Applied Computer Science, vol. 14, no. 2, pp. 5–19
doi:10.23743/acs-2018-09

Submitted: 2018-05-14
Revised: 2018-06-03
Accepted: 2018-06-21

time-dependent orienteering problem with time-windows, evolutionary algorithm, public transport network, tourist trip planning

Krzysztof OSTROWSKI[*]

AN EFFECTIVE METAHEURISTIC FOR TOURIST TRIP PLANNING IN PUBLIC TRANSPORT NETWORKS

Abstract

The Time-Dependent Orienteering Problem with Time Windows (TDOPTW) is a combinatorial optimization problem defined on graphs. Its real life applications are particularly associated with tourist trip planning in transport networks, where travel time between two points depends on the moment of travel start. In the paper an effective TDOPTW solution (evolutionary algorithm with local search operators) was presented and applied to generate attractive tours in real public transport networks of Białystok and Athens. The method achieved very high-quality solutions in a short execution time.

1. INTRODUCTION

The Time-Dependent Orienteering Problem with Time Windows (TDOPTW) belongs to the Orienteering Problem (OP) family. The classic OP is defined on a weighted graph with nonnegative profits associated to vertices and nonnegative costs associated to edges. The goal of the OP is to find a path between given two vertices that maximizes total profit of visited vertices and its total cost does not exceed a given limit. The OP solution does not have to contain all vertices (usually it is impossible because of cost limit) and each vertex can be visited only once.

The Time-Dependent Orienteering Problem with Time Windows (TDOPTW) is a generalization of the OP defined for time-dependent graphs. Edge costs are identified with travel times, which depend on the moment of travel start (edge

[*] Faculty of Computer Science, Białystok University of Technology, Wiejska 45A, 15-001 Białystok, Poland, k.ostrowski@pb.edu.pl

weights are time-dependent functions). In addition each vertex has a visit time and a time-window. Arriving too early means waiting for the time-window to open while arriving too late makes it impossible to visit a given vertex.

Transport networks are examples of time-dependent graphs are. Travel time between two points depends on traffic intensity (i.e. longer during rush hours) and timetables (in case of public transport networks). Problems from the TDOPTW family have many practical applications including tourist trip planning (Garcia, Vansteenwegen, Arbelaitz, Souffriau & Linaza, 2013) and transport logistics. In tourist trip planning each tourist attraction (point of interest – POI) has some profit (i.e. dependent on its popularity), visit time and opening hours (time-window). Finding an attractive tour of a limited duration in a time-dependent transport network is equivalent to solving the TDOPTW.

The paper is organized as follows. In section 2 a mathematical formulation of the TDOPTW is given. In section 3 a literature review is presented. Section 4 describes public transport network as a time-dependent graph. Section 5 describes methods applied. In section 6 experimental results are given. Section 7 is the conclusion of the paper.

2. PROBLEM DEFINITION

Let $G = (V, E)$ be a directed, weighted graph. Each vertex i has a nonnegative profit p_i, a nonnegative visit time τ_i and a time-window $<o_i, c_i>$. Travel time between vertices i and j ($i, j \in V$) is a nonnegative function $w_{ij}(t)$ dependent on the moment of travel start t. The goal of the TDOPTW is to find a path from vertex s to vertex e starting at time t_0 which maximizes total profit of visited vertices without exceeding total travel time (T_{max}) and without violating time-windows of visited vertices. TDOPTW can be formulated as a Mixed Integer Programming (MIP) problem. Let's introduce three additional variables. Variable x_{ij} is 1 if a path contains a direct travel from vertex i to vertex j and 0 otherwise. Let ta_i and td_i be a time of arrival at vertex i and a time of departure from it – these functions are defined only for vertices included in the path. It's assumed that vertices s and e have no profit, zero visit time and no time-window. The purpose of the TDOPTW is to maximize formula 1 without violating constraints 2–8:

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

$$\sum_{j \in V} x_{sj} = \sum_{i \in V} x_{ie} = 1 \tag{2}$$

$$\bigvee_{k \in V \setminus \{s,e\}} \left(\sum_{i \in V} x_{ik} = \sum_{j \in V} x_{kj} \leq 1 \right) \tag{3}$$

$$td_s = t_0 \tag{4}$$

$$\bigvee_{i,j \in V} \left(x_{ij} = 1 \Rightarrow ta_j = td_i + w_{ij}(td_i) \right) \tag{5}$$

$$\bigvee_{i \in V \setminus \{s,e\}} (td_i = \max(ta_i, o_i) + \tau_i) \tag{6}$$

$$ta_e - t_0 \leq T_{\max} \tag{7}$$

$$\bigvee_{i \in V \setminus \{s,e\}} (ta_i \leq c_i) \tag{8}$$

Equation 2 guarantees that the solution starts at vertex *s* and ends at vertex *e* while formula 3 means that each vertex can be visited only once. Formula 4 forces travel to start at time t_0 while formulas 5 and 6 defines a relation between arrival and departure times at subsequent vertices (based on travel times, visit times and time-windows). Assuming that travel times between different vertices are positive, formulas 4–6 guarantee that there are no sub-cycles in the path. Constraints 7 and 8 are associated with maximum travel time and time-windows.

3. LITERATURE REVIEW

Problems from the Orienteering Problem family are NP-hard (Golden, Levy, Vohra, 1987) and exact algorithms can be very time-consuming for larger graphs. For this reason most papers are devoted to metaheuristics. Various approaches for the OP were based i.a. on greedy and randomized construction of solutions (Campos, Marti, Sanchez-Oro & Duarte, 2014), local search methods (Chao, Golden & Wasil, 1996; Vensteenwegen, Souffriau, Vanden Berghe & Oudheusden, 2009), tabu search (Gendreau, Laporte & Semet, 1998), ant-colony optimization (Schilde, Doerner, Hartl & Kiechle, 2009) and genetic algorithms (Tasgetiren, 2001).

Most papers about Time-Dependent versions of the Orienteering Problem were published in recent years and emphasize practical aspects of the problem, especially tourist trip planning in transport networks. Garcia et al. (2013) presented the first paper describing its application in POI and public transport network of San Sebastian. To solve the problem the authors proposed Iterated Local Search method (ILS). However, they performed computations on average daily travel times and assumed periodicity of public transport timetables.

Gavalas et al. (2015) proposed an approach which uses real time-dependent travel times in a transport network of Athens. The authors introduced two fast heuristics (TD_CSCR and TDSlCSCR), which based on ILS and vertex clustering, and made comparisons of a few methods.

Verbeeck at al. (2014) developed new benchmark instances for the TDOP, which model street traffic. The authors proposed an ant-colony approach, which achieved high quality results in a short execution time. Gunawan et al. (2014) modified Verbeeck's benchmarks (discretization of time) and compared a few approaches (adaptive ILS proved to be the most effective of them).

The author's previous papers were devoted to metaheuristics for problems from the OP family. Methods developed by the author (composition of evolutionary algorithms and local search heuristics) proved successful on the OP (Ostrowski, Karbowska-Chilinska, Koszelew & Zabielski, 2017; Ostrowski, 2015) as well as TDOP benchmark instances (Ostrowski, 2017). The algorithms achieved results close to optimal and outperformed other methods: GRASP (Campos et al, 2014), GLS (Vensteenwegen et al., 2009), ACS (Verbeeck at al 2014) and Adaptive ILS (Gunawan et al, 2014). The purpose of this work was to adapt the TDOP algorithm to the TDOPTW, apply it for tourist trip planning in real public transport and POI network and verify quality of its solutions.

4. PUBLIC TRANSPORT AND POI NETWORK AS A TIME-DEPENDENT GRAPH

It's assumed that a tourist uses means of public transport (buses in the city of Białystok) when travelling between attractions (POIs). Travel time between POIs depends on bus timetables. For this reason a network of POIs connected by public transportation is a time-dependent graph. Here are assumptions made by the author:

1. A travel between two POIs can consist of two kinds of edges: walk links and bus connections.
2. Walk links have a limited length (D_{max}) and walk times are determined by assuming that walking speed is 3 km/h. A tourist can walk directly between POIs, between POIs and bus stops and between bus stops (during bus transfers).
3. During a travel between a pair of POIs a tourist can use up to k bus transfers (k+1 bus connections).
4. To compensate for deviations of bus arrival times from timetables a minimum waiting time at a bus stop (3 minutes) was introduced.
5. Each graph weight $w_{ij}(t)$ is the duration of the shortest travel from POI i to POI j starting at time t. To compute weights it's necessary to execute shortest path algorithm in multimodal time-dependent graph.

6. Time is discrete (resolution of 1 minute), which is consistent with time-tables. For this reason there are $1440n^2$ graph weights (*n* is number of POIs and there are 1440 minutes in a day).

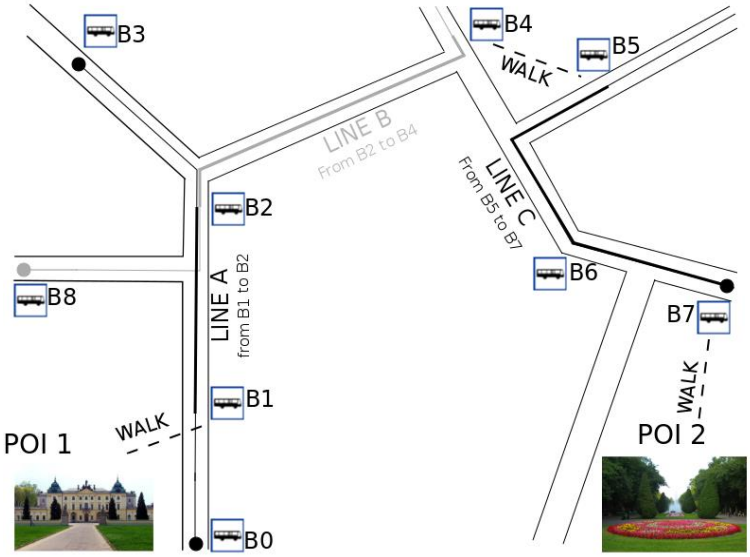

Fig. 1. An exemplary travel between two POIs

In fig. 1. there is an exemplary travel between two POIs consisting of 3 bus connections (2 transfers) and 3 walk links. A tourist leaves POI 1, walks to bus stop B1 and gets on a bus (line A). He gets off at bus stop B2 and waits there for another bus (line B). Afterwards he goes by bus B to bus stop B4. From there he walks to a nearby bus stop B5, where he gets onto another bus (line C) and travels to bus stop B7. From there he walks to his destination (POI 2). Computation of such shortest paths is necessary to get time-dependent weights, which will be used by the metaheuristic solving the Time-Dependent Orienteering Problem.

5. METHOD DESCRIPTION

To realize tourist trip planning in a public transport network of Białystok two tasks should be done:

1. Computation of time-dependent weights based on bus timetables and POIs location.
2. Execution of the TDOP algorithm, which operates on time-dependent weights and generates attractive tours.

5.1. Precomputation of weights

During its execution the TDOP metaheuristic refers to graph weights millions of times. Computing shortest path in a time-dependent network so many times can cause an additional time overhead. For this reason the author decided to precompute and save all $1440n^2$ weights. Precomputed weights are stored in a 3-dimensional array. Thanks to the precomputation step the TDOP algorithm has access to all weights in constant time.

Shortest travels from a given POI at a given start time to all other POIs

In order to compute all weights efficiently the author decided to use modified Ford-Bellman algorithm and optimize some precomputation steps. The basic Ford-Bellman procedure computes shortest travels starting at time t_0 from a given start POI s to all other POIs and bus stops. The algorithm has k+1 main iterations (k – number of bus transfers). It enables to efficiently compute shortest paths consisted of a limited number of bus connections. During the first iteration bus connections starting at bus stops not farther than max. walk distance (D_{max}) from s are considered. Only earliest possible buses of given lines are considered. Afterwards all possible bus stops, where a tourist can get off the bus (exit bus stops), are analysed. From there destination POIs (within walking distance from exit bus stops) are checked. In this way all shortest travel times (consisting of one transport connection) are computed. Analogically the second iteration computes all shortest travels consisted of at most two transport connections (one bus transfer): when analysing an exit bus stop the algorithm searches for other bus connections leaving from the current bus stop and from other neighbouring bus stops (transfer step). The same steps are performed for subsequent iterations. For optimization purposes the algorithm only analyses those bus stops, for which travel time improved in the previous iteration.

Shortest travels from a given POI for all start times

The purpose of precomputation is to compute shortest travels for all 1440 start times (minutes) in a day. Instead of executing the same shortest path algorithm (described above) 1440 times an optimization can be done. It arises from a simple observation (known as FIFO property): $t + w_{ij}(t) \leq (t+1) + w_{ij}(t+1)$. The formula means that earlier travel start implies not later travel finish – in the most pessimistic case travel starting at time t will use the same bus connections as travel starting at time t+1 (the only difference is one more minute of waiting at the first bus stop). Thanks to this property shortest travels for consecutive start times can be computed much faster. When computing shortest travels starting at time t the algorithm uses shortest travel times previously computed for start time t+1. The algorithm considers only those bus connections, which were impossible

to catch one minute later. This significantly reduces precomputation time, especially when frequency of bus connections is less than a minute (which is common in public transport networks – see fig. 2). In the below figure there is an example of time-dependent arrival times: the function is nondecreasing (and constant in intervals). Inside these intervals an equality $t + w_{ij}(t) = (t+1) + w_{ij}(t+1)$ holds, which usually means that the same set of connections is used and computation time can be reduced. What is more, the FIFO property in time-dependent networks implies existence of polynomial time shortest-paths algorithms (Dean, 2004), which made it possible to develop fast precomputation procedure.

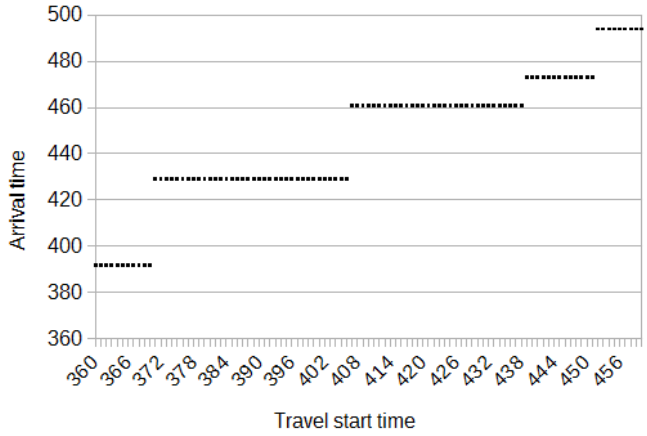

Fig. 2. Arrival time at POI *j* as a function of departure time from POI *i*

Shortest travels between all POIs for all start times

To compute all possible weights for all 1440 start times the above procedures are executed for all starting POIs. Thanks to the described optimization precomputation time was significantly reduced.

5.2. TDOPTW metaheuristic for tourist trip planning

To solve the trip planning problem the author used an evolutionary algorithm, which is based on the method solving the TDOP (Ostrowski, 2017). The author adapted the method to the presence of time-windows. It uses both random and local search operators, 2-point heuristic crossover, disturb operator and deterministic crowding as selection mechanism. What is more, infeasible solutions (too long paths) are present in the population (penalized by the fitness function). A path representation is used – subsequent genes in a chromosome correspond to successive vertices in a path. After a random initialization the algorithm continues computations until a given generations limit (N_g) is achieved or there was no solution improvement in the last C_g generations.

Evaluation

Fitness of a feasible solution s is equal to its profit. Otherwise it is described by the formula: $fitness(s) = p(s) \cdot [T_{max}/t(s)]^k$, where $p(s)$ and $t(s)$ are profit and travel time of solution s. Parameter k (penalty severity) is initially equal to 1 but it is adaptive and increases if more than α percent of paths in the population are infeasible.

Crossover

Parent selection is random. Crossover probability determines the fraction of population chosen for reproduction (selected individuals arranged in random pairs). The algorithm uses specialized 2-point heuristic crossover. Crossover procedure exchanges one pair of path fragments between consecutive common points of both parents. In fig. 3. an example of crossover is illustrated. There are three possible crossings (varying in exchanged fragments). Heuristic crossover chooses the option which maximizes fitness of the better child.

Selection

After crossover children compete with their own parents for places in the population – survivor selection in the form of deterministic crowding (Mahfoud, 1992). Distance metric used bases on the length of longest common subsequence of two solutions. This form of selection preserve population diversity for longer, which allows a more effective search of the solution space (Ostrowski, 2015).

Mutation

Mutation probability determines the fraction of individuals which are selected (randomly) for mutation. Initially, selected paths undergo 2-opt procedure, which tries to reverse a path fragment in order to reduce total travel time as much as possible. Afterwards a vertex insertion or vertex deletion is carried out (each with a probability of 0.5). Both *insert* and *delete* operators have two versions: local search and random. Local search *insert* from all options of inserting a new vertex chooses the one that maximizes profit to travel time increase ratio. Analogically *delete* searches for a vertex which minimizes profit to travel time reduction ratio. Random versions choose vertices arbitrary but insertion place is chosen in order to minimize travel time increase. Probability of local search during mutation is determined by a parameter (heuristic coefficient).

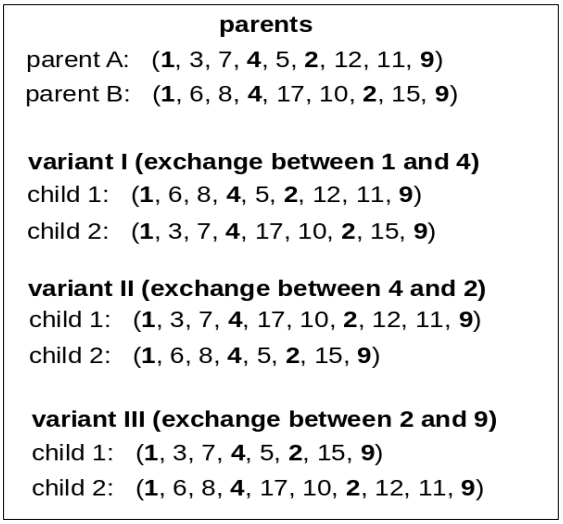

Fig. 3. An example of crossover

Disturb

Disturb procedure is another form of mutation, which applies bigger changes in individuals but is executed rarely. A small fraction of population (determined by disturb probability) is randomly chosen and a path fragment (no longer than 10 percent of vertices) is removed from each of them. Path fragment is chosen randomly or in a heuristic way (to minimize profit to travel time reduction ratio).

Time-Windows

All operators used in the algorithm were modified in order to take into account time-windows. Given a POI with a time-window $<o, c>$ and arrival time t, arriving too early ($t<o$) means that additional waiting time ($o-t$) was added to the tour duration. Arriving too late ($t>c$) made it impossible to visit a vertex and such cases weren't allowed by the algorithm operators.

6. EXPERIMENTAL RESULTS

Experiments were conducted on a computer with Intel Core i7 3.5 GHz processor and the algorithms were implemented in C++. First part of experiments was devoted to precomputation of travel times between POIs in public transport network of Białystok and in the second part the TDOPTW metaheuristic was applied for tourist trip planning in this network (using precomputed weights).

6.1. Precomputation and network properties

Public transport and POI network of Białystok consists of 74 POIs (museums, palaces, churches etc), 37 bus lines and 693 bus stops. Thanks to optimizations (described in the previous section) precomputation time was only a few seconds. Shortest travel times between all pairs of POIs for all start times (1440 minutes in a day) were stored in a 3-dimensional array (occupying 18 MB of RAM). To find out about interesting features of the network, precomputation was executed many times for different values of parameters: maximal walk distance (D_{max}) and maximal number of bus transfers (k). Tests were conducted for two D_{max} values (0.3 and 0.6 km) and four k values (0, 1, 2, 3).

In fig. 4. a dependency between percentage of connected POIs and pre-computation parameters is illustrated. It can be seen that large majority of POI pairs are connected when travel consists of at most one bus transfer (two transport connections) and there is no connectivity improvement for more than 2 bus transfers. Connection percentage improves for larger value of D_{max}. Longer walk links enable to reach larger number of bus stops, which naturally implies more connection options. One can see that for shorter walk links connectivity is always less than 100 percent (regardless of number of bus transfers). This is due to the fact that a few POIs were farther than 0.3 km from nearest bus stop. In fig. 5. it can be seen that most of shortest travels between POIs are very simple (no bus transfer or one transfer) and almost no paths consist of more than 2 transfers.

This is due to the fact that Białystok is a relatively compact city. Larger D_{max} value influences paths simplicity for the same reason as it influenced connections percentage.

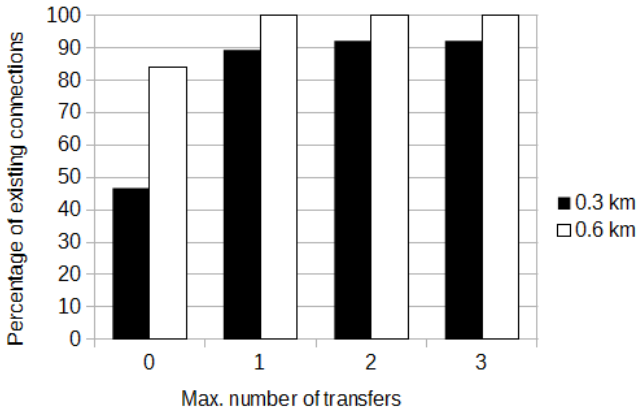

Fig. 4. Percentage of connected POI pairs depending on D_{max} and k

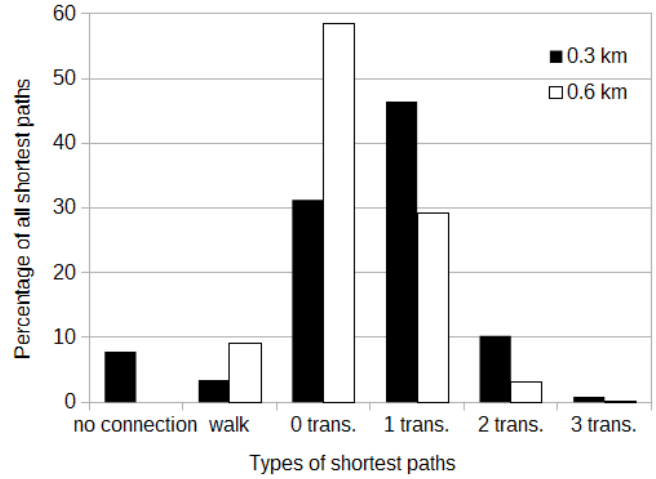

Fig. 5. Percentage of different types of shortest paths depending on D_{max}

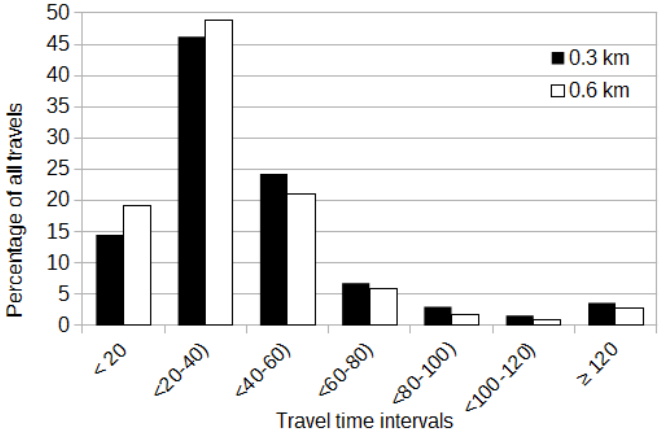

Fig. 6. Histogram of travel times between POIs for different D_{max} values (computed for daily hours 6:00–18:00).

In fig. 6. it can be seen that travel times of 20–40 minutes are most common (46–49 percent) and the majority of travels last less than an hour. In addition, there are more short travels and less long travels when increasing D_{max}. It is associated with the fact that for larger D_{max} value travel times are generally shorter (more bus connections are considered when searching for paths) – average daily travel time is 45 and 41 minutes (for D_{max} 0.3 and 0.6 accordingly).

6.2. Trip planning in Białystok

The tested network included 74 POIs and 2 start/end points. For each attraction a profit, a visit time and a time-window (opening hours) were assigned (link to network: http://p.wi.pb.edu.pl/krzysztof-ostrowski/node/1252). Trips were 3, 6 and 9 hours long (T_{max}) and started at 6:00, 9:00, 12:00 and 15:00 (t_0). Time unit used in test files as well as during computations was a minute i.e. 9:00 is 540 and 4 hours are 240 minutes. There were two variants of trips: starting and ending in the city centre ($s = e = 1$) and in the western part of the city ($s = e = 2$). The author tested two methods: the TDOPTW metaheuristic (evolutionary algorithm with local search heuristics) as well as an exact algorithm (composition of branch-and-bound and dynamic programming developed by the author). Thanks to the exact algorithm optimal solutions are known (up to a few hours of computation time for longest trips) and it's possible to access the quality of paths generated by the metaheuristic.

Algorithm parameter values were derived from EVO100 in the author's previous TDOP article (Ostrowski, 2017) with a small change: $m_h = 0.8$ and $c_h = 1$ instead of $m_h = 1$ and $c_h = 0.8$ (minor error in the article). For each test case the evolutionary algorithm was executed 30 times and average result was calculated. Gaps are given in percent and illustrate relative quality loss to optimal solutions. Execution times are given in seconds. The author's metaheuristic is marked as EVO while OPT indicate profits of optimal tours (expressed as the sum of attractiveness of visited POIs).

In tab. 1. results of trip planning are given (trips start and end in the city centre). One can see that EVO achieves optimal results in most cases and average gap is only 0.02 and 0.19 percent (for D_{max} = 0.3 and 0.6 km). High-quality results are achieved in short execution times (0.3–1.4 s). It can be seen that optimal trips are a few percent better for larger value of D_{max}. This is due to shorter travel times when using longer walk links (as described in the previous subsection), which enables to visit more POIs within a given time frame.

Tab.1. Trip planning results for $s = e = 1$

D_{max} = 0.3 km					D_{max} = 0.6 km				
		EVO					EVO		
T_{max}	t_0	Gap	Time	OPT	T_{max}	t_0	Gap	Time	OPT
3h	**6:00**	0.0	0.3	327	**3 h**	**6:00**	0.0	0.5	342
	9:00	0.0	0.4	475		**9:00**	0.0	0.5	475
	12:00	0.0	0.4	475		**12:00**	0.0	0.5	512
	15:00	0.0	0.5	485		**15:00**	1.6	0.5	506
6 h	**6:00**	0.0	0.9	741	**6 h**	**6:00**	0.1	0.8	769
	9:00	0.0	0.8	813		**9:00**	0.0	0.8	878
	12:00	0.1	0.9	850		**12:00**	0.0	0.7	900
9 h	**6:00**	0.1	1.4	1089	**9 h**	**6:00**	0.0	1.2	1144
	9:00	0.0	1.1	1178		**9:00**	0.0	1.2	1218
Average		**0.02**	**0.75**	–	**Average**		**0.19**	**0.75**	–

In tab. 2. analogical results are presented for trips starting and ending in the western part of the city ($s = e = 2$). Optimal solutions were obtained by EVO for all but 2 test cases in short execution times. Trips quality is lower than it the previous table because most attractions are located in the city centre and additional time is needed to get there.

In fig. 7. a trip generated by the algorithm is presented. It is short and all visited POIs are in the city centre so only one bus connection is needed. It worth noting that the algorithm usually chooses consecutive POIs which are close to each other in order to use the time budget effectively. For this reason travels between POIs found in solutions are usually short (only 5–15 minutes, compared to average of 40–45 minutes) and simple (a walk link or a single bus connection).

Tab. 2. Trip planning results for $s = e = 2$

***Dmax* = 0.3 km**					***Dmax* = 0.6 km**				
		EVO					**EVO**		
T_{max}	t_0	**Gap**	**Time**	**OPT**	T_{max}	t_0	**Gap**	**Time**	**OPT**
3 h	**6:00**	0.0	0.3	226	**3 h**	**6:00**	0.0	0.4	226
	9:00	0.0	0.4	338		**9:00**	0.0	0.4	350
	12:00	0.0	0.4	338		**12:00**	0.0	0.4	338
	15:00	0.0	0.4	358		**15:00**	0.0	0.4	375
6 h	**6:00**	0.0	0.7	659	**6 h**	**6:00**	0.0	0.6	677
	9:00	0.0	0.7	708		**9:00**	0.5	0.7	721
	12:00	0.0	0.8	786		**12:00**	0.0	0.7	805
9 h	**6:00**	0.0	1.1	996	**9 h**	**6:00**	0.1	1.2	1024
	9:00	0.0	1.1	1125		**9:00**	0.0	1.0	1162
Average		**0.0**	**0.65**	–	**Average**		**0.07**	**0.65**	–

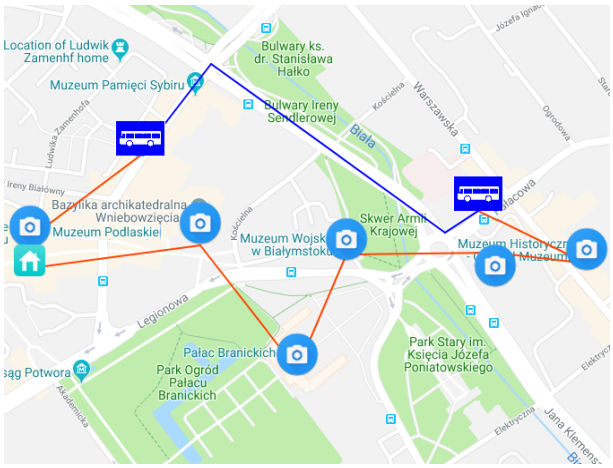

Fig. 7. A tour generated by the algorithm (Google Maps): walk links in red, bus connections in blue; icons: camera (POI), bus (bus stop), house (start/end), T_{max} = 3h, t_0 = 9:00.

6.3. Trip planning in the city of Athens

Additional tests were performed on public transport and POI network in Athens (tests created by Gavalas et al, 2015). The authors proposed 2000 different test cases (varying in topology and tourist preferences) and the presented results are average of all 2000 algorithm runs. The evolutionary algorithm was compared with the following heuristics:

1. Time-dependent heuristics: TD_CSCR, TD_SlCSCR (Gavalas et al, 2015) and their version working on average travel times (AvgCSCR).
2. ILS algorithm working on average travel times (AvgILS, Garcia et al, 2013) and its time-dependent version (TD_ILS).
3. Exact algorithm implemented by the author (OPT).

Compared methods were very fast (execution times of less than 0.1 s) and in order to achieve similar execution times the author tested another version of the evolutionary algorithm with reduced population size (EVO30). Population size and generation parameters were scaled (P_{size} = 30, N_g = 1500, C_g = 150).

In tab. 3. experimental results are presented. It can be seen that the proposed evolutionary algorithm (in both versions) achieves results very close to optimal. They are 0.6–2.0 percent better than the best of other metaheuristics (TD_SlCSCR and TD_CSCR). Gaps to other methods are much bigger (3.5–13.5 percent). It confirms effectiveness of the proposed method.

Tab. 3. Trip planning results for the city of Athens (T_{max} = 5h, t_0 = 10:00)

Method	Score	Gap	Method	Score	Gap
EVO100	344.57	0.02	**Avg_CSCR**	332.01	3.7
EVO30	344.34	0.1	**TD_ILS**	326.28	5.3
TD_SlCSCR	342.06	0.7	**Avg_ILS**	298.53	13.4
TD_CSCR	337.78	2.0	**OPT**	344.6	–

7. CONCLUSION

In this paper a metaheuristic solving the Time-Dependent Orienteering Problem with Time Windows (TDOPTW) was presented and applied to tourist trip planning in public transport networks. The algorithm was tested on public transport and POI networks of Białystok and Athens and in all cases obtained optimal or close to optimal solutions (tours) in short execution times. The composition of evolutionary algorithm and local search heuristics confirmed to be effective for the problems from the Orienteering Problem family (high-quality results were previously obtained by the author for the OP and TDOP benchmarks). Further research will concentrate on adaptation of the proposed method to the Time-Dependent Team Orienteering Problem with Time Windows (TDTOPTW). In this version of the problem *m* paths are generated (instead of one) and effective TDTOPTW solutions can be applied to planning multi-day tours.

REFERENCES

Campos, V., Marti, R., Sanchez-Oro, J., & Duarte, A. (2014). Grasp with Path Relinking for the Orienteering Problem. *Journal of the Oper. Res. Society*, *65*(12), 1800–1813. doi:10.1057/jors.2013.156

Chao, I., Golden, B., & Wasil, E. (1996). Theory and methodology - a fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, *88*(3), 475–489. doi:10.1016/0377-2217(95)00035-6

Dean, B.C. (2004). *Shortest paths in FIFO time-dependent networks: theory and algorithms.* Technical report, MIT Department of Computer Science.

Garcia, A., Vansteenwegen, P., Arbelaitz, O., Souffriau, W., & Linaza, M. T. (2013). Integrating public transportation in personalised electronic tourist guides. *Computers and Operations Research*, *40*(3), 758–774. doi:10.1016/j.cor.2011.03.020

Gavalas, D., Konstantopoulos, C., Mastakas, K., Pantziou, G., & Vathis, N. (2015). Heuristics for the time dependent team orienteering problem: Application to tourist route planning. *Computers and Operation Research*, *62*, 36-50. doi:10.1016/j.cor.2015.03.016

Gendreau, M., Laporte, G., & Semet, F. (1998). A tabu search heuristic for the undirected selective travelling salesman problem. *European Journal of Operational Research*, *106*(2–3), 539–545. doi:10.1016/S0377-2217(97)00289-0

Golden, B., Levy, L., & Vohra, R. (1987). The orienteering problem. *Naval Research Logistics*, *34*, 307-318. doi:10.1002/1520-6750(198706)34:3<307::AID-NAV3220340302>3.0. CO;2-D

Gunawan, A., Yuan, Z., & Lau, H. C. (2014). A Mathematical Model and Metaheuristics for Time Dependent Orienteering Problem. In *PATAT 2014: Proceedings of the 10th International Conference of the Practice and Theory of Automated Timetabling, 26–29 August 2014* (pp. 202–217). Research Collection School Of Information Systems.

Mahfoud, S. W. (1992). Crowding and preselection revisited. In *Proceedings of the 2nd International Conference on Parallel Problem Solving from Nature (PPSN II), Brussels, Belgium, 1992* (pp. 27–36). Amsterdam: Elsevier.

Ostrowski, K. (2015). Parameters Tuning of Evolutionary Algorithm for the Orienteering Problem. *Advances in Computer Science Research*, *12*, 53–78.

Ostrowski, K., Karbowska-Chilinska, J., Koszelew, J., & Zabielski, P. (2017). Evolution-inspired local improvement algorithm solving orienteering problem. *Annals of Operations Research*, *253*(1), 519-543. doi:10.1007/s10479-016-2278-1

Ostrowski, K. (2017). Evolutionary Algorithm for the Time-Dependent Orienteering Problem. In K. Saeed, W. Homenda, & R. Chaki (Eds.), *Computer Information Systems and Industrial Management. CISIM 2017, Lecture Notes in Computer Science* (10244, pp. 50–62). Cham: Springer. doi:10.1007/978-3-319-59105-6_5

Schilde, M., Doerner, K., Hartl, R., & Kiechle, G. (2009). Metaheuristics for the biobjective orienteering problem. *Swarm Intelligence*, *3*(3), 179–201. doi:10.1007/s11721-009-0029-5

Tasgetiren, M. (2001). A genetic algorithm with an adaptive penalty function for the orienteering problem. *Journal of Economic and Social Research*, *4*(2), 1–26.

Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., & Oudheusden, D.V. (2009). A guided local search metaheuristic for the team orienteering problem. *European Journal of the Operational Research*, *196*(1), 118–127. doi:10.1016/j.ejor.2008.02.037

Verbeeck, C., Sörensen, K., Aghezzaf, E.H., & Vansteenwegen, P. (2013). A fast solution method for the time-dependent orienteering problem. *European Journal of Operational Research*, *236*(2), 419–432. doi:10.1016/j.ejor.2013.11.038