GENETIC ALGORITHM FINDS ROUTES IN TRAVELLING SALESMAN PROBLEM WITH PROFITS

Anna Piwońska[1]

[1]Faculty of Computer Science, Bialystok University of Technology, Białystok, Poland

Abstract: Travelling salesman problem with profits is a version of a classic travelling salesman problem 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 in a graph which maximizes collected profit but does not exceed a given cost constraint. This problem is NP-hard. Additional assumptions to this problem were proposed in the paper. We assumed that a graph may not be a complete graph. Moreover, repeated visiting of a given vertex is allowed, however with an assumption that a profit is realized only during first visiting. With these additional assumptions, the problem is more real-life and could have applications in logistics and shipping. To solve the problem, a genetic algorithm with special operators was proposed. The algorithm was tested on networks of cities in some voivodeships of Poland, obtaining very good results.

Keywords: travelling salesman problem with profits, genetic algorithm

1. Introduction

Travelling salesman problem (TSP) is a well known combinatorial optimization problem, studied in operational research and computer science. The problem is formulated as follows. Given the set of n cities and distances between each pair of them, find a closed tour (a cycle) through all cities that visits each city only once and is of minimum length. The problem is known to be NP-hard, therefore many heuristics have been proposed to find near-optimal solutions [9].

While in a classic 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 travelling salesman problem with profits (TSPwP) [2,7]. In this group of problems, usually one of *n* cities has a special meaning - it is considered as a depot. In a one version of TSPwP described in the literature, the problem is to find an elementary cycle (i.e., a cycle such that each vertex is visited at most once) starting from a depot, that maximizes collected profit such that the tour length does not exceed a given constraint. This problem is also known in the literature under the name "the orienteering problem" (OP) [15] or "the selective TSP" [14] and will be considered in the paper. Like TSP, TSPwP belongs to the class of NP-hard problems. Due to high time complexity of the TSPwP, many metaheuristic approaches have been proposed in the literature, such as tabu search [5,13,3], ant colony optimization [10], genetic algorithms [1,7], neural networks [17] and harmony search [4].

In this paper, additional assumptions to this problem were proposed. Firstly, we assumed that a graph may not be a complete: not every pair of vertices must be connected by an edge. Looking at a map one can see that in the real world cities in some region are not all connected to each other. Despite of 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 would result in a lot of unnecessary data introduced to the problem.

The second assumption is that we allow repeated visiting of a given vertex: a cycle we are looking for may not be an elementary one. This assumption results from the fact that a graph is not complete. 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 finding routes in which a city with a highest profit is continually visited while others are not. With these additional assumptions, the problem is more real-life and could have applications in logistics and shipping. To find routes with optimal profit, a genetic algorithm (GA) with special operators was used. The method was implemented and tested on networks of cities in some voivodeships of Poland. Obtained results were encouraging.

The paper is organized as follows. Next section includes formal definition of TSPwP. Section 3 describes in details the GA. Experimental results are presented in Section 4. The paper ends with some remarks about future work.

2. Problem definition

A network of cities in our model 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 $\{i, j\} \in E$ is an element of the set *E* and means that there is a possibility to travel from the city *i* to the city *j* (and vice versa). The weight a_{ij} for an undirected edge $\{i, j\}$ denotes a distance between cities i and j. Additionally, with each vertex a non-negative number meaning a profit is associated. Let $P = \{p_1, p_2, ..., p_n\}$ 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.

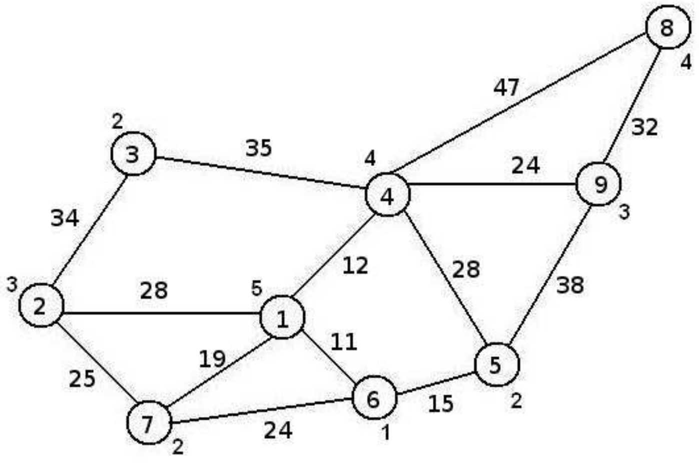

Fig. 1. A graph representation of a network of cities

A graph representation of an exemplary network of cities is shown in Fig. 1. It is a simple example of the network which includes nine cities. The a_{ij} values are marked on the edges and the p_i values are: {5, 3, 2, 4, 2, 1, 2, 4, 3} (marked beside vertices). One can see that the highest profit equals to 5 can be gained during visiting the depot.

The TSPwP 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}.

Assuming $c_{max} = 100$, for the graph presented in Fig. 1, one possible solution could be: 1 - 4 - 9 - 5 - 6 - 1. In this case the tour length equals to 100 and the collected profit equals to 15.

3. GA for discovering routes

GAs are adaptive heuristic search algorithms based on the evolutionary ideas of natural selection and heredity. First introduced by John Holland in the 60s, GAs has been widely studied, experimented and successfully applied in many fields [6,11].

In a typical GA, a population of candidate solutions (called chromosomes) is evolved during artificial evolution. Traditionally, solutions are represented as binary strings, but other encodings are also possible. The GA starts from a population of randomly generated individuals. In each generation, the fitness of every individual is evaluated. Based on the fitness, individuals are stochastically selected from the current population to the next one. This step is called a selection. Individuals in the new population undergo genetic operators: crossover and mutation. The new population is then used in the next iteration of the algorithm. The algorithm usually terminates when a maximum number of generations *ng* has been reached [12].

The pseudocode of the GA described in this paper is presented below.

```
  Genetic algorithm
  Begin
    generate an 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 equals to t\_size;
        divide population into disjont pairs;
        cross each pair if possible;
        mutate each individual if possible;
      End;
    choose the best individual from the final population as the result;
  End;
```

3.1 Genetic reprezentation and initial popualtion generating

The first step in the GA is encoding a solution into a chromosome. We use the path representation which is the most natural for this problem [11]. In this approach, a tour is encoded as a sequence of vertices. For example, the tour 1 - 4 - 5 - 6 - 1 is represented by the sequence 1 - 4 - 5 - 6 - 1, as was described in the Section 2.

The GA starts with a population of *P* solutions of TSPwP. The initial population is generated in a special way. Starting at the depot, with equal probability we choose a city to which we can travel from the depot. 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. One can see 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 GA quickly breaks these symmetries.

Let us construct an individual for the problem presented in Fig. 1 with the assumption that $c_{max} = 150$. We start at the node 1 and have to choose one node from the set $\{2, 4, 6, 7\}$. Let us assume that node 2 was selected. Since the distance between 1 and 2 equals to 28, the current tour length equals to 28 (and is not greater then $c_{max}/2$). The partial tour is 1 - 2. Starting at the node 2, we can select the node 3 or the node 7, with equal probability (we exclude node 1). Let us assume that the node 3 was selected. The current tour is now 1 - 2 - 3 with the length equal to 62. Starting from the node 3 we can only select the node 4 but this situation will cause crossing the treshold value $c_{max}/2$. We must reject the node 4 and return to the depot the same way. Our complete tour is 1 - 2 - 3 - 2 - 1 and has the length equal to 124.

3.2 Fitness computing

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 realized only during first visiting of a given vertex. For example, the fitness of the individual represented by the chromosome: 1 - 2 - 3 - 2 - 1 equals to 10.

3.3 Selection

Once we have the fitness function computed, the GA 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 (randomizing with returns). This step is repeated P times. The parameter t_{size} should be carefully set because the higher t_{size}, the faster convergence of the GA.

3.4 Crossover

We present a new heuristic crossover operator adjusted to our problem. In the first step, 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 can not be done and parents remain unchanged. Crossover is implemented in the following way. First we randomly choose one common gene from the set of common genes in both parents (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 preserve the constraint c_{max}, they replace in the new population their parents. If one offspring individual does not preserve the constraint c_{max}, its position in the new population is occupied by better (more fitter) parent. If both children do not preserve the constraint c_{max}, they are replaced by their parents in the new population. The example of the crossover is presented in Fig. 2. This example concerns Fig. 1 with the assumption that $c_{max} = 200$.

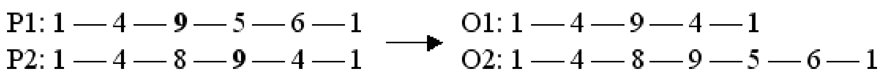
P1: 1 — 4 — 9 — 5 — 6 — 1
P2: 1 — 4 — 8 — 9 — 4 — 1
O1: 1 — 4 — 9 — 4 — 1
O2: 1 — 4 — 8 — 9 — 5 — 6 — 1

Fig. 2. The example of the crossover operator

The length of the tours represented by offsppring are equal to 72 and 155, respectively. Since both offspring individuals preserve the constraint c_{max}, they replace in the new population their parents.

3.5 Mutation

The last genetic operator is a mutation. Each individual in the current population undergo mutation. It is performed in the following way. First we randomly select a position in a chromosome where a mutation will be performed. Then we try to insert a city (from the set of possible cities) at this position. If inserting a city do not violate the constraint c_{max}, we keep this new city in a tour otherwise it is rejected.

For example, let us look at the individual O2 in Fig. 3. Let us assume that this individual is to be muated and randomly selected position in the chromosome is marked with an arrow. The only city we can insert between the cities 6 and 1 is the city 7. Inserting this city will result in the tour length equal to 198. Since $c_{max} = 200$, we keep the city 7 on its position. The new mutated individual O2' replaces O2 in the population.

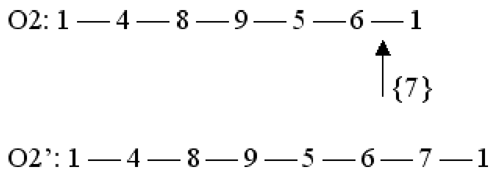
O2: 1 — 4 — 8 — 9 — 5 — 6 — 1
{7}
O2': 1 — 4 — 8 — 9 — 5 — 6 — 7 — 1

Fig. 3. The example of the mutation operator

4. Experimental results

We conducted experiments on networks of cities in the following voivodeships of Poland: podlaskie, mazowieckie and warminsko-mazurskie. From each voivodeship, twenty cities were selected. A network was created from a real map, by including to a graph main segments of roads. Profits associated with a given city were determined according to a number of inhabitants in a given city. The more inhabitants, the higher profit associated with a given city. These rules are presented in a Tab. 1.

Table 1. Profits associated with a given city

number of inhabitants	profit
<= 10000	1
(10000, 20000>	2
(20000, 30000>	3
(30000, 40000>	4
> 40000	5

The data for each voivodeship (profits and distances between cities) are presented in Appendix.

The parameters for the GA were determined through series of experiments described in [8]. Based on results of those experiments we set $P = 200$, $t_{size} = 3$ and $ng = 100$. Tests were performed for three c_{max} values: 300, 500 and 700. Results of experiments are presented in later subsections.

4.1 Voivodeship podlaskie

The best results from ten GA runs are presented in Tab. 2 and 3.

Table 2. The best results for voivodeship podlaskie

	$c_{max} = 300$	$c_{max} = 500$	$c_{max} = 700$
profit	25	37	46
distance	298	498	697

Table 3. Chromosomes of the best individuals for voivodeship podlaskie

	chromosome
$c_{max} = 300$	[1,9,6,19,8,2,14,13,17,15,1]
$c_{max} = 500$	[1,7,5,17,15,13,14,2,8,19,9,6,4,12,1]
$c_{max} = 700$	[1,7,18,11,5,17,15,13,14,2,19,8,19,6,9,6,4,3,4,12,20,1]

4.2 Voivodeship mazowieckie

The best results from ten GA runs are presented in Tab. 4 and 5.

Table 4. The best results for voivodeship mazowieckie

	$c_{max} = 300$	$c_{max} = 500$	$c_{max} = 700$
profit	23	41	50
distance	291	494	700

4.3 Voivodeship warminsko-mazurskie

The best results from ten GA runs are presented in Tab. 6 and 7.

Table 5. Chromosomes of the best individuals for voivodeship mazowieckie

	chromosome
$c_{max} = 300$	[1,10,5,12,6,11,15,1]
$c_{max} = 500$	[1,11,6,12,5,2,3,4,9,14,13,16,1]
$c_{max} = 700$	[1,13,14,9,8,7,8,4,3,2,5,6,12,6,11,15,16,1,10,1]

Table 6. The best results for voivodeship warminsko-mazurskie

	$c_{max} = 300$	$c_{max} = 500$	$c_{max} = 700$
profit	48	57	59
distance	295	500	700

Let us look closely at the results obtained for voivodeship warminsko-mazurskie. Fig. 4 presents the best of ten GA runs for each c_{max}. On each plot we can see the profit of the best individual in a given generation. For all plots presented in this figure, the GA quickly (before 15th generation) finds the optimal (or suboptimal) solutions. Further generations do not bring any improvement.

For $c_{max} = 500$, the profit of the best individual equals to 57. The chromosome of this individual consists of 25 cities and has the tour length equal to 500. However, for $c_{max} = 700$, the profit of the best individual is only 2 greater than for $c_{max} = 500$ and the chromosome length equals to 44 genes (the tour length equals to 700). This problem is explained below.

For voivodeship warminsko-mazurskie, the maximal profit which can be gained equals to 59 for a tour length equal to 435. A solution with the best possible profit is found quickly - in 11th generation. However, the length of the solution (with the maximal profit) from the final generation equals to 700 (Tab. 6). The main reason is the mutation operator. In spite of the fact that a current solution can not be improved (because it is the global maximum), the mutation causes inserting to the current tour cities which increase the total tour length but can not increase the collected profit. This situation is clear when looking at the chromosome of the best individual for $c_{max} = 700$ from Tab. 7. One can see that some citities e.g. 17, 14, 19 repeat in the

Table 7. Chromosomes of the best individuals for voivodeship warminsko-mazurskie

	chromosome
$c_{max} = 300$	[1,5,17,19,14,9,7,10,6,20,6,3,12,8,16,11,4,5,1]
$c_{max} = 500$	[1,16,11,4,5,17,14,9,13,2,13,9,7,10,8,12,3,15,18,20,6,10,7,10,1]
$c_{max} = 700$	[1,5,17,19,17,19,14,17,19,17,14,17,19,17,19,14,19,17,14,17,14,19, 2,13,9,7,10,6,20,18,15,3,12,8,16,11,4,11,16,11,16,11,16,1]

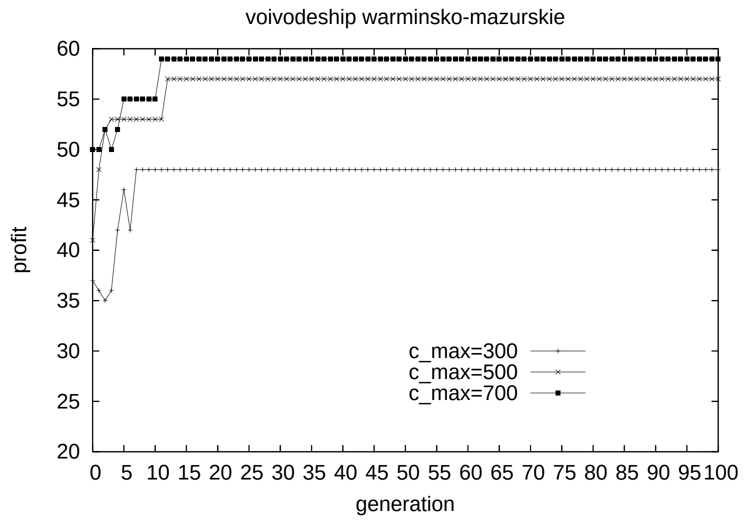

Fig. 4. The GA run for voivodeship warminsko-mazurskie

chromosome but they do not influence fitness value and can be removed from the chromosome. This way of action of the mutation needs to be improved. The new cities should be inserted to the tour only if they cause an improvement of the fitness function, otherwise the tour length is unnecessary incremented.

To verify this hypothesis, we implemented improved version of mutation operator. In this case, a new city is inserted to the tour only if it is not present in a tour yet. Otherwise, mutation is not performed. The new mutation was tested on the same data as the ordinary mutation. Differences were most evidently observed for $c_{max} = 700$ and are presented in Tab. 8.

Table 8. Results of the improved mutation, $c_{max} = 700$

voivodeship	chromosome	profit	distance
podlaskie	[1,7,18,5,11,16,17,15,13,14,2,8,19,6,4,3,10,4,12,20,1]	46	677
mazowieckie	[1,11,6,10,5,2,3,4,7,8,9,14,13,16,18,19,17,15,1]	53	691
warminsko-mazurskie	[1,5,4,11,16,8,12,3,15,18,20,6,10,7,9,13,2,19,14,17,5,1]	59	435

One can see that results obtained with the improved mutation are better in the context of collected profit (mazowieckie) and also in the context of total distance (podlaskie, mazowieckie, warminsko-mazurskie). The problem of repeated cities was majorly eliminated. Of course, repeated cities may again appear in a solution due to crossover operator.

5. Conclusions

In this paper we presented a version of TSPwP. Additional assumptions to this problem were proposed in the paper which make the problem more real-life. The aim of the work was designing a GA to deal with this problem. The GA proposed in the paper was tested on some voivodeships in Poland, obatining satisfactory results. However, these results should be compared with results obtained by the other heuristic algorithms.

Another issue which must be carefully studied is the muation operator. Results of experiments have shown that this operator has significant role in the quality of obtained solutions. Two versions of muation inserting a city to a tour were implemented and tested. However, another kind of mutation can also be considered, for example inserting a fragment of tour or exchanging cities in a tour.

The results presented in this paper are results of preliminary experiments. The future work will be focused on testing the improved version of the GA on bigger and more dense networks, for example for cities from the whole Poland.

6. Appendix

The data for voivodeship podlaskie are presented below.

Voivodeship podlaskie
1 (Bialystok 5) **2** 80 **4** 90 **5** 49 **7** 39 **9** 42 **12** 40 **14** 68 **15** 28 **20** 47
2 (Lomza 5) **1** 80 **8** 32 **14** 24 **19** 25
3 (Suwalki 5) **4** 34 **10** 31
4 (Augustow 4) **1** 90 **3** 34 **6** 42 **10** 43 **12** 76
5 (Bielsk Podlaski 3) **1** 49 **7** 28 **11** 30 **17** 25 **18** 25
6 (Grajewo 3) **4** 42 **9** 37 **19** 38
7 (Hajnowka 3) **1** 44 **5** 28 **18** 27
8 (Kolno 2) **2** 32 **19** 17
9 (Monki 2) **1** 42 **6** 37 **19** 54
10 (Sejny 1) **3** 31 **4** 43
11 (Siemiatycze 2) **5** 30 **16** 38 **18** 38
12 (Sokolka 2) **1** 40 **4** 76 **20** 26
13 (Wysokie Mazowieckie 1) **14** 21 **15** 31 **17** 27

14 (Zambrow 3) **1** 68 **2** 24 **13** 21 **16** 43
15 (Lapy 2) **1** 28 **13** 31 **17** 32
16 (Ciechanowiec 1) **11** 38 **14** 43 **17** 25
17 (Bransk 1) **5** 25 **13** 27 **15** 32 **16** 25
18 (Kleszczele 1) **5** 24 **7** 27 **11** 38
19 (Stawiski 1) **2** 25 **6** 38 **8** 17 **9** 54
20 (Krynki 1) **1** 47 **12** 26

Each city has an unique number. The capital of the voivodeship has the number 1 (this rule is also applied to other voivodeships). Profits associated with cities are given in the brackets. In each line the distances between a given city and other cities are presented (in pairs: a number of a city and a distance). For example let us look at the first line. One can see that the distance between Bialystok and Lomza (number 2) equals to 80, the distance between Bialystok and Augustow (number 4) equals to 90 etc.

The data for voivodeship mazowieckie are presented below.

Voivodeship mazowieckie
1 (Warszawa 5) **3** 81 **7** 58 **10** 37 **11** 54 **13** 40 **16** 34 **15** 46
2 (Ciechanow 5) **12** 82 **5** 36 **3** 37
3 (Makow Mazowiecki 2) **2** 37 **1** 81 **4** 56
4 (Ostrow Mazowiecka 3) **3** 56 **7** 42 **8** 37 **9** 55
5 (Plonsk 3) **12** 49 **2** 36 **6** 31 **10** 33
6 (Wyszogrod 1) **12** 38 **11** 24 **5** 31 **10** 37
7 (Wyszkow 3) **4** 42 **8** 18 **1** 58
8 (Lochow 1) **9** 46 **13** 43 **7** 18 **4** 37
9 (Sokolow Podlaski 2) **4** 55 **8** 46 **14** 30
10 (Nowy Dwor Mazowiecki 3) **5** 33 **6** 37 **1** 37
11 (Sochaczew 4) **6** 24 **1** 54 **15** 64
12 (Plock 5) **2** 82 **6** 38 **5** 49
13 (Minsk Mazowiecki 4) **1** 40 **8** 43 **14** 50 **16** 41
14 (Siedlce 5) **13** 40 **9** 30
15 (Grojec 2) **11** 64 **1** 46 **16** 29 **17** 27
16 (Gora Kalwaria 2) **1** 34 **15** 29 **13** 41 **18** 52
17 (Bialobrzegi 1) **15** 27 **18** 50 **19** 35
18 (Kozienice 2) **16** 52 **17** 50 **19** 37 **20** 28
19 (Radom 5) **17** 35 **18** 37 **20** 33

20 (Zwolen 1) **19** 33 **18** 28

The data for voivodeship warminsko-mazurskie are presented below.

Voivodeship warminsko-mazurskie
1 (Olsztyn 5) **5** 20 **14** 40 **10** 35 **16** 10
2 (Elblag 5) **13** 40 **19** 50
3 (Elk 5) **6** 20 **15** 10 **12** 20
4 (Ilawa 4) **5** 30 **11** 35
5 (Ostroda 4) **4** 30 **19** 15 **17** 10 **1** 20
6 (Gizycko 3) **10** 10 **20** 5 **3** 20
7 (Ketrzyn 3) **9** 20 **10** 5
8 (Szczytno 3) **16** 15 **10** 25 **12** 30
9 (Bartoszyce 3) **13** 50 **7** 20 **14** 10
10 (Mragowo 3) **7** 5 **6** 10 **12** 15 **8** 25 **1** 35
11 (Dzialdowo 3) **4** 35 **16** 20
12 (Pisz 2) **10** 15 **3** 20 **8** 30
13 (Braniewo 2) **2** 40 **9** 50
14 (Lidzbark Warminski 2) **9** 10 **1** 40 **17** 10 **19** 10
15 (Olecko 2) **18** 5 **3** 10
16 (Nidzica 2) **1** 10 **8** 15 **11** 20
17 (Morag 2) **19** 10 **14** 10 **5** 10
18 (Goldap 2) **20** 20 **15** 5
19 (Paslek 2) **2** 50 **14** 10 **17** 10 **5** 15
20 (Wegorzewo 2) **6** 5 **18** 20

References

[1] Fatih Tasgetiren, M.: A Genetic Algorithm with an Adaptive Penalty Function for the Orienteering Problem. Journal of Economic and Social Research, vol. 4 (2), pp. 1-26, 2002.

[2] Feillet, D., Dejax, P., Gendreau, M.: Traveling Salesman Problems with Profits, Transportation Science, vol. 39 (2), pp. 188-205, 2005.

[3] Fischetti, M., Salazar González, J. J., Toth, P.: Solving the orienteering problem through branch-and-cut. INFORMS Journal on Computing, vol. 10 (2), pp. 133-148, 1998.

[4] 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.

[5] 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.

[6] Goldberg, D. E.: Genetic algorithms and their applications. WNT, Warsaw, 1995.

[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] Koszelew, J., Piwonska, A.: Tunning Parameters of Evolutionary Algorithm in Travelling Salesman Problem with Profits and Returns. Archives of Transport System Telematics, to appear (2010).

[9] Lawler, E. L., Lenstra, J. K., Rinnooy Kan, A. H. G., Shmoys, D. B.: The Traveling Salesman Problem: A Guided Tour of Combinatorial Optimization. John Wiley & Sons, 1985.

[10] 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.

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

[12] Mitchell, M.: An Introduction to Genetic Algorithms. MIT Press, 1998.

[13] Ramesh, R., Brown, K. M.: An efficient four-phase heuristic for the generalized orienteering problem. Computers & Operations Research, vol. 18 (2), Elsevier, pp. 151-165, 1991.

[14] 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.

[15] Sevkli, Z., Sevilgen, F. E.: Variable Neighborhood Search for the Orienteering Problem. LNCS, vol. 4263, Springer, pp. 134-143, 2006.

[16] 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.

[17] 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.

ALGORYTM GENETYCZNY ODNAJDUJE TRASY W PROBLEMIE KOMIWOJAŻERA Z ZYSKAMI

Streszczenie Problem komiwojażera z zyskami (ang. TSP with profits) jest pewną wersją klasycznego problemu komiwojażera, w której nie jest konieczne odwiedzenie wszystkich wierzchołków grafu. Zamiast tego, z każdym wierzchołkiem związana jest pewna liczba oznaczająca zysk. Problem polega na znalezieniu cyklu w grafie, który maksymalizuje zysk, ale którego koszt nie przekracza zadanego ograniczenia. Problem ten jest problemem NP-trudnym. Do tak postawionego problemu, w pracy zaproponowano dodatkowe założenia. Przyjęto mianowicie, że graf nie musi być pełny. Ponadto dopuszczona jest możliwość powrotów, czyli ponownego odwiedzenia danego wierzchołka, przy założeniu jednak, iż zysk realizowany jest tylko podczas pierwszego odwiedzenia. Przy tych dodatkowych założeniach problem jest bardziej realny i może mieć konkretne zastosowania w logistyce i spedycji. Do rozwiązania problemu zaproponowano algorytm genetyczny, uzyskując bardzo dobre wyniki.

Słowa kluczowe: problem komiwojażera z zyskami, algorytm genetyczny

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