Genetic Algorithm with Path Relinking for the Orienteering Problem with Time Windows

Joanna Karbowska-Chilinska and Pawel Zabielski

Faculty of Computer Science
Bialystok University of Technology, Poland

Abstract. The Orienteering Problem with Time Windows (OPTW) is an optimisation NP-hard problem. This paper proposes a hybrid genetic algorithm (GAPR) for approximating a solution to the OPTW. Instead of the usual crossover we use a path relinking (PR) strategy as a form of intensification solution. This approach generates a new solution by exploring trajectories between two random solutions: genes not present in one solution are included in the other one. Experiments performed on popular benchmark instances show that the proposed GAPR outperforms our previously published version of GA and yields better results than the well-known iterated local search method (ILS) as well.

Keywords: orienteering problem with time windows, genetic algorithm, path relinking

1 Introduction

The Orienteering Problem with Time Windows (OPTW) is a type of optimisation routing problem first introduced by Kantor et al. [7]. The OPTW can be modelled as a weighted graph with a positive score/profit associated with each vertex. Let G be a graph with n vertices, in which each vertex i has a profit p_i, a service time T_i and a time window $[O_i, C_i]$, where O_i and C_i denote the opening and closing times of a vertex i. Each edge between vertices i and j has a fixed cost t_{ij} associated with it. The value t_{ij} is interpreted as the time or length needed to travel between vertices. The objective is to determine a single route, from a starting point s to a fixed ending point e, that visits some of the vertices within the fixed time windows and maximises the total profit. In addition, the total cost of the edges on the path must be less than the given constraint t_{max} and each vertex on the route is visited only once. It is possible to wait at a vertex for service before its time windows opens.

The OPTW is derived from the more general Orienteering Problem (OP) [23], [14]. In the OP each vertex inserted into the route could be visited in any time interval (there is no restriction in the form of time windows). The OP is seen as a combination of the Knapsack Problem and the Travelling Salesperson Problem, because in the OP the selected route limited in length contains the most profitable vertices. Both the OP and the OPTW are NP-hard [7].

Numerous applications can be found for the OPTW, e.g. in logistics for planning optimal routes, such as profitable delivery routes, as well as in optimisation of production scheduling [20]. The OPTW successfully models problems related to tourism [24]. Tourists visiting a city are usually unable to visit all points of interests (POI) because they are limited by time or money. The most effective heuristics for the OPTW are applied in electronic devices known as mobile tourist guides [24], [2] which make it possible to visit the most valuable POIs (taking into account their opening and closing times) within a fixed time limit. The Team Orienteering Problem with Time Windows (TOPTW) [12], which is an extension of the OPTW, is used to model multiple tours, with each tour satisfying the same fixed travel length or time constraint.

In this paper we present an improved version of the genetic algorithm for the OPTW described in [10]. We use hybridization of our genetic algorithm with a path relinking method (PR) instead of the crossover operator. In the PR approach two random solutions are chosen and routes combining these solutions are explored to provide better solutions: genes not present in one solution are included in the other one. Path relinking was originally described by Glover [5] and Laguna [6] for intensification and diversification of the tabu search method. Moreover, the PR significantly improves the results of the Greedy Randomised Adaptive Search Procedure (GRASP) for the general version of OP [21], [1]. This led us to use this method in combination with the previously developed genetic algorithm for solving the OPTW [10].

The remainder of the paper is organised as follows. The mathematical formulation of the problem is presented in Section 2. An overview of the main approaches in the literature is presented in Section 3. In section 4, we describe the concept of the hybrid genetic algorithm with path relinking. The results of computational experiments illustrating the effectiveness of our approach in comparison with other methods are discussed in Section 5. Concluding remarks and plans for further research are given in Section 6.

2 Mathematical formulation

Based on the notation introduced in the previous section the OPTW can be formulated as an mixed integer problem as follows [26]:

$$max \sum_{i=1}^{n-1} \sum_{j=2}^{n} p_i x_{ij} \tag{1}$$

$$\sum_{j=2}^{n} x_{1j} = \sum_{i=1}^{n-1} x_{in} = 1 \tag{2}$$

$$\sum_{i=1}^{n-1} x_{ik} = \sum_{j=2}^{n} x_{kj} \leqslant 1 \quad \forall k \in \{2, ..., n-1\} \tag{3}$$

$$\sum_{i=1}^{n-1}\sum_{j=2}^{n} t_{ij}x_{ij} \leqslant t_{max} \tag{4}$$

$$start_i + T_i + t_{ij} - start_j \leqslant M \cdot (1 - x_{ij}) \tag{5}$$

$$O_i \leqslant start_i \leqslant C_i \quad \forall i = 1, ..., n \tag{6}$$

where x_{ij} are binary variables, such that $x_{ij} = 1$ if the edge between i and j is included in a solution, and $x_{ij} = 0$ otherwise. Moreover, we assume that s=1 and $e = n$. Let $start_i$ denote the start of service time at vertex i and M be a large constant. The objective function (1) maximises the total collected profit of the route. The constraint in (2) guarantees that the path starts at vertex 1 and ends at vertex n. Constraint (3) requires that there may be at most one visit to any vertex. The constraint in (4) ensures that the time of the route is limited by t_{max}. Constraint (5) ensures the timeline of the route. The start of the service is restricted by a time window as in (6).

3 Literature review

It can be easily observed that the OPTW is a special case of the TOPTW: in the OPTW one route is constructed, while in TOPTW several routes are generated. Methods for the TOPTW could also be applied to the OPTW. Therefore, in this section we present solution approaches described in the literature for the OPTW as well as the TOPTW.

The Orienteering Problem with Time Windows has been studied since Kantor and Rosenweins article [7]. Their insertion heuristic constructs a route by iteratively inserting the vertex with the highest ratio $score/TimeInsertion$ without violating time windows and t_{max} constraints. In the second method proposed, they developed what is known as a tree heuristic, in which a depth-search algorithm constructs routes that begin in a given vertex. If a route is infeasible or unlikely to yield a better result, the route is abandoned. In this case the algorithm backtracks to the previous level of the tree and attempts to insert another vertex.

Righini et al. [18] developed an exact optimisation algorithm for the OPTW based on bi-directional dynamic programming. This technique requires extension of non-dominated states from both sides of the route: forward from the start vertex and backward from the end vertex. The decremental state relaxation method was also introduced for this algorithm with the idea of iteratively reducing the number of explored states [19].

Mansini et al. [17] introduced the Granular Variable Neighbour Search approach for TOPTW based on the idea of exploring a reduced neighbourhood instead of a complete one and not including arcs that are not promising. The

method improved algorithm efficiency with no loss of effectiveness. A more general concept of granularity was described in [11]. The cost of an arc was identified by the formula $(t_{ij} + w_{ij})/(p_i + p_j)$, where w_{ij} is the maximum possible waiting time at j provided that the service at i is assumed to start at the fixed time. Promising arcs were identified as follows: the lower the reduced cost associated with the arc, the higher the probability it will belong to a good solution.

Montemanni et al. [15] proposed a solution model using hierarchical generalization of TOPTW based on an Ant Colony System (ACS) algorithm. Two improvements to the ASC method were included in the ACS [16]: the constructive phase was sped up by considering the best solution computed so far, and the local search procedure was applied only to those solutions to which it had not been applied in the previous iteration (the same route was not optimised too often).

Tricoire et al. [22] adapted a solution to the Multi-Period Orienteering Problem with Multiple Time Windows for the TOPTW. They proposed an exact algorithm for the path feasibility sub-problems, and embedded it in a variable neighbourhood search (VNS) approach to solve the whole problem.

Vansteenwegen et al. proposed the Iterated Local Search approach (ILS) [25] to tackle the TOPTW. Because it is the fastest known heuristic, ILS is applied, for example, in electronic devices such as mobile tourist guides [4], [24]. The ILS method iteratively builds and improves one route by combining an insertion step and deletion of some consecutive locations (a shake step) to escape from a local maximum.

Lambadie et al. [12] developed a method for solving TOPTW which combines a greedy randomised adaptive search procedure (GRASP) with an evolutionary local search (ELS). In the ELS phase deletion and insertion mutations are performed for multiple child solutions. Child solutions are further improved by a variable neighbourhood descent procedure. The GRASP ELS method gives the best results on benchmark instances in comparison with the other methods mentioned [11]

4 Genetic Algorithm

The proposed method, called GAPR, is an extended version of the genetic algorithm GA proposed in [10]. The individuals (routes) are encoded as a sequence of vertices (genes). The GAPR starts by generating an initial population of P_{size} routes. Next, each individual is evaluated by means of the fitness function F. We use F as in [10], [8], [9], which is equal to $TotalProfit^3/TravelTime$. $TotalProfit$ and $TravelTime$ denote the sum of the profits assigned to the vertices on the route and the total travel time from the starting point to the ending point. In subsequent iterations of the GAPR the population is evolved by applying genetic operators selection, recombination and mutation in order to create new, better routes. The optimisation strategy, in contrast with the method proposed in [10], involves a recombination stage (crossover) performed on two random routes: instead of randomly choosing a crossing point between

vertices with similar time windows and starting and ending time of service [10], we used a path relinking process. In this process routes in the graph solution space connecting two random solutions are explored in order to find better solutions [5], [6]. To generate new solutions between selected random routes, genes not present in one route are included in the other. The solution generated by the path relinking process corresponds to the best individuals that could be obtained by applying the crossover operator to the same random parents.

The GAPR terminates after a fixed number of generations (denoted by N_g), or earlier if it converges. The GAPR result is the route in the final population with the highest profit value. The basic structure of the GAPR is as follows:

```
compute initial population;
algLoop=0;
while algLoop< Ng do
   algLoop++;
   tournament grouping selection;
   path relinking;
   mutation;
   if no improvements in last 100 iterations then break;
end;
return the route with the highest profit value;
```

Due to randomization, the GAPR is run several times during the tests. Each successive repetition of the GAPR is independent of the others, so this is a prime target for parallelisation. OpenMP [13], which is an API, is used in the algorithm for parallel computations, which substantially reduce its execution time.
The application of genetic operators for selection, recombination and creation of new individuals is described in more detail below.

4.1 Initialisation

In the approach presented a route is coded as a sequence of vertices. A population of P_{size} routes is generated as follows. First the chromosome is initialized by the s and e vertices. Then the following values are assigned sequentially to the initialized vertices: $arrive_i$ - arrival time at vertex i, $wait_i$ - waiting time, if the arrival at the vertex i is before opening time, $start_i$ and end_i - starting and ending service time at vertex i. Moreover, the maximum time the service of a visit i can be delayed without making other visits infeasible is calculated for each location in the route as follows [25]:

$$MaxShift_i = Min(C_i - start_i - T_i, wait_{i+1} + MaxShift_{i+1}) \tag{7}$$

Let l be the predecessor of vertex e in the route. In the subsequent steps a set of vertices is prepared. Each vertex v from this set is adjacent to vertex l and vertex e and will satisfy the following conditions after insertion: (a) $start_v$ and end_v are within the range $[O_v, C_v]$; (b) the locations after v could be visited

in the route; and (c) the current travel length does not exceed the given t_{max} (including consumption time to insert the vertex v between l and e). A random vertex v is chosen from this set. The values $arrive_v$, $wait_v$, $start_v$ and end_v are calculated and the vertex v is inserted. After the insertion, the values $arrive_e$, $wait_e$, $start_e$ and end_e are updated. Moreover, for each vertex in the tour (from vertex e to s) the $MaxShift$ value is updated as well. The tour generation is continued for as long as locations that have not been included are present and t_{max} is not exceeded.

4.2 Selection

We use tournament grouping selection, which yields better adapted individuals than standard tournament selection [9]. In this method a set of P_{size} individuals is divided into k groups and the tournaments are carried out sequentially in each of the groups. t_{size} random individuals are removed from the group, the chromosome with the highest value for the fitness function $TotalProfit^3/TravelTime$ is copied to the next population, and the t_{size} previously chosen individuals are returned to the old group. After selection from the group currently analysed has been repeated P_{size}/k times, P_{size}/k individuals are chosen for a new population. Finally, when this step has been repeated in each of the remaining groups, a new population is created, containing P_{size} routes.

4.3 Path relinking

First two random routes R_1 and R_2 are selected from the new population chosen in the selection step. Let $V_{R_1-R_2}$ be the set of vertices present in R_1 and not in R_2 , and let $V_{R_2-R_1}$ denote the set of vertices present in R_2 and not in R_1. During $\mathrm{PR}(R_1, R_2)$ we attempt to insert vertices from $V_{R_2-R_1}$ into R_1 in the best possible position. The total consumption time associated with inserting a vertex j between vertex i and k is calculated as follows [25]: $Shift_j = t_{ij} + wait_j + T_j + t_{jk} - t_{ik}$. In addition, we check whether the shift resulting from the new insertion exceeds the constraints associated with the previously calculated $wait$ and $MaxShift$ values for the vertices located directly after the newly inserted one. If the shift exceeds the constraints the vertices from $V_{R_1-R_2}$ are removed to restore the possibility of inserting new locations. For each vertex u from this set a ratio is calculated as follows: $RemovalRatio = (p_u)^2/(end_u - arrive_u)$, with the power 2 having been determined experimentally. After this computation the vertex with the smallest value for $RemovalRatio$ is removed. This removal is repeated until we can insert some vertices into the path. Finally the vertex u with the highest value for $(p_u)^2/Shift(u)$ and not exceeded the mentioned constrains is selected for insertion. After u is inserted the values of $arrive_u$, $wait_u$, $start_u$ and end_u are calculated. For each location after u the arrival time, waiting time, and start and end of service are updated. $MaxShift$ values are also updated for the vertices from the starting point to the ending point of the route. As we can see, the insertion of one vertex from $V_{R_2-R_1}$ into R_1 is a multi-stage process. The process is repeated for as long as t_{max} is not exceeded and

the set $V_{R_2-R_1}$ is not empty. In addition, we perform PR(R_2, R_1) by inserting vertices from $V_{R_1-R_2}$ into R_2. Two new routes are created as a result of PR(R_1, R_2) and PR(R_2, R_1). If the fitness values of the new routes are higher than the fitness value of R_1 and R_2, they replace them.

4.4 Mutation

In this phase a random route is selected from P_{size} individuals. Two types of mutation are possible a gene insertion or gene removal (the probability of each is 0.5). The mutation process is repeated on the selected route N_m times, where N_m is the parameter. During the *insertion mutation*, all possibilities for inclusion of each new vertex (not present in the route) are considered in the same way as in the path relinking process. The locations before and after the inserted vertex should be updated as in the case of the insertion process in the path relinking. In the *deletion mutation* we remove a randomly selected gene (excluding the first and last genes) in order to shorten the travel length. After the gene is removed, all locations after the removed gene are shifted towards the beginning of the route. Furthermore, the locations before and after the removed gene are updated as in the insertion mutation.

5 Experimental Results

The GAPR was coded in C++ and run on an Intel Core i7, 1.73 GHz CPU (turbo boost to 2.93 GHz). The algorithm was tested on Solomon [20] and Cordeau [3] test instances for the OPTW. The number of vertices in the Solomon instances is equal to 100 and different layouts for the vertices are considered: cluster (c), random (r) and random-clustered (rc) classes. The Solomon benchmarks c200, r200, rc200 and c100, r100, rc100 have the same coordinates of vertices, profits and visiting times, but the $c \setminus r \setminus rc200$ instances have approximately three times higher values of t_{max} and larger time windows than the $c \setminus r \setminus rc100$ instances. The Cordeau instances vary between 48 and 288 vertices.

The parameters of the GAPR were determined by performing several tests on a selected subset of Solomon and Cordeau instances. Preliminary tests identified the following as good performing parameters: 150 for the initial population size, 3 for the number of individuals chosen from the group in the tournament selection, and 15 for the number of groups in the tournament selection. Based on the tests described in [10], the N_m number of mutations repeated on the selected route was set to 15.

Detailed results obtained by the GAPR on benchmark instances in comparison with other methods are given in Tables 2 - 4. There are two columns for the GAPR, denoted GAPR(I) and GAPR(II). The first reports the results obtained by considering only PR(R_1, R_2) in the path relinking (in each iteration of the algorithm). The second shows the results of the use of both PR(R_1, R_2) and PR(R_2, R_1) during the path relinking. For comparison of the results, the best known solution value (BK) (solutions obtained by GRASP ELS and ACS

algorithm [12], [16]), the GA (with crossover) [10] and ILS [25] are also reported. In the BK columns the optimal values when are known are marked in italic. The GAPR was tested by performing sixteen runs concurrently two runs each on eight processor cores. The results of the GA were obtained with sixteen runs of the algorithm (without concurrency) on the same computer used to run the GAPR. The total time of the sixteen runs (expressed in seconds) and the minimum, average, and maximum solutions are given in the tables for the GA and the GAPR. The ILS (deterministic algorithm) results were obtained with one run [25]. Tables 2 - 4 also show the average percentage gap between the best known solution (BK) values and the average value of the GAPR, and for comparison, the gaps between BK and the other methods mentioned. An empty cell denotes a gap equal to 0.

The results presented in Tables 2 - 4 indicate that the GAPR outperforms the ILS results on $c \setminus r \setminus rc100$, $c \setminus rc200$ and the Cordeau instances. Only in the case of $r200$ does the ILS perform slightly better than the GAPR (the ILS has a smaller gap than the GAPR, by about 0.4%). The average gap between the BK and the ILS results for all these instances is 3.6%, while the gaps between BK and GAPR(I) and GAPR(II) are 2.4% and 2.5%, respectively. Because the use of PR(R_1, R_2) and PR(R_2, R_1) results in faster convergence of the algorithm, in some cases (e.g. rc200) the creation of two new routes in each iteration of GAPR(II) yields worse results than calculation of only one route as in GAPR(I). For comparison, the average gap between BK and the previous genetic algorithm GA is 5.6%. The application of the path relinking stage in place of the crossover significantly improves the GAPR results by about 6% in comparison to the GA. As a result of the parallel computing, the GAPR is on average 22 times faster than the GA and its execution time is comparable with the ILS. Moreover, the GAPR provides several new best solutions on Cordeau instances p11, p15, p17 and p19, whose improved values are given in bold in Table 4. There are not an known optimal values for these instances [11]. Examples of the best-generated routes by GAPR for pr17 and pr15 are presented in Figure 2. Comparison the GAPR with the GA and the ILS results for these benchmarks is presented in Table 1.

The number of generations in the GAPR was experimentally set to 500 as the stopping criterion. As seen in Figure 1 in the case of the pr11-20 the best routes were generated earlier than 500 generations (the exception is the pr20, where the result was only better about 2% after 620 generations). Therefore, for the optimisation of the execution time, the algorithm was stopped earlier if were not any improvements in the lengths of the routes by 100 generations.

6 Conclusions and Further Work

This paper presents the application of the genetic algorithm hybridised with the path relinking method to the Orienteering Problem with Time Windows. Using path relinking instead of crossover improves the results on benchmark instances by about 6%. Moreover, the proposed GAPR algorithm outperforms

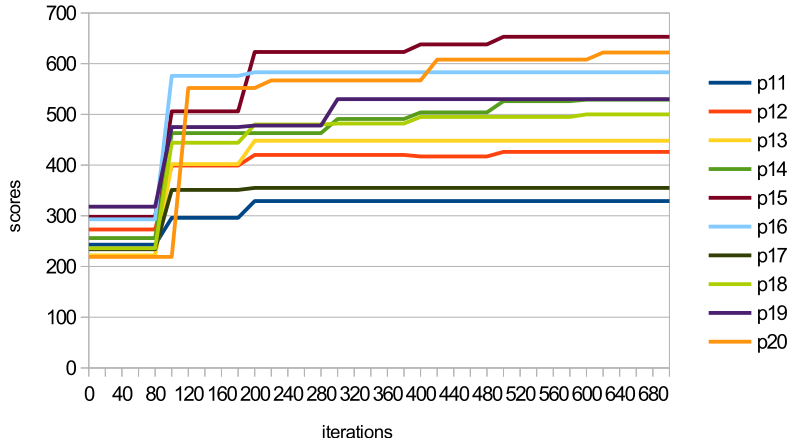

Fig. 1. Convergence of the GAPR for pr11-20 instances.

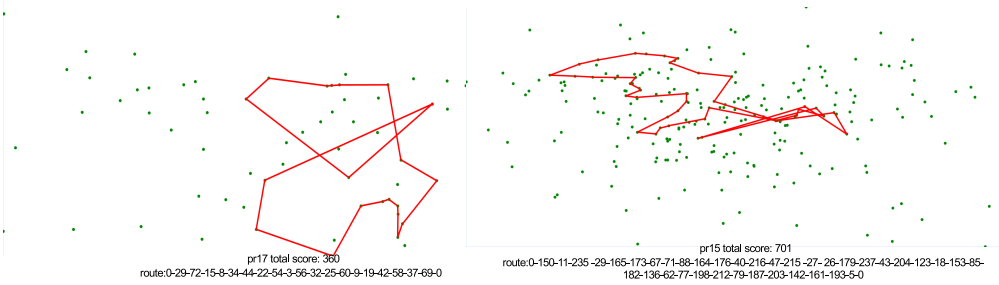

Fig. 2. Examples of the best-generated routes by GAPR for pr17 and pr15.

the results of the ILS heuristic, while the execution times of the two algorithms are comparable. The ILS is very fast and is applied, for example, in mobile tourist guide applications [25]. The proposed heuristic can also be adapted to solve problems related to planning tourist routes.

Further research directions include improving the GAPR results by applying path relinking operators between pairs of elite solutions (known as evolutionary path relinking) and conducting tests on a realistic database. Moreover, we intend to focus our research on developing effective heuristics for the Team Orienteering Problem with Time Windows, which is an extension of the OPTW.

Table 1. Comparison the GAPR with the GA and the ILS routes for pr15 and pr17.

	method	length of the route	profit	route
p15	ILS	69643	630	0-165-158-91-11-81-102-29-64- -62-136-77-198-182-85-153-18-238-123-204-43-237-179- -26-144-119-67-129-154-193-142-203-55-187-161-5-0
	GA	60801	653	0-150-11-29-235-165-173-67-88- -164-71-119-144-26-179-34-204-123-43-72-18-153-85- -182-136-62-77-198-212-79-203-142-187-161-193-0
	GAPR	68775	701	0-150-11-235-29-165-173-67-71-88-164-176- -40-216-47-215-27-26-179-237-43-204-123-18-153-85- -182-136-62-77-198-212-79-187-203-142-161-193-5-0
p17	ILS	70696	346	0-35-32-54-22-44-34-8-4- -21-48-66-30-60-9-19-42-58-61-37-69-0
	GA	70324	353	0-29-63-5-15-56-54-8-44- -34-22-32-6-9-19-42-58-37-69-0
	GAPR	70955	360	0-29-72-15-8-34-44-22-5-3- -56-32-25-60-9-19-42-58-37-69-0

Acknowledgements

The authors gratefully acknowledge support from the Polish Ministry of Science and Higher Education at the Bialystok University of Technology (grant S/WI/1/2011 and W/WI/2/2013).

References

[1] Campos, V., Marti, R.,Sanchez-Oro, J., Duarte, A.: Grasp with Path Relinking for the Orienteering Problem. Technical Raport, 116 (2012)
[2] http://www.citytripplanner.com/en/home . Last access: June 29, 2013
[3] Cordeau, J.F., Gendreau, M., Laporte, G.: A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks **30(2)**(1997) 105-119
[4] Garcia, A., Vansteenwegen, P., Arbelaitz, O., Souffriau, W., Linaz, M.: Integrating Public Transportation in Personalised Electronic Tourist Guides. Computers & Operations Research. **40** (2013) 758–774
[5] Glover, F.: A Template For Scatter Search And Path Relinking. Lecture Notes in Computer Science. **1363** (1997) 13-54
[6] Glover, F., Laguna, M.: Tabu Search. Kluwer Academic Publishers. Boston (1997)
[7] Kantor, M., Rosenwein, M .: The Orienteering Problem with Time Windows. Journal of the Operational Research Society. **43** (1992) 629–635
[8] Karbowska-Chilinska, J., Koszelew, J., Ostrowski, K., Zabielski, P.: Genetic algorithm solving orienteering problem in large networks. Frontiers in Artificial Intelligence and Applications. **243** (2012) 28–38

[9] Karbowska-Chilinska, J., Koszelew, J., Ostrowski, K., Zabielski, P.: A Genetic Algorithm with Grouping Selection and Searching Operators for the Orienteering Problem. (under review)
[10] Karbowska-Chilinska, J., Zabielski, P.:A Genetic Algorithm Solving Orienteering Problem with Time Windows. (accepted for publication in Springer series: Advances in Intelligent Systems and Computing)
[11] Labadie, N., Mansini, R., Melechovsky, J., Wolfler Calvo, R.: The Team Orienteering Problem with Time Windows: An LP-based Granular Variable Neighborhood Search. European Journal of Operational Research. **220(1)** (2012) 15-27
[12] Labadie, N., Melechovsk, J., Wolfler Calvo, R.: Hybridized evolutionary local search algorithm for the team orienteering problem with time windows. Journal of Heuristics. **17(6)** (2011) 729-753
[13] http://openmp.org/wp/ . Last access: June 29, 2013
[14] Ostrowski, K., Koszelew, J.: The comparison of genetic algorithm which solve Orienteering Problem using complete an incomplete graph. Zeszyty Naukowe, Politechnika Bialostocka. Informatyka **8** (2011), 61–77
[15] Montemanni, R., Gambardella, L.M.: Ant colony system for team orienteering problems with time windows. Foundations of Computing and Decision Sciences. **34** (2009)
[16] Montemanni, R., Weyland, D., Gambardella L. M.: An Enhanced Ant Colony System for the Team Orienteering Problem with Time Windows. Proceedings of IEEE ISCCS 2011 The 2011 International Symposium on Computer Science and Society, Kota Kinabalu, Malaysia 381-384 (2011)
[17] Mansini, R., Pelizzari, M. Wolfler, R.: A Granular Variable Neighborhood Search for the Tour Orienteering Problem with Time Windows, Technical Report of the Department of Electronics for Automation, University of Brescia (2006)
[18] Righini, G., Salani, M.: New dynamic programming algorithms for the resource constrained elementary shortest path. Networks. **51(3)** (2008) 155 –170
[19] Righini, G., Salani, M.: Decremental state space relaxation strategies and initialization heuristics for solving the Orienteering Problem with Time Windows with dynamic programming. Computers & Operations Research. **36** (2009) 1191 – 1203
[20] Solomon, M.: Algorithms for the Vehicle Routing and Scheduling Problems with Time Window Constraints. Operations Research **35(2)** (1987) 254-265
[21] Souffriau, W.,Vansteenwegen, P., Vanden Berghe, G., Van Oudheusden, D.: A Path Relinking approach for the Team Orienteering Problem, Computers & Operations Research, **37(11)** (2010) 1853-1859
[22] Tricoire, F., Romauch, M., Doerner, K. F., Hartl, R. F. Heuristics for the multi-period orienteering problem with multiple time windows. Comput. Oper. Res. **34(2)** (2010) 351–367
[23] Tsiligirides, T.: Heuristic methods applied to orienteering. Journal of the Operational Research Society. **35(9)** (1984) 797–809
[24] Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D.: The City Trip Planner: An expert system for tourists. Expert Systems with Applications. **38(6)** (2011) 6540–6546
[25] Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D.: Iterated local search for the team orienteering problem with time windows. Computers O.R. **36** (2009) 3281–3290
[26] Vansteenwegen, P., Souffriau, W., Van Oudheusden, D.. The Orienteering Problem: A survey. European Journal of Operational Research. **209(1)** (2011) 1–10.

Table 2. Results for Solomon's test problems (n=100).

	BK	ILS		GA				GAPR(I)				GAPR(II)					% gap BK with			
name	score	score	time	min	avg.	max	time	min	avg.	max	time	min	avg.	max	time		ILS	GA	GAPR(I)	GAPR(II)
c101	*320*	320	0.4	310	317	320	4.2	320	320	320	0.2	320	320	320	0.2		1.0			
c102	*360*	360	0.3	360	360	360	4.4	360	360	360	0.2	360	360	360	0.3					
c103	*400*	390	0.5	380	389	400	6.3	400	400	400	0.3	400	400	400	0.3		2.5	2.8		
c104	*420*	400	0.3	400	403	420	6.3	420	420	420	0.3	400	409	410	0.4		4.8	4.2		2.6
c105	*340*	340	0.3	340	340	340	4.3	340	340	340	0.2	340	340	340	0.2					
c106	*340*	340	0.3	340	340	340	4.0	340	340	340	0.2	340	340	340	0.2					
c107	*370*	360	0.3	360	362	370	4.5	360	360	360	0.2	360	360	360	0.2		2.7	2.2	2.7	2.7
c108	*370*	370	0.3	370	370	370	5.1	370	370	370	0.3	370	370	370	0.2					
c109	*380*	380	0.3	380	380	380	4.9	380	380	380	0.3	380	380	380	0.2					
sum:	**3300**	**3260**	**3.0**	**3240**	**3260**	**3300**	**44**	**3290**	**3290**	**3290**	**2.2**	**3270**	**3279**	**3280**	**2.2**	**avg.:**	**1.2**	**1.2**	**0.3**	**0.6**
r101	*198*	182	0.1	182	189	198	3.2	197	197	197	0.2	198	198	198	0.2		8.1	4.8	0.5	
r102	*286*	286	0.2	281	286	286	5.4	286	286	286	0.2	286	286	286	0.2			0.1		
r103	*293*	286	0.2	286	290	293	5.3	293	293	293	0.3	293	293	293	0.3		2.4	1.1		
r104	*303*	297	0.2	297	297	298	4.9	303	303	303	0.3	297	298	298	0.3		2.0	2.0		1.7
r105	*247*	247	0.1	240	244	247	3.9	247	247	247	0.2	247	247	247	0.2			1.2		
r106	*293*	293	0.2	281	292	293	4.7	293	293	293	0.2	293	293	293	0.2			0.3		
r107	*299*	288	0.2	286	292	297	6.1	299	299	299	0.3	299	299	299	0.3		3.7	2.3		
r108	*308*	297	0.2	297	300	308	5.5	308	308	308	0.4	303	308	308	0.4		3.6	2.5		
r109	*277*	276	0.2	258	270	274	4.6	270	272	274	0.2	270	273	274	0.3		0.4	2.5	1.7	1.4
r110	*284*	281	0.3	274	277	281	4.7	283	283	283	0.3	281	281	281	0.2		1.1	2.4	0.4	1.1
r111	*297*	295	0.2	275	293	295	5.8	297	297	297	0.3	297	297	297	0.3		0.7	1.3		
r112	*298*	295	0.2	290	293	295	6.2	292	292	292	0.2	295	295	295	0.3		1.0	1.6	2.0	1.0
sum.	**3383**	**3323**	**2.3**	**3247**	**3323**	**3365**	**60.4**	**3368**	**3370**	**3372**	**3.0**	**3359**	**3368**	**3369**	**3.2**	**avg.:**	**1.8**	**1.8**	**0.4**	**0.4**
rc101	*219*	219	0.2	210	216	219	3.5	216	216	216	0.2	216	216	216	0.2			1.4	1.4	1.4
rc102	*266*	259	0.2	255	261	266	4.8	266	266	266	0.3	266	266	266	0.2		2.6	1.9		
rc103	*266*	265	0.3	253	262	266	5.3	259	265	266	0.3	265	265	266	0.3		0.4	1.5	0.5	0.4
rc104	*301*	297	0.3	276	294	301	6.1	301	301	301	0.2	297	301	301	0.3		1.3	2.3		
rc105	*244*	221	0.2	230	236	241	4.5	241	241	241	0.2	241	241	241	0.2		9.4	3.3	1.2	1.2
rc106	*252*	239	0.2	233	245	250	4.8	128	242	250	0.2	249	250	250	0.3		5.2	2.8	3.8	0.8
rc107	*277*	274	0.2	264	269	274	5.0	122	258	277	0.2	271	273	277	0.3		1.1	2.9	6.9	1.4
rc108	*298*	288	0.2	278	289	298	4.9	298	298	298	0.2	288	297	298	0.3		3.4	3.0		0.3
sum.	**2123**	**2062**	**1.8**	**1809**	**2072**	**2115**	**39.1**	**1831**	**2087**	**2115**	**1.8**	**2093**	**2109**	**2115**	**2.1**	**avg.:**	**2.9**	**2.4**	**1.7**	**0.7**

Table 3. Results for Solomon's test problems, cont. (n=100).

	BK	ILS		GA				GAPR(I)				GAPR(II)					% gap BK with			
name	score	score	time	min	avg.	max	time	min	avg.	max	time	min	avg.	max	time		ILS	GA	GAPR(I)	GAPR(II)
c201	870	840	1.1	840	848	860	13.2	860	860	860	0.4	860	860	860	0.6		3.4	2.5	1.1	1.1
c202	930	910	2.8	890	901	920	15.9	920	920	920	0.5	920	924	930	0.7		2.2	3.1	1.1	0.6
c203	960	940	1.7	910	931	940	17.8	950	957	960	0.7	960	960	960	0.7		2.1	3.0	0.3	
c204	980	950	1.6	940	952	970	24.2	960	968	970	0.8	970	970	970	0.8		3.1	2.9	1.3	1.0
c205	910	900	1.2	880	893	900	16	900	900	900	0.4	900	900	900	0.5		1.1	1.9	1.1	1.1
c206	930	910	1.6	890	905	910	15.6	900	919	920	0.5	850	906	910	0.5		2.2	2.7	1.2	2.6
c207	930	910	2.1	890	910	930	16.1	930	930	930	0.7	920	920	920	0.5		2.2	2.2		1.1
c208	950	930	1.6	920	931	940	15.3	940	940	940	0.5	950	950	950	0.5		2.1	2.0	1.1	
sum:	**7460**	**7290**	**13.7**	**7160**	**7271**	**7370**	**134.2**	**7360**	**7394**	**7400**	**4.5**	**7330**	**7390**	**7400**	**4.8**	**avg.:**	**2.3**	**2.5**	**0.9**	**0.9**
r201	797	788	1.2	733	760	782	18.9	772	776	778	0.6	787	789	789	0.8		1.1	4.6	2.6	1.0
r202	929	880	1.4	834	867	892	28.3	883	885	891	1.1	888	890	892	0.8		5.3	6.7	4.8	4.2
r203	1021	980	1.6	925	954	980	35.1	965	969	974	1.2	961	968	979	1.2		4.0	6.6	5.1	5.2
r204	1086	1073	1.7	968	1018	1051	40.4	1031	1031	1031	1.0	1042	1042	1042	0.8		1.2	6.4	5.1	4.1
r205	953	931	1.4	851	876	925	25.7	900	925	933	1.3	894	895	897	0.9		2.3	8.1	3.0	6.1
r206	1029	996	1.5	930	954	987	32.3	996	1007	1011	1.1	999	1005	1009	1.2		3.2	7.4	2.1	2.3
r207	1072	1038	2.0	939	986	1022	33.1	1023	1041	1051	1.3	1023	1031	1035	1.5		3.2	8.0	2.9	3.9
r208	1112	1069	1.6	1002	1042	1086	38.7	1057	1074	1075	1.4	1080	1098	1099	1.4		3.9	6.3	3.4	1.3
r209	950	926	2.4	866	897	927	27.1	903	927	943	1.0	920	923	930	1.2		2.5	5.6	2.4	2.8
r210	987	958	1.9	870	915	944	28.2	933	944	946	1.0	960	960	960	0.8		2.9	7.3	4.4	2.7
r211	1046	1023	1.6	934	967	1007	37.6	1002	1003	1003	0.8	1019	1019	1019	1.1		2.2	7.5	4.2	2.6
sum:	**10982**	**10662**	**18.3**	**9852**	**10235**	**10603**	**345**.8	**10465**	**10581**	**10636**	**11.9**	**10573**	**10619**	**10651**	**11.7**	**avg.:**	**2.9**	**6.8**	**3.7**	**3.3**
rc201	795	780	1.0	670	768	794	21.3	787	789	790	0.6	768	770	771	0.6		1.9	3.4	0.7	3.1
rc202	936	882	1.3	822	866	932	20.8	919	919	919	0.7	887	900	905	0.8		5.8	7.5	1.8	3.9
rc203	1003	960	2.7	822	866	932	20.8	953	956	956	1.0	948	963	973	1.2		4.3	7.1	4.7	13.9
rc204	1136	1117	2.3	978	1044	1107	32.2	1123	1123	1123	0.9	1077	1079	1079	1.3		1.7	8.1	1.1	5.1
rc205	859	840	1.0	751	808	837	20.8	842	842	842	0.7	838	842	849	0.9		2.2	5.9	2.0	1.9
rc206	895	860	1.1	819	850	865	20.8	857	859	860	0.7	856	856	856	0.8		3.9	5.0	4.0	4.4
rc207	983	926	1.3	859	896	928	26.7	946	948	950	0.9	916	930	940	1.0		5.8	8.9	3.6	5.4
rc208	1053	1037	2.3	937	976	1032	23.1	1005	1005	1005	0.6	1034	1034	1034	0.8		1.5	7.3	4.6	1.8
sum:	**7660**	**7402**	**13**	**6658**	**7140**	**7427**	**191.5**	**7432**	**7442**	**7445**	**6.2**	**7324**	**7374**	**7407**	**7.4**	**avg.:**	**3.4**	**6.8**	**2.9**	**5.1**

Table 4. Results for Cordeau's test problems (n from 48 to 288).

	BK	ILS		GA				GAPR(I)				GAPR(II)					% gap BK with			
n name	score	score	time	min	avg.	max	time	min	avg.	max	time	min	avg.	max	time		ILS	GA	GAPR(I)	GAPR(II)
48 pr1	*308*	304	0.5	275	298	308	6.4	305	305	305	0.2	305	306	308	0.3		1.3	3.2	1.0	0.6
96 pr2	*404*	385	0.6	370	381	393	10.1	398	400	401	0.4	382	386	386	0.4		4.7	5.7	1.1	4.5
144 pr3	*394*	384	1.0	344	373	392	12.8	393	393	393	0.4	393	393	393	0.6		2.5	5.3	0.3	0.3
192 pr4	*489*	447	1.9	413	449	471	21.1	487	487	487	0.6	470	470	470	0.6		8.6	8.2	0.4	3.9
240 pr5	*595*	576	4.6	521	553	585	33.4	549	559	577	2.0	567	578	587	1.6		3.3	7.1	6.1	2.9
288 pr6	*590*	538	2.5	471	512	568	27.8	567	573	579	1.6	580	582	583	1.6		8.8	13.2	2.9	1.4
72 pr7	*298*	291	0.4	270	282	291	6.0	288	288	288	0.2	289	289	291	0.3		2.3	5.4	3.4	3.0
144 pr8	*463*	463	1.0	410	426	447	14.1	463	463	463	0.4	463	463	463	0.6			8.0		
216 pr9	*493*	461	1.4	422	448	470	24.9	450	451	459	0.9	455	456	458	1.0		6.5	9.1	8.5	7.5
288 pr10	594	539	3.6	502	536	568	33.5	536	556	570	1.8	539	540	541	1.7		9.5	9.8	6.3	9.1
sum:	**4628**	**4388**	**17.5**	**3998**	**4258**	**4491**	**190.2**	**4436**	**4475**	**4522**	**8.6**	**4443**	**4463**	**4483**	**8.7**	**avg.:**	**5.2**	**8.0**	**3.3**	**3.6**
48 pr11	330	330	0.3	310	332	342	7.0	342	344	**345**	0.3	**345**	**345**	**345**	0.2			-0.6	-4.3	-4.5
96 pr12	442	431	0.9	401	417	433	10.7	424	432	434	0.4	437	437	437	0.5		2.5	5.7	2.3	1.1
144 pr13	461	450	1.9	410	429	446	18.8	431	455	457	0.9	451	458	469	1.2		2.4	6.9	1.3	0.6
192 pr14	567	482	1.1	464	489	524	23.1	522	533	535	1.5	510	518	526	1.1		15.0	13.8	5.9	8.6
240 pr15	685	638	5.3	558	611	676	39.6	687	700	**701**	2.4	649	656	663	1.8		6.9	10.8	-2.1	4.2
288 pr16	674	559	4.1	525	564	611	35.6	600	607	609	1.5	605	626	635	2.5		17.1	16.3	10.0	7.1
72 pr17	359	346	0.2	332	349	356	8.4	350	359	**360**	0.3	353	353	353	0.3		3.6	2.8	-0.4	1.7
144 pr18	535	479	0.8	432	468	508	16.3	528	528	528	0.7	536	**536**	536	0.6		10.5	12.5	1.3	-0.2
216 pr19	562	499	2.7	450	501	530	29.1	532	533	535	1.3	531	534	546	1.3		11.2	10.9	5.1	4.9
288 pr20	667	570	2.5	569	590	613	29.1	601	611	612	2.1	623	627	629	2.2		14.5	11.5	8.3	6.0
sum:	**5282**	**4784**	**19.8**	**4451**	**4750**	**5039**	**232.9**	**5017**	**5102**	**5116**	**11.5**	**5040**	**5090**	**5139**	**12.0**	**avg.:**	**9.4**	**10.1**	**3.4**	**3.6**