DIFFERENT APPROACHES TO INFEASIBLE SOLUTIONS IN EVOLUTIONARY ALGORITHMS FOR THE ORIENTEERING PROBLEM

Krzysztof Ostrowski

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

Abstract: The Orienteering Problem (OP) is a combinatorial optimization problem defined on weighted graphs. The purpose of the OP is to find a path of limited length which maximizes total profit (collected in vertices). This paper presents comparison of different approaches to infeasible solutions (too long paths) in evolutionary algorithms solving the OP. A group of evolutionary algorithms (varying in crossover and selection operators) was tested in different configurations: with and without infeasible solutions in populations. Parameters for all algorithm configurations were obtained from automatic tuning procedure (ParamILS). Results show that presence of too long paths in a population can improve quality of resulting solutions. The presented metaheuristic generated optimal or close to optimal solutions for the tested benchmark networks.

Keywords: infeasible solutions, evolutionary algorithms, Orienteering Problem

1. Introduction

The Orienteering Problem (OP) is a combinatorial optimization problem defined on graphs. Its practical applications include tourist trip planning, transport logistics, DNA sequencing problem and others [1] [2]. The OP is an NP-hard problem [3] and computing exact solutions for larger graphs can be very time-consuming. For this reason most of proposed approaches to solve the OP were heuristic. Evolutionary algorithms (EAs) are among most popular metaheuristics for solving optimization problems. Performance of EAs is determined by various factors like choice of recombination operators and parameters values. For optimization problems with constraints another important aspect is a way of dealing with infeasible solutions in a population. The paper presents comparison of two groups of EAs solving the OP: one without infeasible solutions (presented in [4]) and another with infeasible solutions (adaptive

Advances in Computer Science Research, vol. 14, pp. 143-161, 2018. DOI: 10.24427/acsr-2018-vol14-0009

penalty). The results show superiority of EAs with infeasible solutions in populations. The article is organized as follows. Section 2 presents mathematical definition of the OP. Section 3 presents a review of the literature on the OP. In Section 4 the proposed EA is described. Experimental results are included in section 5 and conclusions are drawn in Section 6.

2. Mathematical definition of the Orienteering Problem

The OP is defined on a weighted graph $G = (V, E)$. Each edge has associated a nonnegative cost and each vertex has a nonnegative profit. The purpose of the OP is to find a path (or cycle) between a given pair of vertices (s - start vertex, e - end vertex) which maximizes total collected profit (sum of profits of visited vertices) and its total cost (sum of costs of visited edges) is limited by a given constraint (C_{max}). Each vertex can be included in the path at most once (except the situation when $s = e$). Let w_{ij} be a cost of edge (i, j) and p_i be a profit of vertex i. Let x_{ij} be a binary variable equal to 1 if a solution contains edge (i, j) and 0 otherwise. Let r_i be a position of vertex i in a solution - it is defined only for vertices included in a path. The OP can be formulated mathematically:

$$max \sum_{i \in V} \sum_{j \in V} (p_i \cdot x_{ij}) \tag{1}$$

$$\sum_{i \in V} \sum_{j \in V} w_{ij} \cdot x_{ij} \leq C_{max} \tag{2}$$

$$\sum_{j \in V} x_{sj} = \sum_{i \in V} x_{ie} = 1 \tag{3}$$

$$\bigvee_{k \in V \setminus \{s,e\}} (\sum_{i \in V} x_{ik} = \sum_{j \in V} x_{kj} \leq 1) \tag{4}$$

$$r_s = 1 \tag{5}$$

$$\bigvee_{i \in V,\ j \in V \setminus \{s\},} (x_{ij} = 1 \Rightarrow r_j = r_i + 1) \tag{6}$$

Objective function maximization is described by formula 1. Constraint 2 relates to maximal path cost, which cannot exceed C_{max}. Constraint 3 indicates that the path starts in vertex s and ends in vertex e while formula 4 guarantees that all other vertices are included at most once in the path. Formulas 5-6 assure that the solution is a continuous path without sub-cycles.

3. Literature review of the Orienteering Problem

The OP was introduced in 1984 [5] and since then various approaches have been proposed to solve the problem. Branch-and-cut and branch-and-bound methods are among few exact algorithms applied for the OP [6] [7]. These methods usually needed long time to solve larger OP instances. Therefore most researchers concentrated on heuristic methods solving the OP.

Tsiligirides [5] proposed first method for the OP (S-algorithm), which based on the Monte Carlo method and heuristic procedure. Golden et al. [3] proposed another approach (use of greedy route construction and a centre-of-gravity heuristic). Chao et al. [8] introduced a two-step iterative heuristic, which obtained results of highest quality at that time.

Tasgetiren [9] was the first to present a genetic algorithm for the OP. The genetic operators used were: tournament selection, injection crossover and mutation with local search methods (add, omit, replace and swap operators). A tabu search heuristic for the OP was presented by Gendreau et al. [10]. High-quality solutions were obtained by the algorithm on randomly generated test instances (up to 300 nodes).

Vansteenwegen et al. [11] applied the guided local search method (GLS) for the Team Orienteering Problem (TOP). This method uses local search heuristics (like insert, replace, 2-opt) and reduces the likelihood of becoming trapped in a local optimum (thanks to disturb operator). The GLS meta-heuristic method yields satisfactory results for small-sized networks and is used in the Mobile Tourist Guide [1]. Heuristics for the TOP generate m disjoint routes (except start/end vertices) and total collected profit is maximized. For $m = 1$ the GLS solves classic Orienteering Problem.

In 2009 Schilde et al. [12] published two metaheuristics: Variable Neighbourhood Search (VNS) and Ant Colony Optimization (ACO). For standard benchmark instances both algorithms achieved better average results than Chao's method [8] and GLS [11].

Souffriau et al. [13] proposed Greedy Randomized Adaptive Search Procedure (GRASP) for the TOP. Later the method was adapted to the classic OP by Campos et al. [14]. The authors also added a path relinking (PR) method to GRASP. Construction of initial solutions is partially greedy and partially random: ratio between greediness and randomness is used when inserting new vertices (four different insertion methods are used). Afterwards, local search procedures (exchange and insert) are applied in order to reduce path length and increase its profit. GRASPwPR metaheuristic has an additional step (path relinking), which is performed for each

pair of solutions P_1 and P_2 generated by GRASP: the P_1 path is gradually transformed into the P_2 by a sequence of vertex insertions and deletions. The best intermediate paths are then improved by local search operators. GRASPwPR was tested on many benchmark instances and generated very high-quality solutions (one of the best among metaheuristics).

The author's research is focused on metaheuristic solutions for various problems from the OP family (mainly evolutionary algorithms and their composition with local search methods). Apart from the classic OP [15] [4] [16] [17] the author also proposed metaheuristics for the Time-Dependent Orienteering Problem (costs of edges vary with time) [18] [19] [20] focusing also on practical aspects of the problem (trip planning in public transport networks). Obtained results showed advantage of evolutionary algorithms over other known meta-heuristics (like GLS, GRASP, GRASPwPR), especially for larger test instances.

4. Description of the proposed evolutionary algorithm

The author proposed evolutionary algorithm (EA) with embedded local search operators to solve the Orienteering Problem. Path representation is used in the EA: successive genes in a chromosome are equivalent to successive vertices included in a solution path. At first, an initial population of P_{size} random routes is created. Afterwards, evolutionary phase takes place - operators of selection, crossover, mutation and disturb are applied repeatedly. Mutation, crossover and disturb operators can be either random or heuristic (local search) and frequency of using random and heuristic operators is determined by algorithm parameters (heuristic coefficients). Evolutionary phase terminates after a fixed number N_g of generations, or earlier if there have been no improvements in the last C_g generations. Finally the population undergo local improvement procedure. The best feasible path (with highest total profit) obtained during algorithm run is EA final result. Three different crossover operators and three different selection methods were tested. The algorithm is derived from [4]. However, in the previous publication only feasible solutions were present in populations while in this paper a comparison between different ways of dealing with solutions infeasibility is made. A short description of the algorithm is given below (more details about the operators are presented in [4]).

4.1 Selection

Three different selection procedures were tested:
1) Unbiased tournament parent selection [21]

2) Fitness proportionate parent selection with stochastic universal sampling [22]
3) Deterministic crowding survivor selection [23]
First two methods (parent selections) are executed at the beginning of each EA iteration and they create intermediate population, which undergo crossover and mutation. The last method (deterministic crowding) works differently: it is executed after crossover procedure and consists of competitions of child-parent pairs for a place in the next generation.

4.2 Crossover

Initially, pairs of individuals (parents) are randomly chosen from the population (number of pairs depends on crossover probability). Afterwards, each pair undergo crossover procedure. Crossover methods tested in the evolutionary algorithms are: 2-point crossover [4], injection crossover [9], and path relinking crossover [14]. Each crossover variant has two versions: heuristic and random. Probability of using heuristic version is determined by an algorithm parameter (crossover heuristic coefficient).

4.3 Mutation

First, a group of individuals is randomly chosen from a population (size of the group depends on mutation probability). Afterwards, each of them undergoes mutation, which consists of 2-opt procedure (reversing one path fragment to shorten the path) as well as one vertex insertion or deletion. Afterwards, a small group of individuals undergo disturb procedure. This is another type of mutation which removes some path fragment. This procedure helps to escape from local optima but it is destructive and should be executed rarely. Insert/delete/disturb procedures have two types: heuristic (local search) and random. Frequency of these two types of mutation is determined by algorithm parameters (heuristic coefficients) in an analogical way to crossover procedure. For more details regarding mutation see [4].

4.4 Approaches to infeasibility

In this paper two different approaches were tested.

No infeasible solutions allowed: This approach was described in [4]. For feasible solutions the fitness function was equal to path profit but for infeasible solutions the fitness was 0. Genetic operators weren't allowed to create too long paths.

Adaptive penalty of infeasible solutions: Too long paths are allowed in populations (they can be created by genetic operators) but they are penalized by fitness function. Let S be a path with total profit p_S and total cost c_S. Fitness function of a feasible path (not exceeding C_{max}) is equal to its profit (as in previous subsection). If path S is too long its fitness is expressed by the following formula:

$$fitness(S) = p_S \cdot (\frac{C_{max}}{c_S})^k \tag{7}$$

Fitness function decreases as solution is farther from feasibility boarder C_{max}. Parameter k represents penalty severity. For $k = 1$ fitness function can be interpreted as expected profit of a path after shortening it to C_{max} limit. Larger k values mean stronger penalties. Fitness function is adaptive and depends on number of infeasible solutions in a population. At the beginning $k = 1$ and every 10 generations number of infeasible solutions is checked. If more than α percent of individuals are infeasible then k is increased by 0.1. It enables to control number of infeasible solutions in a population.

4.5 Algorithm parameters

Algorithm parameters are described in table 1 and table 2. Probability parameters p describe percentage of population which is chosen for mutation/crossover/disturb procedures in each generation. Heuristic coefficient parameters z determine the probability of using heuristic operator version (1-z is the probability of using random version). Values of population parameters from table 2 were set by the author. Population size is a compromise between exploration ability and computation time. Parameters associated with generations number were determined during earlier experiments and should not stop EA prematurely. The remaining parameters values (table 1) were determined by automatic tuning procedure - Parameters Iterated Local Search (ParamsILS) [24]. This meta-algorithm searches multi-dimensional parameter space using local search procedures. Vectors of parameter values (meta-solutions) processed by ParamsILS are evaluated by averaging results from multiple runs of the tuned EA. More details about the tuning procedure can be found in [4].

5. Experimental results

All experiments (calibration and testing) were carried out on a computer with Intel i7 3.6 GHz processor and 4GB of RAM. Programs were implemented in C++ and executed on Linux operating system.

Table 1. EA parameters tuned automatically

parameter	**description**
p_k	crossover probability
p_m	mutation probability
p_z	disturb probability
z_k	crossover heuristic coefficient
z_m	mutation heuristic coefficient
z_z	disturb heuristic coefficient
α	max. percentage of infeasible solutions in a population

Table 2. EA parameters set by the author

parameter	**description**	**value**
P_{size}	population size	100
Ng	maximum number of generations	5000
Cg	maximum number of generations without improvement	500

5.1 Problem instances

In table 3 there are all OP instances used in this experiment. Class I tests come from TSPLIB library (distance matrices in XML) and were adapted to the OP by Fischetti et al. [6]. Profits of vertices were generated according to the formula:

$$p_i = 1 + (7141 \cdot i + 73)(mod\, 100) \tag{8}$$

where p_i is profit of vertex i. C_{max} values for class I instances were set as 50 percent of shortest hamiltonian cycles. Distances between vertices were truncated to integer numbers.

Class II tests were Vehicle Routing Problem (VRP) instances adapted to OP by Fischetti et al [6]. In these instances customer demands from VRP were interpreted as vertices profits in the OP. C_{max} values were set as 25, 50 and 75 percent of shortest hamiltonian cycles. Distances between vertices were rounded to nearest integers.

Class III tests were created by the author and base on a network of 908 cities in Poland [25]. Profits are equal to numbers of inhabitants (expressed in thousands) and graph weights were calculated as big circle distances between cities (in km). For all instances of all classes paths start and end in vertex 1.

Table 3. All problem instances with C_{max} values

Class	Instance name	C_{max}	Class	Instance name	C_{max}
I	kroA100	10641	II	eil101A	158
	kroB100	11071		cmt121A	137
	kroC100	10375		cmt151A	175
	kroD100	10647		cmt200A	191
	kroE100	11034		gil262A	595
	rd100	3955		eil101B	315
	eil101	315		cmt121B	273
	lin105	7190		cmt151B	350
	pr107	22152		cmt200B	382
	gr120	3471		gil262B	1189
	pr124	29515		eil101C	472
	bier127	59141		cmt121C	409
	pr136	48386		cmt151C	525
	gr137	34927		cmt200C	573
	pr144	29269		gil262C	1784
	kroA150	13262	III	pl500	500
	kroB150	13065		pl1000	1000
	pr152	36841		pl1500	1500
	u159	21040		pl2000	2000
	rat195	1162		pl2500	2500
	d198	7890		pl3000	3000
	kroA200	14684			
	kroB200	14719			
	ts225	63322			
	pr226	40185			
	gil262	1189			
	pr264	24568			
	pr299	24096			
	lin318	21015			
	rd400	7641			

5.2 Tuning results

Calibration process was carried out on three problem instances: pr299 and rd400 (from class I) and gil262 (from class II). These instances are among largest with known optimal solutions. To assess quality of the algorithm with given parameter values it was run 10 times for each calibration network and its results were averaged. Gaps are expressed in percent and were obtained according to formula $100 \cdot (1 - \frac{P_{alg}}{P_{opt}})$ where P_{alg} is average route profit obtained by EA while P_{opt} is profit of an optimal route. The experiment was conducted for 18 different algorithm versions (all combinations of 3 selection procedures, 3 crossovers and 2 approaches to infeasible solutions).

The calibration results are presented in table 4. They were divided into two groups (varying in a way of dealing with infeasible solutions) and results of the best configurations of each group are in bold. It can be seen that for all 9 combinations (varying in crossover and selection methods) EAs with infeasible solutions obtain better average results than those without infeasible solutions (the difference is about 0.4 percent). The difference between EAs with and without infeasible solutions is biggest for rd400 network (0.6 percent on average), which suggests that searching both sides or feasibility boarder is more important for instances with larger solution space.

In almost all cases the best results were obtained by a combination of 2-point crossover and deterministic crowding selection. Solutions of the highest quality (only 0.13 percent of average gap to optimal solutions) were produced by EA version with adaptive penalty - they are 0.5 percent better than those obtained by the best EA configuration without infeasible solutions.

Calibration results show the importance of using local search operators in EAs (heuristic coefficients between 0.6 and 1 in most cases). One can see that disturb procedures (which can be destructive when overused) are executed rarely compared to mutation and crossover operators. It can also be seen that penalty parameter ($\alpha \geq 70$) is not severe in most cases - a lot of infeasible solutions are allowed in populations.

Table 4. Tuning results for different EA configurations with best found sets of parameter values and average gaps to optimal results for calibration networks. Crossover types: 2P - two point crossover, INJ - injection crossover, PR - path relinking crossover. Selection types: TUR - unbiased tournament selection, SUS - fitness proportionate selection with stochastic universal sampling, CRO - deterministic crowding. The result of the best EA configuration from each group in bold.

Infeasible solutions	Crossover	Selection	Calibrated parameters			rd400	pr299	gil262C	All 3 networks
			p_k p_m p_z	z_k z_m z_z	α	gap (%)	gap (%)	gap (%)	avg. gap (%)
No infeasible solutions	2-P	TUR	0.6 1.0 0.70	0.4 0.6 0.4	-	3.47	3.32	1.14	2.64
		SUS	0.6 1.0 0.10	0.8 0.8 0.6	-	2.37	1.42	0.81	1.53
		CRO	1.0 1.0 0.10	0.6 0.8 1.0	-	0.63	0.91	0.38	**0.64**
	INJ	TUR	0.8 1.0 0.50	0.6 0.6 0.6	-	4.84	2.40	1.53	2.92
		SUS	0.6 1.0 0.00	0.8 0.8 -	-	3.20	2.37	0.74	2.10
		CRO	1.0 1.0 0.00	0.4 0.6 -	-	5.96	2.34	2.54	3.61
	PR	TUR	0.6 1.0 0.70	1.0 0.6 0.6	-	3.86	1.90	0.63	2.13
		SUS	0.4 1.0 0.10	1.0 0.8 0.6	-	2.09	2.63	0.41	1.71
		CRO	0.8 1.0 0.03	0.4 0.8 0.2	-	2.36	0.93	0.45	1.25
Adaptive penalty	2-P	TUR	0.8 1.0 1.0	0.8 0.6 0.2	50	3.98	2.20	1.12	2.43
		SUS	0.4 1.0 0.03	1.0 0.8 1.0	90	1.45	1.15	0.46	1.02
		CRO	1.0 1.0 0.01	1.0 0.8 0.8	90	0.18	0.14	0.07	**0.13**
	INJ	TUR	0.2 1.0 0.3	0.4 0.6 1.0	90	3.49	2.11	1.34	2.32
		SUS	0.2 1.0 0.03	1.0 0.8 0.6	50	2.04	2.45	0.54	1.68
		CRO	1.0 1.0 0.00	0.0 0.6 -	30	5.01	2.56	2.35	3.31
	PR	TUR	0.8 0.8 0.5	0.6 0.6 1.0	70	3.49	1.03	0.77	1.76
		SUS	0.6 1.0 0.3	1.0 1.0 0.2	90	2.06	0.99	0.35	1.13
		CRO	0.6 1.0 0.03	0.4 1.0 1.0	70	1.89	0.45	0.55	0.96

Table 5. Collective average results for best EA configurations (2-point crossover + deterministic crowding selection) of both groups for all instances from all classes (* indicates that optimal solutions are unknown and gap to the best solution found by the EA is given). 95-percent confidence intervals for average gaps are given in brackets.

Solutions feasibility	calibration networks	class I networks	class II networks	class III networks
	avg. gap (%)	avg. gap (%)	avg. gap (%)	avg. gap (%)
No infeasible solutions	**0.64** (±0.07)	**0.41** (±0.03)	**0.34** (±0.02)	**0.55*** (±0.08)
Adaptive penalty function	**0.13** (±0.07)	**0.08** (±0.01)	**0.18** (±0.02)	**0.06*** (±0.03)

5.3 Results for all test instances

For each problem instance EA was executed 30 times and average gap was computed. In table 5 collective average results of best tuned EA configurations for all instances from all classes are displayed and compared to results from calibration phase. Results obtained during tuning procedures are consistent with those obtained for all test cases: better calibration results imply better overall results. EA configuration with adaptive penalty function produces extremely good overall results (average gaps of 0.08 and 0.19 percent) and is 0.2-0.5 percent better than EA without infeasible solutions. It can be seen that overall average gaps between algorithms for class I and II are smaller than gaps obtained for calibrated networks. It results from the fact that calibration was performed on larger instances (probably the hardest to obtain high-quality results) with 262-400 vertices while most test networks from class I and II had 100-200 nodes. Average gap between EAs is biggest for class III (large network of 908 cities). Average differences between two EA versions are statistically significant (confidence intervals don't overlap).

In table 6 results for all instances from class I are given for best EA configurations (with and without infeasible solutions). EA version with adaptive penalty is on average more than 0.3 percent better than EA without infeasible paths (results generated in similar execution times). The biggest difference between EAs was about 2 percent (pr144 network). The algorithm with adaptive penalty achieves optimal or nearly optimal solutions in all test cases and clearly outperforms GRASP and GRASPwPR metaheuristics (by 2.4 and 1.2 percent respectively).

In table 7 analogical results are given for instances from class II. EA with adaptive penalty performs better than its version without infeasible solutions (by almost 0.2 percent) and its execution times are shorter. The biggest difference between EAs is about 1.1 percent (cmt200B network). Both EAs clearly outperform GRASP and GRASPwPR (respectively by 1.7-1.9 and 3.1-3.3 percent).

In table 8 results for class III instances are presented. EA with penalty function is on average 0.5 percent better than EA without infeasible solutions and the biggest difference (1 percent) is obtained for longest paths (3000 km). EAs are on average 3-5 percent better than other metaheuristics. It can be seen that bigger gaps between algorithms are associated with larger test network (908 vertices).

Table 6. Detailed comparison of results of the best tuned EA configurations (2-point crossover + deterministic crowding) with results of GRASP. GRASPwPR and best known solutions (class I). Execution time is given in seconds.

Instance	$EA_{AdaptivePenalty}$			$EA_{NoInfeasibleSolutions}$			GRASP		GRASP PR		Best solution
	profit	gap (%)	time	profit	gap (%)	time	profit	gap (%)	profit	gap (%)	
kroA100	3180	0.03	1.1	3177.8	0.10	1.3	3135	1.45	3181	0.00	3181
kroB100	3187.1	0.25	1.1	3191	0.13	1.3	3183	0.38	3191	0.13	3195
kroC100	3043.3	0.02	1.5	3025.7	0.60	1.5	3044	0.00	3044	0.00	3044
kroD100	3223.3	0.08	1.3	3222.3	0.11	1.7	3152	2.29	3212	0.43	3226
kroE100	3310	0.00	1	3303.9	0.18	1.2	3260	1.51	3310	0.00	3310
rd100	3470	0.00	1.2	3448.7	0.61	1.1	3449	0.61	3453	0.49	3470
eil101	3668	0.00	1.2	3667.4	0.02	1.1	3596	1.96	3645	0.63	3668
lin105	3577	0.00	1.6	3576.7	0.01	1.4	3577	0.00	3577	0.00	3577
pr107	2681	0.00	1.2	2681	0.00	0.8	2681	0.00	2681	0.00	2681
gr120	4223	0.00	1.3	4198.5	0.58	1.4	4138	2.01	4201	0.52	4223
pr124	3840	0.00	2	3840	0.00	1.8	3840	0.00	3840	0.00	3840
bier127	5375	0.02	3.4	5374.7	0.02	4.1	5154	4.13	5254	2.27	5376
pr136	4221.2	0.04	2	4214.2	0.21	1.7	4170	1.26	4213	0.24	4223
gr137	4274.7	0.38	2.2	4272.1	0.44	1.7	4255	0.84	4284	0.16	4291
pr144	3989.1	0.12	2.5	3911.5	2.07	2	3902	2.30	3994	0.00	3994
kroA150	4919	0.00	2	4916.8	0.04	2.3	4768	3.07	4915	0.08	4919
kroB150	5017	0.00	1.8	5014.5	0.05	2.1	4967	1.00	5001	0.32	5017
pr152	4193.4	0.06	1.9	4192.4	0.09	1.9	4094	2.43	4175	0.50	4196
u159	5044	0.00	2.2	5028.3	0.31	2.3	4809	4.66	4987	1.13	5044
rat195	5933	0.05	2.5	5895.1	0.69	2.9	5693	4.09	5693	4.09	5936
d198	6537.9	0.02	3.9	6507.9	0.48	3.3	6347	2.94	6476	0.96	6539
kroA200	6611.9	0.06	4	6583.3	0.49	3.3	6447	2.55	6551	0.98	6616
kroB200	6596.2	0.01	2.5	6581.6	0.23	3.5	6357	3.64	6409	2.85	6597
ts225	6807.5	0.07	3.6	6732.1	1.17	4.2	6701	1.63	6784	0.41	6812
pr226	6690.7	0.00	6.5	6685	0.09	6.5	6375	4.72	6614	1.15	6691
gil262	9146.8	0.13	6.4	9135.8	0.25	5.6	8847	3.41	8941	2.38	9159
pr264	6657.8	0.12	3.9	6666	0.00	3.8	6666	0.00	6666	0.00	6666
pr299	9091.9	0.17	6	9010	1.07	6.3	8645	5.07	8689	4.59	9107
lin318	10902	0.55	8.4	10795.6	1.52	8.4	10074	8.10	10339	5.68	10962
rd400	13534.4	0.15	14.6	13461.3	0.69	16.2	12365	8.78	12365	8.78	13555
Avg.	**5431.5**	**0.08**	**3.2**	**5410.4**	**0.41**	**3.2**	**5256.4**	**2.49**	**5322.8**	**1.29**	**5437.2**

Table 7. Detailed comparison of results of the best tuned EA configurations (2-point crossover + deterministic crowding) with results of GRASP, GRASPwPR and best known solutions (class II). Execution time is given in seconds.

Instance	$EA_{AdaptivePenalty}$			$EA_{NoInfeasibleSolutions}$			GRASP		GRASPwPR		Best solution
	profit	gap (%)	time	profit	gap (%)	time	profit	gap (%)	profit	gap (%)	
eil101A	572	0.00	0.6	571.9	0.02	0.7	566	1.05	572	0	572
cmt121A	406.9	1.24	1	408.9	0.75	0.7	412	0	412	0	412
cmt151A	824	0.00	0.9	824	0.00	0.8	815	1.09	824	0	824
cmt200A	1202.8	0.18	1.7	1204.7	0.02	2	1145	4.98	1181	1.99	1205
gil262A	4508.9	0.02	1.9	4492.9	0.38	2.4	3916	13.17	4050	10.2	4510
eil101B	1049	0.00	1	1047.1	0.18	1.2	1024	2.38	1032	1.62	1049
cmt121B	708.9	0.85	1.3	712.7	0.32	1.5	699	2.24	707	1.12	715
cmt151B	1536.5	0.03	1.9	1535.7	0.08	2.1	1482	3.58	1528	0.59	1537
cmt200B	2196.5	0.07	5.5	2172.4	1.16	5.2	2073	5.69	2105	4.23	2198
gil262B	8449.6	0.08	6.3	8420.6	0.42	5.7	7946	6.03	8074	4.52	8456
eil101C	1336	0.00	1.9	1333.6	0.18	2.7	1295	3.07	1302	2.54	1336
cmt121C	1133.8	0.02	2.6	1129.5	0.40	2.1	1120	1.23	1125	0.79	1134
cmt151C	2001.9	0.05	4.3	1993.3	0.48	6	1965	1.9	1996	0.35	2003
cmt200C	2876.1	0.17	9	2873.3	0.27	10.9	2791	3.12	2824	1.98	2881
gil262C	11185.4	0.09	10.1	11153.6	0.37	13.3	10938	2.3	11046	1.33	11195
Avg.	**2665.9**	**0.18**	**3.3**	**2658.3**	**0.34**	**3.8**	**2545.8**	**3.46**	**2585.2**	**2.08**	**2668.5**

Table 8. Detailed comparison of results of the best tuned EA configurations (2-point crossover + deterministic crowding) with results of GRASP, GRASPwPR and best known solutions (class III). Execution time is given in seconds.

Instance	$EA_{AdaptivePenalty}$			$EA_{NoInfeasibleSolutions}$			GRASP		GRASPwPR		Best solution
	profit	gap	time	profit	gap(%)	time	profit	gap (%)	profit	gap (%)	
pl500	3735	0.11	1.2	3711.6	0.74	1	3637	2.73	3696	1.15	3739
pl1000	7936.2	0.06	3.8	7927.6	0.17	3.3	7598	4.32	7801	1.76	7941
pl1500	10379.4	0.04	5.5	10336.8	0.45	5.1	9576	7.78	9974	3.95	10384
pl2000	12119.7	0.05	7.1	12092.1	0.28	7.4	11739	3.19	11739	3.19	12126
pl2500	13574.2	0.04	11	13500.1	0.59	11.7	12654	6.82	12774	5.94	13580
pl3000	14956.6	0.06	16.6	14805.1	1.07	17	14022	6.31	14222	4.97	14966
Avg.	**10450.2**	**0.06**	**7.5**	**10395.6**	**0.55**	**7.6**	**9871.0**	**5.19**	**10034.3**	**3.49**	**10456**

In figure 1 the best path generated for a network of Polish cities is illustrated. All major cities (except Bialystok) are included in the path. A comparison between runs of EAs with static penalty (parameter k is always equal to 1) and adaptive penalty is shown in figures 2 and 3 (network pr144). Without adaptation all paths in the population exceeds length limit in further generations. This is rare among tested benchmark instances and usually signals convergence to very fit solutions violating infeasibility border. On the other hand, EA version with adaptive penalty reduces number of infeasible solutions allowing for further improvement (k parameter increased to 1.6).

Fig. 1. The best path found for a network of Poland ($C_{max} = 3000$ km). The route starts and ends in Warsaw, it's length is 2999.81 km and profit is 14966 (almost 15 million of inhabitants in visited cities).

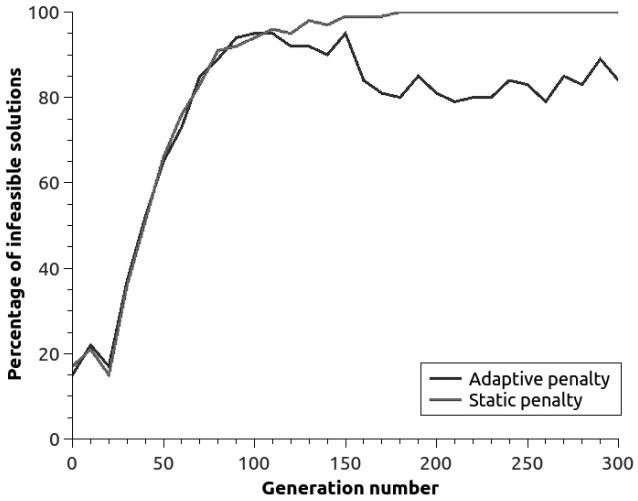

Fig. 2. EA runs with static and adaptive penalties (percentage of infeasible solutions in the population).

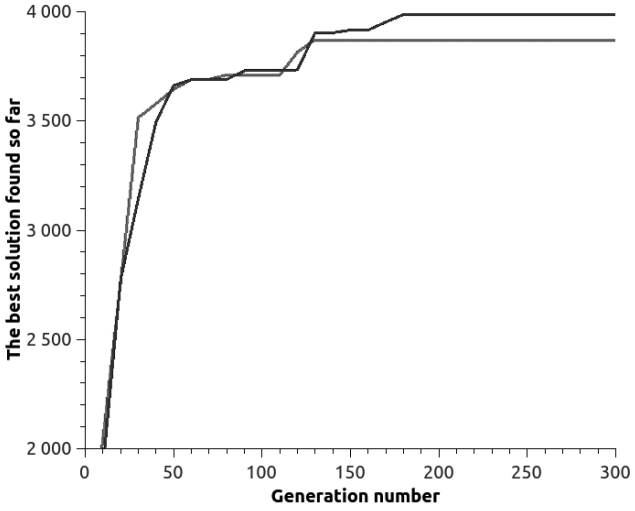

Fig. 3. EA runs with static and adaptive penalties (profit of best solution found).

6. Conclusions and further research

In the paper two strategies of dealing with solutions infeasiblity were compared for evolutionary algorithms solving the Orienteering Problem. Nine different algorithm configurations (varying in selection and crossover phases) were tuned and tested in two versions: with infeasible solutions (too long paths) in populations (adaptive penalty) and without them. Parameters of EAs were tuned with ParamsILS local search algorithm. It was found that all EA configurations with infeasible solutions outperformed their counterparts without too long paths in populations. The best EA configurations from both groups used 2-point crossover and deterministic crowding and the difference between them were on average about 0.2-0.5 percent (up to 1-2 percent in some cases). Best EA configuration with adaptive penalty obtained optimal or nearly optimal results for all test networks clearly outperforming other compared metehauristics (GLS, GRASP, GRASPwPR).

The author's further research is concentrated on OP versions more applicable to trip planning: the Time-Dependent Team Orienteering Problem with Time Windows (finding a multi-day tour in time-dependent graps i.e. public transport networks) and the Orienteering Problem with Hotel Selection (finding a tour divided into stages with acommodation in hotels).

Acknowledgment

The author gratefully acknowledges support from the Polish Ministry of Science and Higher Education at the Bialystok University of Technology (grant S/WI/1/2014).

References

[1] Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D.: The City Trip Planner: An expert system for tourists. Expert Systems with Applications, vol. 38(6), 6540-6546, 2011.

[2] Caserta, M., Voss, S.: A hybrid algorithm for the DNA sequencing problem. Discrete Applied Mathematics, vol. 163(1), 87-99, 2014.

[3] Golden, B., Levy, L., Vohra, R.: The orienteering problem. Naval Research Logistics, vol. 34, 307-318, 1987.

[4] Ostrowski, K.: Parameters tuning of evolutionary algorithm for the Orienteering Problem. Advances in Computer Science Research, vol. 12, 53-78, 2015.

[5] Tsiligirides, T.: Heuristic methods applied to orienteering. Journal of the Operational Research Society, vol. 35 (9), 797-809, 1984.

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

[7] Gendreau, M., Laporte, G., Semet, F.: A branch-and-cut algorithm for the undirected selective traveling salesman problem. Networks, vol. 32(4), 263-273, 1998.

[8] Chao, I., Golden, B., Wasil, E.: Theory and methodology - a fast and effective heuristic for the orienteering problem. European Journal of Operational Research, vol. 88, 475-489, 1996.

[9] Tasgetiren, M.: A genetic algorithm with an adaptive penalty function for the orienteering problem. Journal of Economic and Social Research, vol. 4 (2), 1-26. 2001.

[10] Gendreau, M., Laporte, G., Semet, F.: A tabu search heuristic for the undirected selective travelling salesman problem. European Journal of Operational Research, vol. 106, 539-545, 1998.

[11] Vansteenwegen, P., Souffriau, W., Vanden Berghe, G. and Oudheusden, D.V.: A guided local search metaheuristic for the team orienteering problem. European Journal of the Operational Research, vol. 196(1), 118-127, 2009.

[12] Schilde, M., Doerner, K., Hartl, R., Kiechle, G.: Metaheuristics for the biobjective orienteering problem. Swarm Intelligence, vol. 3, 179-201, 2009.

[13] Souffriau, W., Vansteenwegen, P., Vanden Berghe, G. and Oudheusden, D.V.: A path relinking approach for the team orienteering problem. Computers and Operations Research, vol. 37, 1853-1859, 2010.

[14] Campos, V., Marti, R.,Sanchez-Oro, J., Duarte, A.: Grasp with Path Relinking for the Orienteering Problem. Journal of the Operational Research Society, vol. 156, 1-14, 2013.

[15] Koszelew. J., Ostrowski. K.: A Genetic Algorithm with Multiple Mutation which Solves Orienteering Problem in Large Networks. Computational Collective Intelligence. Technologies and Applications, vol. 8083, 356-366, 2013.

[16] Zabielski. P., Karbowska-Chilinska, J., Koszelew, J., Ostrowski, K.: A Genetic Algorithm with Grouping Selection and Searching Operators for the Orienteering Problem. Lecture Notes in Artificial Intelligence, vol. 9012, 31-40, 2015.

[17] Ostrowski, K., Karbowska-Chilinska, J., Koszelew, J., Zabielski, P.: Evolution-inspired local improvement algorithm solving orienteering problem. Annals of Operations Research, vol. 253(1), 519-543, 2017.

[18] Ostrowski, K.: Comparison of Different Graph Weights Representations Used to Solve the Time-Dependent Orienteering Problem. Trends in Contemporary Computer Science, Podlasie 2014, Bialystok University of Technology Publishing Office, 144-154, 2014.

[19] Ostrowski, K.: Evolutionary Algorithm for the Time-Dependent Orienteering Problem. Proceedings from the Computer Information Systems and Industrial Management : 16th IFIP TC8 International Conference : CISIM 2017 (eds. Khalid Saeed, Wladyslaw Homenda, Rituparna Chaki). Lecture Notes in Computer Science, vol. 10244, 50-62, 2017.
[20] Ostrowski, K.: An Effective Metaheuristic for Tourist Trip Planning in Public Transport Networks. Applied Computer Science, vol. 14(2), 2018.
[21] Sokolov, A., Whitley, D.: Unbiased tournament selection. Proceedings of Genetic and Evolutionary Computation Conference. ACM Press, 1131-1138, 2005.
[22] Baker, J.E.: Reducing Bias and Inefficiency in the Selection Algorithm. Proceedings of the Second International Conference on Genetic Algorithms and their Application, Hillsdale, New Jersey: L. Erlbaum Associates, 14-21, 1987.
[23] Mahfoud, S.W.: Crowding and preselection revisited. Proceedings of the 2nd International Conference on Parallel Problem Solving from Nature (PPSN II), Brussels, Belgium, 1992. Elsevier, Amsterdam, The Netherlands, 27-36, 1992.
[24] Hutter, F., Hoos, H.H., Leyton-Brown, K., Stutzle, T.: ParamILS: an automatic algorithm configuration framework. Journal of Artificial Intelligence Research, vol. 36, 267-306, 2009.
[25] Network of 908 cities in Poland. http://p.wi.pb.edu.pl/sites/default/files/krzysztof-ostrowski/files/polska908.txt. Accessed 1 November 2018.

RÓŻNE METODY TRAKTOWANIA ROZWIĄZAŃ NIEDOPUSZCZALNYCH W ALGORYTMACH EWOLUCYJNYCH ROZWIĄZUJĄCYCH ORIENTEERING PROBLEM

Streszczenie Orienteering Problem (OP) należy do problemów optymalizacji kombinatorycznej i jest zdefiniowany na grafach ważonych. Celem OP jest znalezienie ścieżki o ograniczonej długości i maksymalnym łącznym proficie (zbieranym w wierzchołkach). Artykuł prezentuje porównanie różnych metod radzenia z rozwiązaniami niedopuszczalnymi (zbyt długimi ścieżkami) w algorytmach ewolucyjnych rozwiązujących OP. Grupa algorytmów ewolucyjnych (różniących się operatorami selekcji i krzyżowania) została przetestowana w dwóch konfiguracjach: z osobnikami dopuszczalnymi w populacji oraz bez nich. Wartości parametrów algorytmów zostały ustawione za pomocą automatycznej procedury kalibracji

(ParamILS). Wyniki wskazują, że obecność zbyt długich ścieżek w populacji może poprawić jakość rozwiązań. Prezentowana meta-heurystyka uzyskiwała rozwiązania optymalne lub bliskie optymalnym dla sieci testowych.

Słowa kluczowe: rozwiązania niedopuszczlne, algorytmy ewolucyjne, Orienteering Problem