertices. This permutation is interpreted as a large elementary cycle, which is obtained from a CVRP-instance solution by concatenating the vehicle routes and by omitting visits to the depot. Note that the same chromosome can in fact represent many different solutions. Still, we use a unique decoding, which is based on the greedy approach. Thus it is assumed that the first vehicle visits as many customers from the initial part of the chromosome as it is possible according to the vehicle capacity C, the second vehicle serves as many customers as possible from the following part of the chromosome, etc. For instance, let the vehicle capacity be $C = 20$, and suppose that we have $n = 9$ customers whose demands $d_1, d_2, \ldots d_9$ are in turn:

$$2,\ 4,\ 7,\ 5,\ 3,\ 5,\ 8,\ 6,\ 1.$$

Then the chromosome

$$p\ =\ (2\,5\,8\,9\,3\,7\,6\,1\,4)$$

is decoded by the greedy approach to the following three vehicle routes:

$$(2\,5\,8\,9)\ \ (3\,7\,6)\ \ (1\,4).$$

It is assumed, but not explicitly written, that each route starts and ends in the depot.

In our algorithm, we use the crossover operator called *order-crossover* (OX) proposed in [23]. First, two cut points are randomly selected, and the chromosome part located between those cut points on the first parent is assigned to the child. The remaining positions are then filled one at a time, starting after the second cut point, by considering the customer vertices in order found on the second chromosome (wrapping around when the end of the list is reached). For instance, let the two parents and the two cut points "|" be as follows:

$$\begin{aligned} p_1 &= (1\,2\,3\mid 5\,4\,6\,7\mid 8\,9),\\ p_2 &= (4\,5\,2\mid 1\,8\,7\,6\mid 9\,3). \end{aligned}$$

Then the first child c_1 is:

$$c_1\ =\ (2\,1\,8\mid 5\,4\,6\,7\mid 9\,3).$$

If we exchange the roles of the two parents p_1 and p_2, we can obtain the second child:

$$c_2\ =\ (3\,5\,4\mid 1\,8\,7\,6\mid 9\,2).$$

In our algorithm, we use three different mutation operators, called *remove-and-reinsert* (RARM), *swap* (SM) and *invert* (IM), as proposed in [14], [23] and [11], respectively. The operators start in the same way by randomly choosing two positions within the chromosome, and then they proceed in different ways. Namely, RARM removes the vertex (customer) from the first position and reinserts it to the second position, SM swaps the vertices at the two positions, while IM inverts the sequence of vertices between the two positions. For instance, let the starting chromosome be

$$p\ =\ (1\,2\,3\,4\,5\,6\,7\,8\,9).$$

Suppose that the randomly chosen positions are 3 and 7. Then the three operators produce in turn the following three mutants:

$$\begin{aligned} \bar{p}_{\mathrm{RARM}} &= (1\,2\,4\,5\,6\,7\,3\,8\,9),\\ \bar{p}_{\mathrm{SM}} &= (1\,2\,7\,4\,5\,6\,3\,8\,9),\\ \bar{p}_{\mathrm{IM}} &= (1\,2\,7\,6\,5\,4\,3\,8\,9). \end{aligned}$$

Each of the three mutation operators is further on used in two variants called *global* and *local*, respectively, thus making altogether six variants denoted by RARMG, RARML, SMG, SML, IMG and IML. In the global variant, all positions within the chromosome are considered. In the local variant, both the old and the new positions of the moved customers should belong to the route of the same vehicle.

4. Overall Design of the Algorithm

The structure of our evolutionary algorithm follows the outline from Section 2, i.e. it is roughly the same as shown in Figure 1. The initial population `P[0]` is created by producing random permutations according to the uniform probability distribution. A series of generations `P[t]` is produced by applying the crossover operator OX and six mutation variants RARMG, RARML, SMG, SML, IMG, IML. Evaluation of chromosomes is done directly by using the objective function of the CVRP, i.e. the fitness of a chromosome is equal to the total transportation cost of the corresponding solution. The algorithm stops when no further improvement is possible or when a prescribed time limit is reached.

As already mentioned, our algorithm uses mutations in a special way called *repeated mutations*. It means that mutation operators are never applied directly or separately. Instead, they are grouped within special procedures. Such a procedure considers some (or even all) possible mutations of a given chromosome, and finally applies only the best one among considered, i.e. the one that maximally increases the fitness of that chromosome. A separate procedure is built for each of the six variants of mutations.

If the number of considered mutations within a repeated-mutations procedure is small, then those mutations are chosen randomly, and the whole procedure is still randomized. However, in the extreme case when all possible mutations are considered, the procedure becomes deterministic and equivalent to a local search procedure.

Now follows a detailed description of the way how our algorithm transforms one generation of chromosomes `P[t-1]` into the next generation `P[t]`. Selection of `P[t]` at the beginning of iteration `t` is done trivially: all chromosomes from `P[t-1]` are inherited. A little bit more complicated is the step of altering `P[t]` during iteration `t`, which is accomplished in the following way.

- Two "good" chromosomes are chosen from the current population, by using the so-called *tournament selection* [17]. The chosen chromosomes serve as parents for crossover. A prescribed number of children is generated from the same parents with different randomly generated cut points in the OX operator. Only the most fit child is retained, while the others are discarded.
- The only remaining child is improved by repeated mutations. First, the child is improved by applying for instance the RARMG procedure; then the improved child (mutant) is further improved by the same procedure; then the further improved child (mutant's mutant) is even further improved again by the same procedure, etc. When there is no further improvement, we switch to the next procedure, e.g. RARML, ... and so on until all procedures are tried.
- The final improved child is inserted into the current population so that it replaces a "similar" chromosome. Two chromosomes being similar means that relative difference of their fitness measures is below a prescribed threshold. If there is no similar chromosome, then the improved child replaces a "bad" chromosome chosen by a form of tournament selection with *elitism* [17].

As we see, two consecutive generations differ in only one chromosome. Diversity of solutions within one generation is maintained by avoiding to store similar chromosomes.

Note that our algorithm relies on many parameters, which are left free to be adjusted for certain problem instances. Such parameters are: population size, number of children generated from the same parents within one iteration, number of mutations considered by a repeated-mutations procedure, percent of population to be considered by tournament selection, similarity threshold, time limit, etc.

5. Experiments and Results

To enable experimenting, we have developed a C++ implementation of our algorithm. The implementation consists of three C++ classes, whose objects correspond to CVRP instances, chromosomes and populations, respectively. Various components of the algorithm have been realized as methods of those classes, for instance crossovers and mutations are methods of the chromosome class, while tournament selections are methods of the population class. Repeated-mutation procedures have been implemented in a flexible way, so that they can consider either

all possible mutations of a given chromosome, or only a specified number of randomly chosen mutations.

The implemented algorithm has been experimentally evaluated on seven benchmark ES-CVRP instances from the so-called Christofides-Mingozzi-Toth library. The whole library is available at the on-line repository [9]. Table 1 gives some basic properties of the considered test examples, including the costs of their optimal solutions or best solutions known so far.

During a preliminary part of experimenting, we first determined acceptable values for some of the free parameters in our algorithm. Thus the population size was fixed to 30. Similarly, the percentage of population to be considered by tournament selection of good and bad chromosomes was set to 70% and 15%, respectively. The execution time limit was chosen as 10 minutes, 45 minutes or 3 hours, depending on the problem instance size. The similarity threshold was lowered, so that only equal integer fitness measures are treated as similar. On the other hand, the number of mutations considered by a repeated-mutations procedure and the number of children of the same parents were left over for dynamic adjustment.

To accomplish the above mentioned dynamic adjustment, we have implemented an additional procedure, which measures the progress of evolution and changes the two parameters accordingly. The present version of the procedure is quite simple. When the fitness of the best chromosome improves during few iterations, then the number of mutations considered by a repeated-mutations procedure is decreased to enable faster execution of the forthcoming iterations. Otherwise, the number of mutations is increased, thus giving more chance for finding better solutions in the following iterations. The number of children of the same parents is always changed in the opposite direction, in order to partially compensate side-effects of the first change.

Of course, the two dynamic parameters can range only within certain prescribed limits. Thus the number of mutations is always kept between 1% and 20% of the total number of all possible mutations, and it is changed with offsets $+1\%$ and -10%, respectively. Similarly, the number of children of the same parents may vary from 1 to 100 with steps $+5$ and -1. Note that we explore only up to 20% of all possible mutations, so that mutation always remains a randomized process. Consequently, within the presented experiments we actually do not use the possibility of transforming repeated mutations into something equivalent to local search.

The main part of experimenting has been accomplished with the parameters already adjusted as described above. The obtained results are summarized again in Table 1. Each row presents our solutions for one particular problem instance. The shown values can easily be compared with the results obtained by other authors with other metaheuristics. Since our algorithm still relies on random numbers, different runs with the same data and parameters can produce different solutions. Table 1 therefore presents statistical values computed over exactly 10 runs.

As we can see from Table 1, our algorithm solves smaller problem instances to optimality. The obtained best solutions to medium test examples are up to 1.2% worse than the best from

CVRP instance	Number of customers	Number of vehicles	Cost of the best known solution	Costs of our solutions: Avg	Min	Max	StdDev
CMT01	50	5	521	521.7	521	528	2.2
CMT02	75	10	830	848.0	830	859	9.5
CMT03	100	8	815	841.7	817	860	13.5
CMT04	150	12	1015	1124.6	1028	1198	49.2
CMT05	199	16	1289	1390.7	1349	1422	20.5
CMT11	120	9	1034	1140.7	1034	1181	50.0
CMT12	100	10	820	869.4	829	884	16.0

Table 1. Summary of experimental results.

literature. The algorithm is slightly less successful in solving larger instances, where the relative errors range from 1.2% to 4.7%.

Our best result for CMT03, being only 0.25% above the optimum, is visualized in Figure 2. Similarly, Figure 3 depicts the obtained best solution to CMT12 that is 1.1% worse than the corresponding optimal solution. In both figures, the customers and the depot are shown as points of the plane with given coordinates. Also, the vehicle routes forming the solution are plotted by solid lines.

In the course of experimenting, we have also gained some rough impressions about computing times required by our algorithm. On a PC with a 2.4 GHz Pentium processor, the solution

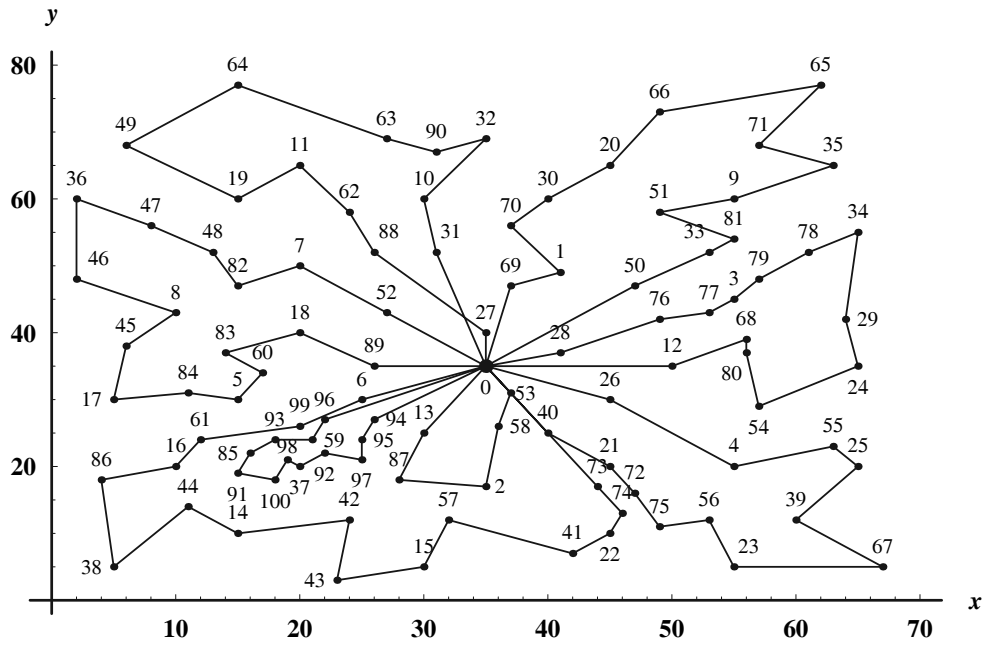

Figure 2. Our best solution to CMT03 (cost: 817).

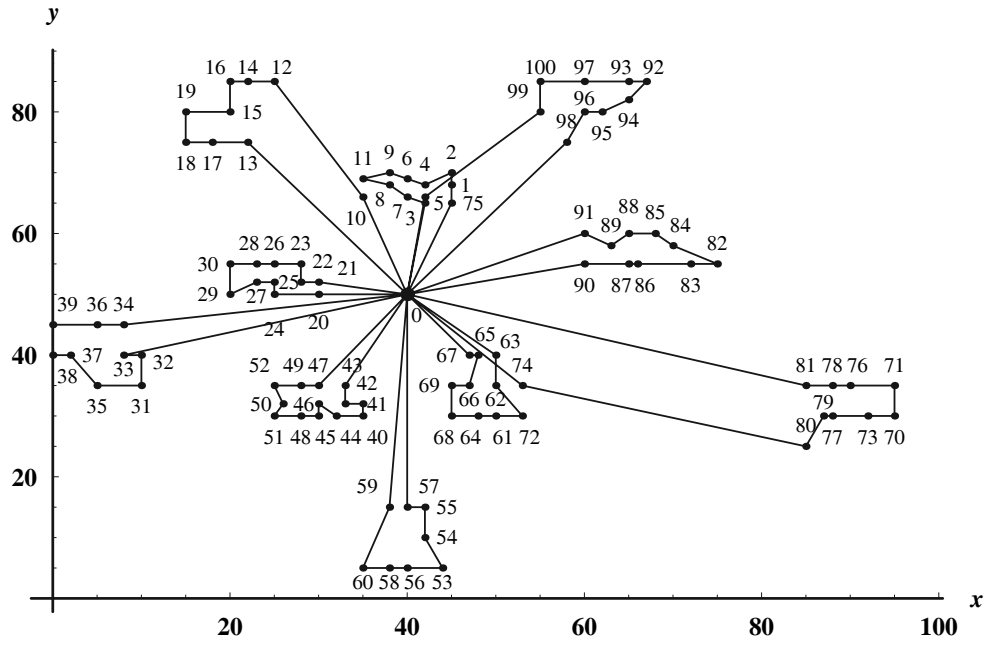

Figure 3. Our best solution to CMT12 (cost: 829).

is usually reached in 5-10 minutes. Smaller problem instances are often solved faster, i.e. in less than 1 minute. For larger instances, it can happen that the algorithm converges only after 1-2 hours.

It is important to note that our experiments always start from completely random populations of chromosomes, and finish with nearly-optimal solutions. This must be taken into account when comparing our results, specially computing times, with other papers. Namely, the other authors usually start with already good solutions obtained by some simpler heuristics, and use evolution only to improve such solutions. Therefore their computations, sometimes running only a few seconds, may seem at the first sight to be more efficient. Still, we believe that our way of measuring performance of an evolutionary algorithm is more accurate, since it excludes influences of other algorithms. Starting with random chromosomes certainly puts the evolutionary process to a more rigorous test of robustness and efficiency.

6. Conclusion

The most distinguished feature of our evolutionary algorithm for solving the vehicle routing problem is its use of repeated mutations. One can say that this feature is very similar to local search. Indeed, through repeated mutations the algorithm in fact explores the neighbourhood of a chosen chromosome consisting of all its mutants.

In spite of this similarity, we believe that our algorithm is still different from hybrid algorithms where the mutation operator is replaced by a local-search procedure. Namely, the presented approach is in a way opposite: we do not abandon mutation, but use it extensively in a manner which is only similar to local search. Thanks to genuine genetic operators and relatively low percentage of mutations considered, our mutation process still bears a large dose of nondeterminism as it is expected from mutation.

The presented results clearly indicate that our evolutionary algorithm is competitive compared to the other metaheuristics, and that our approach based on repeated mutations can assure similar performance results as the approach based on local search. Indeed, on the chosen set of benchmark problem instances, we have obtained nearly-optimal solutions at tolerable computational costs.

Our future plan is to further improve the algorithm by developing a more sophisticated procedure for dynamic adjustment of parameters. Also, we plan to try more appropriate chromosomes or better ways to decode a solution from a chromosome.

References

[1] E. ALBA, B. DORRONSORO , "Solving the vehicle routing problem by using cellular genetic algorithms", in: J. Gottlieb, G. R. Raidl (editors), *Proceedings of the 4th European Conference on Evolutionary Computation in Combinatorial Optimization – EvoCOP 2004, Coimbra, Portugal, April 5–7, 2004*, LNCS Vol 3004, Springer Verlag, Berlin, 11–20, 2004.

[2] E. ALBA, B. DORRONSORO , "Computing nine new best-so-far solutions for capacitated VRP with a cellular genetic algorithm", *Information Processing Letters*, Vol 98, 225–230, (2006).

[3] C. ALABAS-USLU, B. DENGIZ, "A self-adaptive local search algorithm for the classical vehicle routing problem", *Expert Systems with Applications*, Vol 38 (7), 8990–8998, 2011.

[4] B. M. BAKER, M. A. AYECHEW, "A genetic algorithm for the vehicle routing problem", *Computers and Operations Research*, Vol 30, 787–800, (2003).

[5] J. BERGER, M. BARKAOUI, "A new hybrid genetic algorithm for the capacitated vehicle routing problem", *Journal of the Operational Research Society*, Vol 54, 1254–1262, (2003).

[6] P. BEULLENS, L. MUYLDERMANS, D. CATTRYSSE, D. VAN OUDHEUSDEN , "A guided local search heuristic for the capacitated arc routing problem", *European Journal of Operational Research*, Vol 147, 629–643, (2003).

[7] A. BORTFELDT , "A hybrid algorithm for the capacitated vehicle routing problem with three-dimensional loading constraints", *Computers and Operations Research*, online 2011.

[8] O. BRAYSY, G. HASLE, W. DULLAERT , "A multi-start local search algorithm for the vehicle routing problem with time windows", *European Journal of Operational Research*, Vol 159, 586–605, (2004).

[9] B. D. DIAZ, *The VRP Web*, Languages and Computation Sciences Department, University of Malaga, 2011, `http://neo.lcc.uma.es/radi-aeb/WebVRP`

[10] C. DUHAMEL, P. LACOMME, C. PRODHON , "A hybrid evolutionary local search with depth first search split procedure for the heterogeneous vehicle routing problems", Engineering Applications of Artificial Intelligence, online 2011.

[11] H-S. HWANG, "An improved model for vehicle routing problem with time constraint based on genetic algorithm", *Computers and Industrial Engineering*, Vol 42, 361–369, (2002).

[12] A. JASZKIEWICZ, P. KOMINEK, "Genetic local search with distance preserving recombination operator for a vehicle routing problem", *European Journal of Operational Research* Vol 151, 352–364, (2003).

[13] N. JOZEFOWIEZ, F. SEMET, E. TALBI, "An evolutionary algorithm for the vehicle routing problem with route balancing", *European Journal of Operational Research*, Vol 195 (3), 761–769, 2009.

[14] P. LARRANAGA, C. M. H. KUIJPERS, R. H. MURGA, I. INZA, S. DIZDAREVIC, "Genetic algorithms for the travelling salesman problem: a review of representations and operators", *Artificial Intelligence Review*, Vol 13, 129–170, (1999).

[15] S. LIU, W. HUANG, H. MA, "An effective genetic algorithm for the fleet size and mix vehicle routing problems", *Transportation Research Part E: Logistics and Transportation Review*, Vol 45, (3), 434–445, 2009.

[16] MESTER D., BRAYSY O., "Active guided evolution strategies for large–scale capacitated vehicle routing problems", *Computers and Operations Research*, Vol. 34, 2964–2975, (2007).

[17] MICHALEWICZ Z., *Genetic Algorithms + Data Structures = Evolution Programs*, Third Edition, Springer, New York, 1995.

[18] M. J. W. MORGAN, C. L. MUMFORD, "Capacitated vehicle routing – perturbing the landscape to fool an algorithm", in: D. Corne, Z. Michalewicz (editors), *Proceedings of the IEEE Congress on Evolutionary Computation (CEC), Edinburgh, Scotland, September 2–5, 2005*, IEEE Press, Piscataway, New Jersey, 2271–2277, 2005.

[19] NAZIF H., LEE L.S., "Optimised crossover genetic algorithm for capacitated vehicle routing problem", *Applied Mathematical Modelling*, online 2011

[20] ULRICH NGUEVEU S., PRINS C., CALVO R., "An effective memetic algorithm for the cumulative capacitated vehicle routing problem", *Computers and Operations Research*, Vol 37 (11), 1877–1885, 2010.

[21] PAPADIMITRIOU C.H., STEIGLITZ K., *Combinatorial Optimization – Algorithms and Complexity*, Dover, Mineola, New York, 1998.

[22] F. B. PEREIRA, J. TAVARES, P. MACHADO, E. COSTA, "GVR – a new genetic representation for the vehicle routing problem", in: O'Neil M. et al. (editors), *Proceedings of the 13th Irish International Conference on Artificial Intelligence and Cognitive Science – AICS 2002, Limerick, Ireland, September 12–13, 2002*, LNAI Vol 2464, Springer Verlag, Berlin, 95–102, 2002.

[23] P. PONGCHAROEN, D. J. STEWARDSON, C. HICKS, P. M. BRAIDEN, "Applying designed experiments to optimize the performance of genetic algorithms used for scheduling complex products in the capital goods industry", *Journal of Applied Statistics*, Vol 28, No 3&4, 441–455, (2001).

[24] C. PRINS, "A simple and effective evolutionary algorithm for the vehicle routing problem", *Computers and Operations Research*, Vol 31 (2004), 1985–2002.

[25] C. PRINS, "Two memetic algorithms for heterogeneous fleet vehicle routing problems", *Engineering Applications of Artificial Intelligence*, Vol 22 (6), 916–928, 2009.

[26] K. C. TAN, L. H. LEE, K. OU, "Artificial intelligence heuristics in solving the vehicle routing problems with time window constraints", *Engineering Applications of Artificial Intelligence* Vol 14, 825–837, (2001).

[27] J. TAVARES, F. B. PEREIRA, P. MACHADO, E. COSTA, "Crossover and diversity – a study about GVR", in: Barry A.M. (editor), *Proceedings of the Bird of a Feather Workshops, 2002 Genetic and Evolutionary Computation Conference – GECCO 2003, Chicago, Illinois, USA, July 12–16, 2003*, AAAI Press, Menlo Park, California, 27–33, 2003.

[28] P. TOTH, D. VIGO (EDITORS), *The Vehicle Routing Problem*, SIAM Monographs on Discrete Mathematics and Applications, SIAM, Philadelphia, 2002.

[29] Z. URSANI, D. ESSAM, D. CORNFORTH, R. STOCKER, "Localized genetic algorithm for vehicle routing problem with time windows", *Applied Soft Computing*, Vol 11 (8), 5375–5390, 2011.

[30] ZHU K.Q., "A diversity-controlling adaptive genetic algorithm for the vehicle routing problem with time windows", in: Chen I.R. (editor), *Proceedings of 15-th IEEE International Conference on Tools with Artificial Intelligence – ICTAI'03, Sacramento, California, USA, November 3–5, 2003*, IEEE Computer Society, Washington, 176–183, 2003.

Received: December, 2011
Accepted: February, 2012

Contact address:
Krunoslav Puljić
Department of Mathematics
University of Zagreb
Bijenička cesta 30
10000 Zagreb, Croatia
e-mail: nuno@math.hr

KRUNOSLAV PULJIĆ graduated from the Department of Mathematics, Faculty of Natural Sciences, University of Zagreb in 1999. He has been working as a research assistant at the same Department since 2000. He obtained the M.Sc. degree in 2004 with the work on "The evolutionary algorithms for the vehicle routing problem". Within the "DataGrid" project of the EU he spent some time at the CERN Institute in Geneva. In 2009 he received his Ph.D. degree with the work on "The distributed evolutionary algorithms for the vehicle routing problem". His professional and scientific interest is in combinatorial optimization, parallel and distributed algorithms, metaheuristics, web programming and database systems.