AN IMPROVED GENETIC ALGORITHM FOR SOLVING THE SELECTIVE TRAVELLING SALESMAN PROBLEM ON A ROAD NETWORK

Anna Piwońska

Faculty of Computer Science, Bialystok University of Technology, Białystok, Poland

Abstract: The Selective Travelling Salesman Problem (STSP) is a modified version of the Travelling Salesman Problem (TSP) where it is not necessary to visit all vertices. Instead of it, with each vertex a number meaning a profit is associated. The problem is to find a cycle which maximizes collected profit but does not exceed a given cost constraint. A direct application of the STSP, e.g. in Intelligent Transportation Systems, is finding an optimal tour in road networks. However, while the classic STSP is defined on a complete graph, a road network is in general not complete and often has a rather sparse edge set. This paper presents the STSP defined on a road network (R-STSP). Since the R-STSP is NP-hard, the improved genetic algorithm (IGA) is proposed which is the next version of our previous GA. The main aim of this paper is to investigate the role of the deletion mutation in the performance of the IGA.

Keywords: travelling salesman problem with profits, genetic algorithm, deletion mutation

1. Introduction

The TSP is an NP-hard problem studied in operations research and computer science [1]. The problem is formulated as follows. Given a list of cities and their pairwise distances, the task is to find the shortest possible tour that visits each city exactly once.

While in the TSP a salesman needs to visit all cities, some variant problems enforce to visit only selected ones, depending on a profit gained during visiting. This feature gives rise to a number of problems which are called in the literature the Travelling Salesman Problem with Profits (TSPwP) [3]. In this group of problems, usually one of n cities has a special meaning - it is considered as a depot. In one version of the TSPwP described in the literature, the problem is to find an elementary cycle starting from a depot, that maximizes collected profit such that the tour length does not exceed a given constraint. This problem appears under the name "the orienteering problem" [13] or "the selective TSP" [12]. Since the TSPwP belongs to the class of NP-hard problems, many metaheuristic approaches have been proposed in the literature e.g. tabu search [6], ant colony optimization [8], genetic algorithms [7], neural networks [15] and harmony search [5].

The R-STSP was first formulated in the author's previous paper [11] and is some modification of the problem described above, with two important assumptions introduced. Firstly, a graph may not be complete: not every pair of vertices must be connected by an edge. In fact, a road network is in general not complete and often has a rather sparse edge set. This issue was considered by Fleischmann [4], who introduced the notion of the Travelling Salesman Problem on a Road Network (R-TSP). Despite the fact that we can transform such a not complete graph in a complete one by introducing dummy edges, such an approach seems to be ineffective. It increases the search space and has a direct impact on the execution time of the algorithm.

The second assumption is that we allow repeated visiting of a given city: a cycle we are looking for may not be an elementary one. This assumption results from the fact that a graph may not be complete. Moreover, in the real world returns are natural: one may want to travel using repeated fragments of a route. However, while a salesman can be in a given city more than once, a profit is realized only during first visiting. This assumption prevents from generating routes in which a city with the highest profit is continually visited while others are not. With these additional assumptions, the problem is more realistic and could have practical applications in logistics and shipping.

In [11] the GA with special crossover and mutation operators was proposed. In this paper we present the improved version of our previous GA (IGA) in which a new mutation is introduced as an additional operator, namely the deletion mutation. Moreover, the mutation which inserts a city to a tour is improved. The main aim of the paper is to investigate the role of the deletion mutation in the performance of the IGA. Experiments conducted on the real network of 160 cities in eastern and central Poland show that the deletion mutation significantly improves the quality of solutions generated by the IGA.

The rest of the paper is organized as follows. Section 2. presents formal definition of the R-STSP. Section 3. describes the details of the IGA, with particular focus on both mutations. Experimental results are reported in Section 4.. The last section contains conclusions and some remarks about future work.

2. Definition of the R-STSP

A network of cities is represented by a weighted, undirected graph $G = \langle V, E \rangle$. $V = \{1, 2, ..., n\}$ is a set of n vertices and E is a set of edges. Each node in G corresponds to a city in a network. Vertex 1 has a special meaning and is interpreted as the depot. An undirected edge $\{i, j\} \in E$ means that there is a possibility to travel from the city i to the city j (and vice versa). The weight d_{ij} of the edge $\{i, j\}$ denotes a distance between cities i and j. Additionally, each vertex has assigned a non-negative number meaning a profit. Let $F = \{f_1, f_2, ..., f_n\}$ be a vector of profits for all vertices. An important assumption is that a profit is realized only during first visiting of a given vertex. The exemplary graph is shown in Fig. 1.

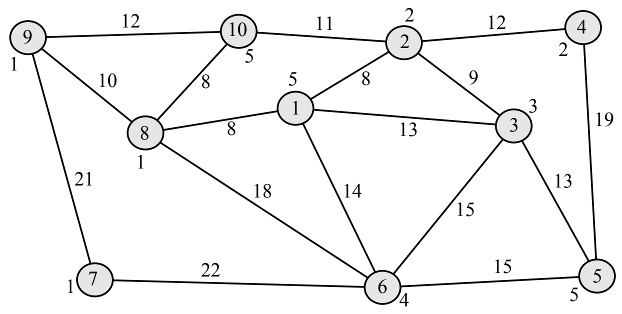

Fig. 1. A graph representation of a network of cities; the d_{ij} values are marked on the edges, the f_i values are marked next to the nodes

The R-STSP can be formulated as follows. The goal is to find a cycle starting from the depot that maximizes collected profit such that the tour length does not exceed a given constraint c_{max}.

3. The IGA for the R-STSP

During the last years several methods were proposed for handling constraints in GAs. Most of them are based on the concept of a penalty function [9]. In the IGA, as well as in our previous GA, a different approach is proposed. Due to the special way of generating the initial population and using specialized operators, the IGA searches solutions only in the feasible region.

The difference between our previous GA and the IGA is in a mutation operator: the IGA uses improved, heuristic mutation (hereafter called the insertion mutation) and as an additional operator, the deletion mutation. The deletion mutation is presented in the IGA in two forms: the first tries to delete from a tour repeated cities (the deletion mutation I) and the second tries to delete from a tour a random city (the deletion mutation II).

The pseudocode of the IGA is presented as Algorithm 1.

```
Algorithm 1: the IGA for the R-STSP
Begin
 generate the initial population of individuals of size P;
 compute fitness function for each individual;
 for i:=1 to ng do
  begin
   select the population i from the population i-1
   by means of tournament selection
   with the group size equal to t_size;
   divide population into disjoint pairs;
   cross each pair if possible;
   apply the deletion mutation I to each individual;
   apply the deletion mutation II to each individual;
   apply the insertion mutation to each individual;
   compute fitness function for each individual;
  end
 choose the best individual from the population as the result;
End
```

When we want to solve a given problem by a GA, first we must encode a solution into a chromosome. Like in the most TSP-based problems, we decide to use the path representation [10]. In this approach, a tour is encoded as a sequence of vertices. For example, the tour 1 - 2 - 3 - 5 - 6 - 1 is represented by the sequence (1 2 3 5 6 1).

The IGA starts with a randomly generated population of P solutions. The initial population is generated in a special way. Starting at the depot, we choose a city to which we can travel from the depot with equal probability. We add the distance between the depot and the chosen city to the current tour length. If the current tour length is not greater than $c_{max}/2$, we continue, but instead of starting at the depot, we start at the chosen city. We again randomly select a city, but this time we exclude from the set of possible cities the city from which we have just arrived (the last city in a partial tour). This assumption prevents from continual visiting a given city but is relaxed if there is no possibility to choose another city. If the current tour length is greater than $c_{max}/2$, we reject the last city and return to the depot the same way. In this case the tour length does not exceed c_{max} therefore the constraint imposed by the problem is preserved. It is easy to observe that such an idea of generating the initial population causes that individuals are symmetrical in respect of the middle city in the tour. However, experiments show that the IGA quickly removes these symmetries.

The next step is to evaluate individuals in the initial population by means of the fitness function. The fitness of a given individual is equal to collected profit under the assumption that a profit is gained only during first visiting of a given vertex.

Subsequently the IGA starts to improve the initial population through repetitive application of selection, crossover and mutation. In our experiments we use tournament selection: we select t_{size} individuals from the current population and determine the best one from the group. The winner is copied to the next population and the whole tournament group is returned to the old population.

In the first step of crossover, individuals are randomly coupled. Then, each couple is tested if crossover can take place. If two parents do not have at least one common gene (with the exception of the depot), crossover cannot be done.

Fig. 2 illustrates how the genetic material is swapped during crossover. First we randomly choose one common gene from the set of common genes in both parents: P1 and P2 (we exclude the depot from this set). This gene will be the crossing point. Then we exchange fragments of tours from the crossing point to the end of the chromosome in two parent individuals. If offspring individuals (O1 and O2) preserve the constraint c_{max}, they replace their parents in the new population. If one offspring individual does not preserve the constraint c_{max}, its position in the new population is occupied by fitter parent. If both children do not preserve the constraint c_{max}, they are replaced by their parents in the new population.

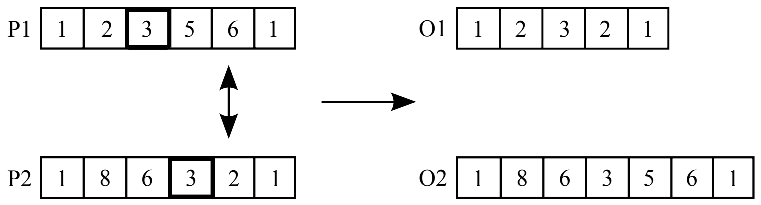

Fig. 2. An example of crossover operator

The last genetic operator is mutation. The role of mutations in a GA is to increase genetic diversity in a population [2]. In this paper we present two kinds of mutation adjusted to our problem: the deletion mutation and the insertion mutation.

The idea of using the deletion mutation comes from genetics. In living organisms, the deletion mutation takes place when a part of a chromosome or sequence of DNA is missing. In our problem, the deletion mutation I tries to eliminate from a chromosome every appearance of repeated cities (with the exception of the first and the last gene which are established). They can appear in an individual after crossover (Fig. 2) and are also presented in chromosomes in the initial population. Such cities do not influence fitness function value and an attempt of removing them from a chromosome should be undertaken (obviously, this mutation leaves in a chromosome one appearance of a given city). Let us assume that a partial tour is $...i, j, k....$ The city j can be removed from a chromosome if there is the edge $\{i,k\} \in E$. The deletion mutation I is described as Algorithm 2.

```
Algorithm 2: the deletion mutation I
Begin
 for i:=2 to chromosome_length-1 do
  if a city in the position i is presented in a chromosome
  more than once then
   try to remove this city from a chromosome;
End
```

After removing repeated cities from chromosomes, a population undergo the deletion mutation II which tries to delete from each chromosome a random city (with the exception of the first and the last gene). The deletion mutation II is described as Algorithm 3.

```
Algorithm 3: the deletion mutation II
Begin
 choose a random city in a chromosome;
 try to remove this city from a chromosome;
End
```

The last mutation is the insertion mutation. The principle of operation of this mutation is presented as Algorithm 4.

```
Algorithm 4: the insertion mutation
Begin
 randomly choose in a chromosome two neighboring cities i and j;
 create a set S of possible cities which are not presented
 in a chromosome and which can be inserted between i and j;
 sort cities in S decreasingly according to profits;
 if profits are equal sort increasingly according to
 increment of a total length of a tour
 after inserting a given city;
 while S is not empty do
  begin
   take the first city in S, namely k, and remove k from S;
   if inserting city k between cities i and j does not violate
   the constraint c_max then
    begin
     insert city k between cities i and j;
     break {a chromosome is mutated};
    end;
  end;
End
```

In the insertion mutation, a city is inserted into a tour only if it has not been inserted in a tour yet. The reason behind this is that inserting a city so far not presented in a tour improves fitness of a given individual. Moreover, the best city from the set of all possible cities is inserted. The best means the city with the highest profit and (if profits of cities are equal) the one city that will cause the least increment of the total length of a tour. If no city can be inserted into a chromosome without violating c_{max}, the mutation is not performed.

It is important that the insertion mutation is performed after both deletion mutations. If deletion mutations remove some cities from a tour, the total length of a tour decreases. This way the probability of successful application of the insertion mutation increases.

The IGA terminates when ng generations is reached.

4. Experimental Results

Computer experiments were performed on network of 160 cities in Poland. Twenty cities from each of eight provinces of eastern and central Poland were chosen. The network used in experiments was created from a real map, by including to a graph main segments of roads. Profits associated with cities were determined according to a number of inhabitants in a given city. As the depot, the capital of Poland, Warsaw, was chosen. Data concerning this network are accessible on the website http://piwonska.pl/p/research/ in two text files: cities.txt and distances.txt.

The line number i in both files represents information about city number i. The number of lines in each file is equal to n. Format of the line i in cities.txt file is: i name-of-the-city f_i. Format of the line i in distances.txt file is: $i\ j_1\ d_{ij_1}\ ...\ j_k\ d_{ij_k}$, where $j_1\ ...\ j_k$ are numbers of cities connected to the city i and $d_{ij_1}\ ...\ d_{ij_k}$ are distances between them.

We performed experiments for 11 c_{max} values: $\{500, 600, 700, ..., 1500\}$. We set $t_{size} = 3$ and $P = 300$. Higher values of P did not lead to an improvement in results. Since the algorithm converges quickly, setting $ng = 100$ was enough to obtain good results.

To investigate the role of the deletion mutation, three series of experiments were performed. In the first case, we turn off both deletion mutations. The only mutation performed during the IGA was the insertion mutation. In the second case, the deletion mutation I was turn on and in the third case the IGA performed both deletion mutations. Ten runs were performed for each case, what resulted in thirty runs for each c_{max}. Tab. 1 presents the average, the best and the worst (in brackets: the best; the worst) collected profits.

Table 1. The average, the best and the worst collected profits from 10 runs of the IGA

c_{max}	insertion mutation	insertion mutation + deletion mutation I	insertion mutation + deletion mutation I and II
500	55.8 (60; 48)	56.7 (60; 53)	56.8 (60; 50)
600	66.4 (73; 57)	67.0 (73; 60)	69.4 (73; 65)
700	72.9 (80; 66)	76.3 (80; 71)	77.7 (80; 71)
800	81.4 (86; 77)	84.1 (87; 80)	84.9 (87; 82)
900	88.4 (93; 81)	91.9 (101; 88)	92.3 (101; 90)
1000	94.4 (104; 85)	97.5 (109; 94)	105.1 (114; 95)
1100	103.0 (112; 95)	107.5 (117; 100)	113.1 (122; 106)
1200	107.0 (118; 99)	111.6 (119; 101)	120.5 (129; 109)
1300	119.5 (136; 106)	125.8 (141; 114)	133.3 (143; 110)
1400	126.9 (139; 117)	131.6 (146; 121)	139.7 (151; 131)
1500	131.3 (152; 115)	139.0 (158; 124)	147.4 (160; 133)

One can see that the average profits are the worst in the case of both deletion mutations turned off. Turning on the deletion mutation I improves the results, however the best improvement is gained when both deletion mutations are turned on. This effect is observed for all c_{max} values. In general, as c_{max} increases, the differences between the IGA without deletion mutations and the IGA with both deletion mutations start to deepen. The largest improvement rate is observed for $c_{max} = 1200$ and is equal to 12.6%.

Only for $c_{max} = \{500, 600, 700\}$ the best profits for the IGA without deletion mutations are the same as the best profits for the IGA with both deletion mutations. Starting from $c_{max} = 800$ the best results obtained by the IGA without deletion mutations are always worse than the best results in the case of both deletion mutations turned on.

Looking at the best and the worst collected profits one can see that all algorithms are not stable: the differences between the best and the worst results are relatively large. This issue will be investigate in our future research.

The chromosomes of the best individuals found by the IGA with both deletion mutations are presented in Tab. 2.

Table 2. The best individuals found by the IGA with the insertion mutation, the deletion mutation I and the deletion mutation II

c_{max}	profit	total distance	chromosome
500	60	487	1 11 72 71 70 69 79 61 80 62 66 64 63 73 15 16 1
600	73	592	1 10 5 6 12 70 69 79 61 80 62 66 68 64 63 73 72 71 11 1
700	80	684	1 11 71 72 73 67 63 64 68 66 65 62 80 61 79 69 70 12 5 10 1
800	87	798	1 10 5 6 12 70 69 79 61 80 62 65 66 64 68 119 117 118 67 63 73 72 71 11 1
900	101	898	1 16 18 97 91 93 81 92 90 89 107 102 106 105 118 67 63 64 66 62 80 61 79 69 70 71 72 11 1
1000	114	992	1 10 5 12 6 11 71 72 74 61 80 62 66 68 64 63 67 118 105 106 102 107 109 131 132 89 90 91 97 18 16 1
1100	122	1100	1 11 71 72 74 61 79 80 62 66 68 64 63 67 118 105 106 102 107 109 131 132 133 88 89 90 91 92 81 93 95 94 98 16 1
1200	129	1196	1 10 5 6 11 71 72 73 63 74 61 80 62 66 64 68 119 103 115 101 120 106 105 106 102 107 109 131 132 133 88 89 81 92 91 97 18 16 13 1
1300	143	1297	1 13 14 98 94 93 81 89 107 109 132 131 111 110 114 101 120 102 106 105 118 67 63 64 68 66 62 80 61 79 69 70 12 6 11 71 72 73 15 16 1
1400	151	1392	1 10 5 2 12 70 71 11 72 74 61 79 80 62 66 68 64 63 67 118 105 106 102 107 109 131 132 133 88 89 90 91 92 81 93 95 94 96 14 13 1
1500	160	1498	1 10 5 12 70 71 11 72 73 74 61 62 80 79 69 78 65 66 68 64 63 67 118 105 104 101 120 106 102 107 109 131 132 133 88 89 81 93 91 97 94 98 14 13 1

One can see that there are almost no repeated cities in these chromosomes. The exception is the individual for $c_{max} = 1200$ in which the city 106 occurs twice. However, this city can not be removed from the chromosome due to the lack of adequate edges.

As an example, Fig. 3 presents the best run of the IGA with both deletion mutations for $c_{max} = 1500$. One can see that the IGA converges quickly: before 50th generation. The potential reason of this problem could be the fact that the crossover and the mutation operators often can not produce modified individuals (e.g. due to the constraint c_{max}). For example, detailed analysis shows that in a typical run of the IGA

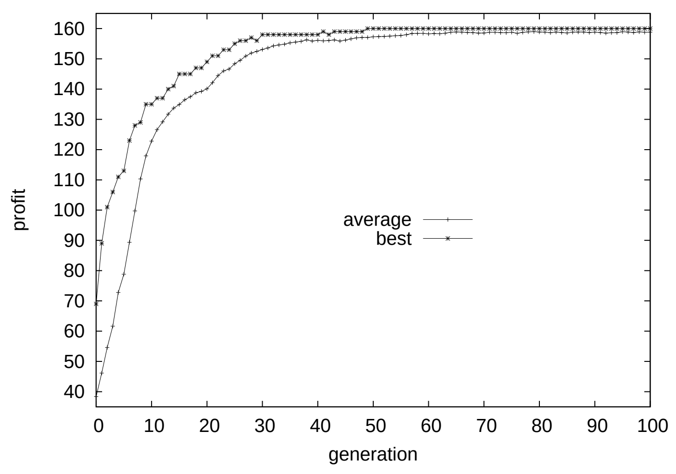

Fig. 3. The IGA run with the insertion mutation and both deletion mutations for $c_{max} = 1500$

on average only one out of every three individuals undergoes the deletion mutation II. That is why the population quickly fills with the copies of the same individual.

5. Conclusions

In this paper we presented the IGA for solving the R-STSP. Three mutation operators adjusted to the problem were proposed: the insertion mutation, the deletion mutation which tries to delete from a tour repeated cities (the deletion mutation I) and the deletion mutation which tries to delete from a tour a random city (the deletion mutation II).

The aim of the work was to investigate the role of deletion mutations on the performance of the IGA. Computer experiments conducted on the real network of 160 cities in Poland indicated that deletion mutations (as additional operators) improved the quality of obtained results.

Since the R-STSP is defined on the graph which in general is not complete, well known recombination operators for the classic TSP, e.g. inversion, mutual swap, cannot be used. Thus there is a need for designing specialized genetic operators adjusted to this problem. Also, there is a problem of premature convergence of the IGA which should be tackled. Another subject of future research is to investigate the effect of introducing elitism and niching techniques into the IGA.

The issue which must be carefully studied is another approach to handling constraint c_{max}. We plan to implement the IGA with the penalty function and compare obtained results. Also, other heuristics will be tested, e.g. ant colony optimization, tabu search or harmony search.

An important issue is reusing discovered solutions. Optimal tours obtained for a given c_{max} can be potentially reused when solving problems for larger c_{max}. We think that such an idea could improve performance of the IGA. It will be the successive subject of our future research.

References

[1] Applegate D.L., Bixby R.E., Chvátal V., Cook W.J., The Traveling Salesman Problem: A Computational Study. Princeton University Press, 2006.

[2] De Falco I., Della Cioppa A., Tarantino E., Mutation-based genetic algorithm: performance evaluation. Applied Soft Computing, vol. 1 (4), Elsevier, pp. 285-299, 2002.

[3] Feillet D., Dejax P., Gendreau M., Traveling Salesman Problems with Profits, Transportation Science, Vol. 39, No. 2, pp. 188-205, 2005.

[4] Fleischmann B., A new class of cutting planes for the symmetric travelling salesman problem. Mathematical Programming, vol. 40, pp. 225-246, 1988.

[5] Geem Z.W., Tseng Ch.-L., Park Y., Harmony Search for Generalized Orienteering Problem: Best Touring in China. LNCS, vol. 3612, Springer, pp. 741-750, 2005.

[6] Gendreau M., Laporte G., Semet F., A tabu search heuristic for the undirected selective travelling salesman problem. European Journal of Operational Research, vol. 106 (2-3), Elsevier, pp. 539-545, 1998.

[7] Jozefowiez N., Glover F., Laguna M., Multi-objective Meta-heuristics for the Traveling Salesman Problem with Profits. Journal of Mathematical Modelling and Algorithms, vol. 7 (2), pp. 177-195, 2008.

[8] Liang Y.-C., Smith A.E., An ant colony approach to the orienteering problem. Technical report. Department of Industrial and Systems Engineering, Auburn University, Auburn, USA, 2001.

[9] Michalewicz Z., Genetic Algorithms, Numerical Optimization and Constraints. Proceedings of the 6th International Conference on Genetic Algorithms, Pittsburgh, July 15-19, pp. 151-158, 1995.

[10] Michalewicz Z., Genetic Algorithms+Data Structures=Evolution Programs. WNT, Warsaw, 1996.

[11] Piwonska A., Genetic algorithm finds routes in travelling salesman problem with profits. Zeszyty naukowe Politechniki Bialostockiej. Informatyka (in Polish), Vol. 5, pp. 51-65, 2010.

[12] Qin H., Lim A., Xu D., The Selective Traveling Salesman Problem with Regular Working Time Windows. Studies in Computational Intelligence, Vol. 214, pp. 291-296, 2009.

[13] Sevkli Z., Sevilgen F.E., Variable Neighborhood Search for the Orienteering Problem. LNCS, Vol. 4263, pp. 134-143, 2006.

[14] Souffriau W., Vansteenwegen P., Berghe G.V., Oudheusden D.V., A Greedy Randomised Adaptive Search Procedure for the Team Orienteering Problem. In proceedings of EU/MEeting 2008 - Troyes, France, 2008.

[15] Wang Q., Sun X., Golden B.L., Jia J., Using artificial neural networks to solve the orienteering problem. Annals of Operations Research, vol. 61, Springer, pp. 111-120, 1995.

ULEPSZONY ALGORYTM GENETYCZNY DO ROZWIĄZANIA SELEKTYWNEGO PROBLEMU KOMIWOJAŻERA W SIECI DROGOWEJ

Streszczenie Selektywny problem komiwojażera (STSP) jest zmodyfikowaną wersją problemu komiwojażera (TSP), w której nie jest konieczne odwiedzenie wszystkich wierzchołków. Zamiast tego, z każdym wierzchołkiem związana jest liczba oznaczająca zysk. Problem polega na znalezieniu cyklu w grafie, który maksymalizuje zysk, ale którego koszt nie przekracza zadanego ograniczenia. Bezpośrednim zastosowaniem problemu STSP, np. w Inteligentnych Systemach Transportowych, jest odnajdywanie optymalnej trasy w sieci drogowej. Jednakże, podczas gdy klasyczny problem STSP jest zdefiniowany na grafie zupełnym, sieć drogowa zwykle nie jest grafem pełnym i często ma rzadki zbiór krawędzi. Artykuł przedstawia problem STSP zdefinowany w sieci drogowej (R-STSP). Ponieważ R-STSP jest NP-trudny, zaproponowano ulepszony algorytm genetyczny (IGA), który jest rozszerzoną wersją poprzedniego algorytmu genetycznego. Głównym celem artykułu jest zbadanie roli mutacji usuwającej w w jakości wyników IGA.

Słowa kluczowe: selektywny problem komiwojażera na sieci drogowej, algorytm genetyczny, mutacja usuwająca

Artykuł zrealizowano w ramach pracy badawczej S/WI/2/2008