A Memetic-GRASP Algorithm for the Solution of the Orienteering Problem

Yannis Marinakis, Michael Politis, Magdalene Marinaki, and Nikolaos Matsatsinis

School of Production Engineering and Management, Technical University of Crete, Chania, Greece
{marinakis,nikos}@ergasya.tuc.gr, magda@dssl.tuc.gr

Abstract. The last decade a large number of applications in logistics, tourism and other fields have been studied and modeled as Orienteering Problems (OPs). In the orienteering problem, a standard amount of nodes are given, each with a specific score. The goal is to determine a path, limited in length, from the start point to the end point through a subset of locations in order to maximize the total path score. In this paper, we present a new hybrid evolutionary algorithm for the solution of the Orienteering Problem. The algorithm combines a Greedy Randomized Adaptive Search Procedure (GRASP), an Evolutionary Algorithm and two local search procedures. The algorithm was tested in a number of benchmark instances from the literature and in most of them the best known solutions were found.

Keywords: Orienteering Problem, Memetic Algorithm, Greedy Randomized Adaptive Search Procedure, Local Search.

1 Introduction

In this paper, we present an algorithm for the solution of the Orienteering Problem. The Orienteering Problem was introduced by Golden et al. [5] when they described a game played in the mountains. The idea behind this game is that we have a number of players that they start from a specified control point, they have a map that informs them about a number of checkpoints and the scores associated with each one of them and they try to pass from as many checkpoints as possible adding to their total score the score of the specific point and to return to the control point (or to go to a different control point) within a specific time period [2]. The winner is the one that maximizes its total collected score [16]. This problem is one of the first problems that belongs in a category that is called Vehicle Routing Problems with Profit, where in each node (customer) except of the demand of the point, a profit is associated and, thus, the main goal is instead of (or in addition to) the minimization of the total distance or the total travel time, the maximization of the additive profit of visiting of the most profitable customers. In all these problems, it is possible not to visit all the customers but there is a selection of the customers that we have to visit as

H.A. Le Thi et al. (eds.), *Model. Comput. & Optim. in Inf. Syst. & Manage. Sci.*,
Advances in Intelligent Systems and Computing 360, DOI: 10.1007/978-3-319-18167-7_10

there is usually a time limit in order to perform the service of the customers. The most known variant of the Orienteering Problem is the Team Orienteering Problem where instead of one path, a number of P paths should be determined where the total collected score should be maximized [3,12]. Other variants of the Orienteering Problem are the Orienteering and the Team Orienteering Problems with Time Windows [7,9] where, also, time windows are assigned in each arc of the graph.

The Orienteering Problem has many applications in real life problems. One of them is its use for creating a Tourist Guide for a museum or for a specific city [11,14]. When a tourist visits a museum or a city it is often impossible to visit everything, thus, he should select the most interesting exhibitions or landmarks, respectively. If the tourist has plenty of time to visit everything inside the museum, then, the problem that he has to solve is only the finding of the sequence of visiting of the paintings or the sculptures. However, if the tourist has limited time, then, he has to make a selection of the most important parts of the museum for him. Thus, an ideal way to visit as more of his interesting exhibition parts in the museum as possible is to add, initially, a score to everything that exists in the museum (based on his personal preferences) and, then, to try to maximize this score by visiting the exhibition parts that have the highest scores. Thus, it has to solve a classic Orienteering Problem. One problem that may arise from the formulation of a routing problem in a museum as an orienteering problem is that the visiting time in each exhibition part cannot be calculated exactly as either the tourist would like to stay more or there is a lot of people around it and, thus, another constraint have to be added to the Orienteering Problem. There is a number of ways to solve this problem. One way is to add a specific time to the traveling time. However, this is not the most representative way as this time is not fixed for every visitor. Another way is to add a stochastic variant in each node of the graph which will correspond to the time that a visitor will stay to this exhibition part of the museum. Finally, a third way is the way that the authors of [17] formulate their problem as an Open Shop Scheduling problem in which they divide the visitors in different groups where they all are together and spend the same time in the exhibition rooms. The visitors are the jobs while the exhibition rooms are the machines. The time each visitor group spends in an exhibition room is considered analogous to the processing time required for each job on a particular machine [17].

In this paper, we solve the Orienteering Problem (OP) as a first step for creating a Tourist Guide either for a museum or for a specific city. The city that the algorithm will be applied is the city of Chania, Crete in Greece which is a very popular tourist destination. The idea behind this paper is to develop an algorithm that it could solve satisfactory the Orienteering Problem, with very good results in classic benchmark instances and in short computational time. In this algorithm, the waiting time of the tourists was added in the traveling time of an arc and it was thought constant for every tourist. The reason that we developed the algorithm with this assumption is that as we do not have benchmark instances for the museum (or tourist) routing problem that we would like

to formulate and solve, we begin with the application of the algorithm in a well known problem with well known benchmark instances and, then, we are going to proceed in the more demanding problem. Thus, we propose an algorithm that is an efficient hybridization of three algorithms, a Greedy Randomized Adaptive Search Procedure (GRASP) [4] for the creation of part of the initial solutions, an Evolutionary Algorithm as the main part of the algorithm and two local search phases (2-opt and 1-1 exchange) in order to improve each of the individuals of the population separately and, thus, to increase the exploitation abilities of the algorithm. The algorithm is denoted as Memetic-GRASP algorithm (MemGRASP). The algorithm is denoted as memetic algorithm [8] as it is an evolutionary algorithm with a local search phase. The reason that we use an evolutionary algorithm is that as the final application of the algorithm will be in the design of an tourist guide planner where the tourist will have the best option (meaning the visiting sequence of as many as possible points of interest) based on his preferences, the use of an evolutionary algorithm give us the possibility to give to the tourist alternative, very effective, options if for some reason his preferences change during his exhibition. Thus, the ability of a memetic algorithm to increase the exploration abilities of the procedure by searching in different places of the solution space give us the possibility of having good solutions in different solution space. The rest of the paper is organized as follows: In the next section a formulation of the Orienteering Problem is presented while in Section 3 the proposed algorithm is analyzed in detailed. In Section 4 the results of the proposed algorithm in the classic benchmark instances from the literature for the Orienteering Problem are given and in the last section the conclusions and the future research are presented.

2 Orienteering Problem

The Orienteering Problem (OP) can be described using a graph $G = (V, A)$, where $V, i = 1, \cdots, N$ denotes the set of nodes, each one having a score r_i and A is the set of arcs between points in V. There are two fixed points, the starting point (usually node 1) and the ending point (usually node N), where these two nodes could be the same or not and where these two nodes have zero score. There is a symmetric nonnegative cost c_{ij} associated with each arc, where c_{ij} denotes the time between point i and point j. Each node can be visited at most once and the total time taken to visit all points cannot exceed the specified limit T_{max} [16]. The main target of the solution of the OP is to determine a path, limited by T_{max}, that visits some of the nodes in order to maximise the total collected score. The scores are assumed to be entirely additive and each node can be visited at most once [16].

Making use of the notation introduced above, the OP can be formulated as an integer problem. The following decision variables are used: x_{ij} equal to 1 if a visit to node i is followed by a visit to node j, otherwise the value of x_{ij} is zero; y_i equal to 1 if node i is in the path or zero otherwise.

$$z = \max \sum_{j \in V} r_j y_j \tag{1}$$

$$s.t.$$

$$\sum_{j \in V} x_{1j} = 1, \tag{2}$$

$$\sum_{j \in V} x_{in} = 1, \tag{3}$$

$$\sum_{(i,j) \in A} x_{ij} + \sum_{(j,k) \in A} x_{jk} = 2y_j \ \forall j \in V, \tag{4}$$

$$\sum_{(i,j) \in A} c_{ij} x_{ij} \leq T_{\max} \tag{5}$$

$$\sum_{(i,j) \in S} x_{ij} \leq |S| - 1 \ \forall S \subseteq V \tag{6}$$

3 Memetic-Greedy Randomized Adaptive Search Procedure (MemGRASP) for the Orienteering Problem

In this section, the proposed algorithm is described in detail. Initially, in the MemGRASP algorithm we have to create the initial population. The first member of the population is produced by using a Greedy Algorithm. The path begins from node 1, which represents the starting point of each solution. Then, two different conditions are taken into account, the most profitable node and the nearest node to the last node inserted in the path. As there is a possibility the most profitable node not to be the nearest node to the last node inserted in the path and vice versa, we have to select which one of these two conditions is the most important. The most important condition is considered the condition of the profitable node as this is used for the calculation of the fitness function of the problem. However, as it is very important not to select a distant node, we have to add in the path a node as near as possible to the last added node in the path. In the case where more than one possible nodes have the same score (profit), which is a very possible situation in the instances the algorithm was tested, then, the nearest node is selected to be inserted in the path. Thus, the Greedy algorithm uses a hierarchical procedure where the most important condition is the profit of the node and the second most important is the distance.

At the end of the Greedy Algorithm, when no other node can be inserted in the path due to the fact that the constraint that restricts the length of the path has been violated, a complete solution is produced. We have to mention that in the formulation of the problem, described in Section 2, we mentioned that in the problem each tourist has a time limit and not a distance limit. However, in general we can consider that these two constraints could have the same role in the problem, meaning that if we have data that correspond to distance from one node to another and a limit in the length of the path, the constraint (5) could

have the same role in the OP as if we had the time traveled from one node to another and a maximum time limit for the path. The solution is represented with a path representation only of the nodes that have been selected to construct the path and the 2 nodes corresponding to the starting node and the ending node (for example, the entrance and the exit of the museum (these two nodes could be the same node)). Thus, the vector corresponding to each member of the population could have different length as there is a possibility in two solutions different number of nodes to be selected. Then, for the solution the fitness function is calculated which is the summation of the profit (score) of each of the node added in the path.

Next the initial path (solution), as every other path (solution) of the population, is tried to be improved using a combination of two local search algorithms, one is a type of exchange algorithm inside the path and the other is an insertion algorithm of a node that is not in the path between two nodes that are already in the path. The first one is used in such a way that the nodes of the path do not change but only the sequence of the nodes in the selected path may change. The reason that this procedure is used is that by finding a new sequence with shorter length (or less time) it is possible to add a new node in the path without violating the constraint (5) and, thus, to increase the summation of the score (profit) and to improve the solution. The second local search procedure takes the path produced by the first local search and tries to improve the fitness function of the solution by adding new nodes in the path. This procedure is applied even if the previous local search does not improve the initial path. In this procedure, an arc is removed and between the two nodes a new node is inserted (without violation of the constraint (5)) in order to improve the solution. With this procedure we ensure that if one successful move is realized, then, the fitness function of the solution will be improved, as the only condition in order to improve a solution is to add a new node in the path without deleting some of the existing nodes. If an improvement in the solution is achieved, then, the first local search algorithm is applied again in order to see if there is a possibility of more improvement in the solution. This local search algorithm is applied in every solution during the whole iterations of the algorithm.

The next step is the calculation of the solution of the rest members of the population. From member 2 to $NP/2$, where NP is the population number, the members of the population are calculated with a random procedure in order to spread the solutions in the whole space. In the random procedure, the solution begins from the starting node and, then, a node is selected at random without violating the limit (either time or distance) constraint and is added in the path. When the constraint is violated, the path is ended in the exit node. Then, the solution is tried to be improved using the local search procedure described previously. The last members of the population are calculated using Greedy Randomized Adaptive Search Procedure (GRASP). In the GRASP algorithm, a solution is created step by step where the best node is not added in the path but a list, the Restricted Candidate List - RCL with the most possible for inclusion nodes in the path, is created and one of them is selected randomly to

be included in the path. In the proposed algorithm, the RCL contains the most profitable nodes (based on their score (profit)) taking into account not to violate the limit (either time or distance) constraint and, then, in each step of the algorithm one of them is selected at random and is added in the path. Then, the RCL is updated with one node not in the path in order to keep its size (number of candidate nodes) constant. Finally, the local search described previously is applied.

All these solutions construct the initial population of the algorithm. Then, with the roulette wheel selection procedure the two parents are selected. A very interesting part of the algorithm is the crossover operator that is used in the next step. A 1-point crossover is used but it was very difficult to use it directly to the solutions as each solution contains the sequence of the nodes that a tourist will visit and, thus, two different solutions are possible to have different number of nodes. In order to solve this problem, each solution is mapped in a new vector with zeros and ones where a value equal to zero means that the node is not visited and a value equal to one means the opposite. Thus, the new vectors of two parents have the same size and the crossover operator can easily be applied and the two offspring are produced. In these vectors (either the parent vector or the offspring vector) the sequence of the nodes is not appeared. In the offspring in order to calculate the sequence of the nodes we produce a new vector using the nodes that have value equal to one and we apply the greedy algorithm described previously until the solution violates the constraint (5). Next the mutation operator is applied. The role of the mutation operator in this algorithm is either to improve a feasible solution or to transform an infeasible solution to feasible or, finally, to reject an infeasible solution that could not be transformed to a feasible one. If the limit constraint is violated, then, initially we apply the local search algorithm in order to find a solution which contains all the nodes without violating the limit constraint and if we could not find such a solution, then, we remove the less profitable nodes until the solution becomes feasible. On the other hand if all nodes of the offspring construct an feasible solution, then, all of them are selected. In order to improve the offspring, the local search described previously is applied.

Finally, the new population for the next generation is constructed. The new population contains the best solutions of the parents and of the offspring based on their fitness function taking into account not to have two solutions with the same path in order to avoid a fast convergence of the algorithm and the size of the population to remain constant in all generations. Then, the procedure continues with the selection of the new parents. All the steps of the algorithm (besides the creation of the initial population) are repeated until a maximum number of generations have been reached.

4 Computational Results

In this section, the computational results of the algorithm are presented and discussed in detail. As it was mentioned previously, we formulate the tourist

(or museum) routing problem discussed in this paper as an Orienteering Problem with the waiting time in each node to be equal for all tourists and to be added in the traveling time between two nodes. Thus, as we would like to test the effectiveness of the algorithm, we have a number of sets of benchmark instances in the literature to be used for the comparisons. Thus, in the webpage http://www.mech.kuleuven.be/en/cib/op/ there is a number of benchmark instances for different variants of Orienteering Problem. We select five sets of benchmark instances, three of them proposed in [13] and two of them proposed in [1]. The three sets of Tsiligirides [13] have 18, 11 and 20 instances, respectively, with number of nodes 32, 21 and 33, respectively, each one having different value in T_{max} in the limit constraint of the problem. For example, for the first set of instances of Tsiligirides the value of T_{max} varies between 5 to 85. The increase of the value allows more nodes to be visited in the best route. For each set of benchmark instances the value of T_{max} is presented in the corresponding Table. Finally, the last two sets of benchmark instances have 26 and 14 instances with 66 and 64 nodes, respectively.

Table 1. Computational results for Tsilligirides's Set 1

	T_{max}	MemGRASP	GLS	OPT	D(R - I)	S(R - I)	Knapsack	MVP
1	5	10	10	10			10	10
2	10	15	15	15			15	15
3	15	45	45	45			45	45
4	20	65	55	65	65	65	65	65
5	25	90	90	90	90	90	90	90
6	30	110	80	110	110	110	110	110
7	35	135	135	135	135	135	125	130
8	40	155	145	155	150	150	140	155
9	46	175	175	175	175	175	165	175
10	50	190	180	190	190	190	180	185
11	55	205	200	205	200	205	200	200
12	60	225	220	225	220	220	205	225
13	65	240	240	240	240	240	220	240
14	70	260	260	260	260	245	245	260
15	73	265	265	265	265	265	255	265
16	75	270	270	270	275	275	265	270
17	80	280	280	280	280	280	275	280
18	85	285	285	285	285	285	285	285

In Tables 1-3 the results of the proposed algorithm in the three set of instances of Tsilligirides are presented. More analytically, in the first column of each one of the Tables the number of the instance is presented, in the second column the value of the T_{max} for each instance is given, in the third column the value of the objective function produced by the proposed algorithm (MemGRASP) is given, in the fourth and fifth columns the values of the objective function of the algorithm published in [15] and of the best values from the literature (OPT) published in the same paper are presented and, finally, in sixth to nine columns the results of four algorithms presented in [6] are given. In Figure 1, three different solutions having different T_{max} for each set of the Tsilligirides

instances are presented. In Table 4, the results of the last two data sets are presented. The structure of the Table is as in the previous Tables, however as we have less instances from the literature to compare the results of the proposed algorithm, we separate the Table in two parts, where in the first part the results of the benchmark instances presented by [1] (denoted as Square-shaped instances) are given (columns 1 to 6) and in the second part the results of the benchmark instances presented by [1] (denoted as Diamond-shaped instances) (columns 7 to 12) are given, respectively. In both parts of the Table, besides the results of the proposed algorithm, the results of the algorithm presented in [15] (denoted as GLS), the results of the algorithm presented in [2] (denoted as Chao) and the results of the algorithm presented in [10] (denoted as DStPSO) are given. In Figure 2, three different solutions having different T_{max} for each set of the Chao instances are presented.

The proposed algorithm is tested in 87 instances in total. The best known solution from the literature is found by the proposed algorithm in 71 of them.

Table 2. Computational results for Tsilligirides's Set 2

	T_{max}	MemGRASP	GLS	OPT	D(R - I)	S(R - I)	Knapsack	MVP
1	15	120	120	120	120	120	120	120
2	20	200	200	200	200	200	200	200
3	23	210	210	210	210	210	210	210
4	25	230	230	230	230	230	230	230
5	27	230	220	230	230	230	230	230
6	30	265	260	265	265	260	260	260
7	32	300	300	300	300	300	275	300
8	35	320	305	320	320	320	305	320
9	38	360	360	360	355	355	355	360
10	40	395	380	395	385	395	380	380
11	45	450	450	450	450	450	450	450

Table 3. Computational results for Tsilligirides's Set 3

	T_{max}	MemGRASP	GLS	OPT	D(R - I)	S(R - I)	Knapsack	MVP
1	15	170	170	170	100	100	170	170
2	20	200	200	200	140	140	200	200
3	25	260	250	260	190	190	250	260
4	30	320	310	320	240	240	320	320
5	35	390	390	390	280	290	380	370
6	40	430	430	430	340	330	420	430
7	45	460	470	470	370	370	450	460
8	50	520	520	520	420	420	500	520
9	55	550	540	550	440	460	520	550
10	60	580	570	580	500	500	580	570
11	65	610	610	610	530	530	600	610
12	70	630	630	640	560	560	640	640
13	75	670	670	670	600	590	650	670
14	80	710	710	710	640	640	700	700
15	85	730	740	740	670	670	720	740
16	90	760	770	770	700	700	770	760
17	95	790	790	790	740	730	790	790
18	100	800	800	800	770	760	800	800
19	105	800	800	800	790	790	800	800
20	110	800	800	800	800	800	800	800

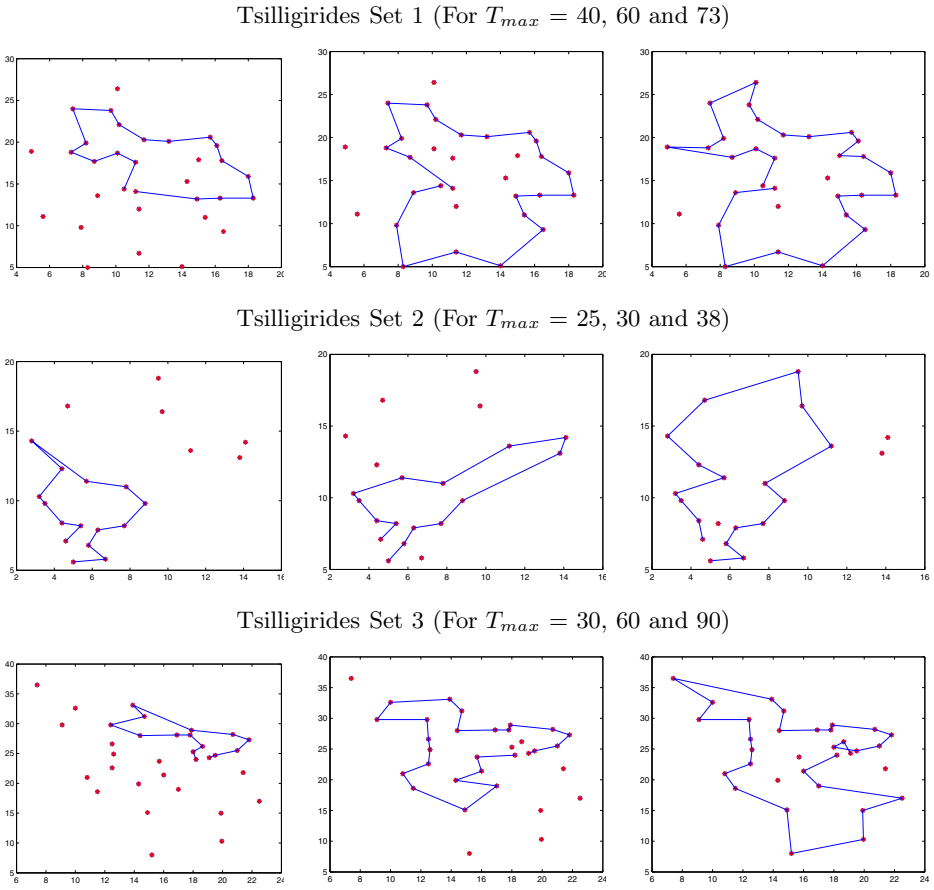

Fig. 1. Representative drawings for Tsilligirides' Sets

In the other 16 the solution found by the proposed algorithm is near to the best known solution without large deviation from the best known solution. From Figures 1 and 2, we can see that as the T_{max} is increased the number of nodes that are included in the best solution is, also, increased. It should be noted that the convergence time of the algorithm was quite satisfactory as the average time (Average CPU Time) was 44 seconds with a minimum of 7 seconds and a maximum of 164 seconds for the more demanding instance. As we observed from the figures the instances are divided in two categories. In the first one the nodes are randomly scattered in the solution space (the three first sets of benchmark instances) while in the second one the nodes are placed in a diamond shape and in a square shape. These two different distributions were the reason that these instances were selected for testing the proposed algorithm. The idea behind this algorithm, as it was analyzed earlier, was the tourists that they would like to

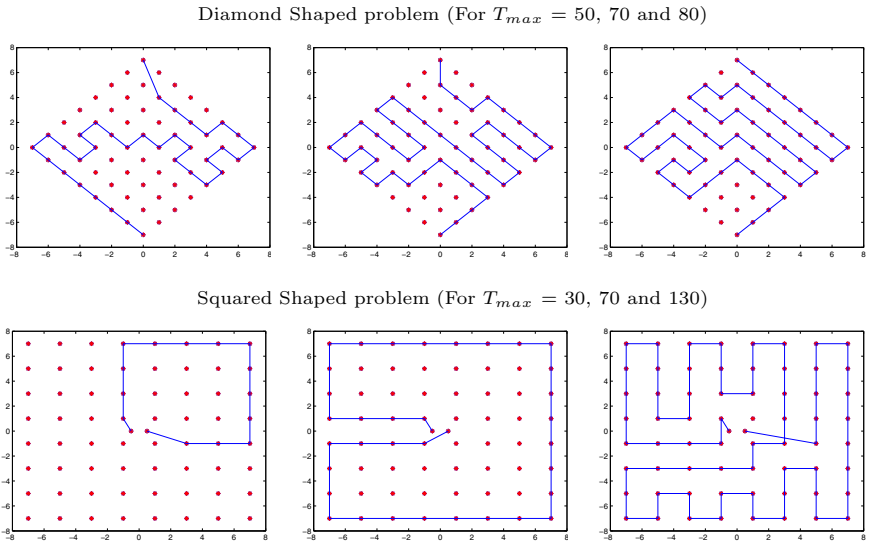

Fig. 2. Representative drawings for Chao Sets

Table 4. Computational results for Square-shaped and Diamond-shaped test sets

Square-shaped test set						Diamond-shaped test set					
	T_{max}	**Mem-GRASP**	**GLS**	**Chao**	**DStPSO**		T_{max}	**Mem-GRASP**	**GLS**	**Chao**	**DStPSO**
1	5	10	10	10	10	1	15	96	96	96	96
2	10	40	40	40	40	2	20	294	294	294	294
3	15	120	120	120	120	3	25	390	390	390	390
4	20	205	175	195	205	4	30	474	474	474	474
5	25	290	290	290	290	5	35	576	552	570	576
6	30	400	400	400	400	6	40	714	702	714	714
7	35	465	465	460	465	7	45	816	780	816	816
8	40	575	575	575	575	8	50	894	888	900	900
9	45	650	640	650	650	9	55	978	972	984	984
10	50	730	710	730	730	10	60	1062	1062	1044	1062
11	55	825	825	825	825	11	65	1116	1110	1116	1116
12	60	915	905	915	915	12	70	1188	1188	1176	1188
13	65	980	935	980	980	13	75	1230	1236	1224	1236
14	70	1070	1070	1070	1070	14	80	1278	1260	1272	1284
15	75	1140	1140	1140	1140						
16	80	1215	1195	1215	1215						
17	85	1260	1265	1270	1270						
18	90	1340	1300	1340	1340						
19	95	1385	1385	1380	1395						
20	100	1445	1445	1435	1465						
21	105	1515	1505	1510	1520						
22	110	1545	1560	1550	1560						
23	115	1590	1580	1595	1595						
24	120	1610	1635	1635	1635						
25	125	1655	1665	1655	1665						
26	130	1675	1680	1680	1680						

visit a town where a number of places of interest must be selected that are scattered in the whole town or a museum where all the paintings or sculptures are placed in rooms the one next to the other. Thus, we would like to present an efficient and fast algorithm that it will perform equally well in instances that describe both cases. The results of the proposed algorithm both in quality of the solutions and in computational time needed to converge to its best solution give us the possibility to proceed to the next step which is to include this algorithm to the tourist guide planner.

5 Conclusions and Future Research

In this paper, an algorithm for the solution of the Orienteering Problem is presented. The algorithm is the first step of a complete decision support system that will help a tourist to see the most important, based on his preferences, attractions of a city or a museum that he would like to visit during his vacations. The algorithm, denoted as Memetic-GRASP algorithm, is a hybridization of three well known algorithms, the Greedy Randomized Adaptive Search Procedure, an Evolutionary Algorithm and Local Search algorithms. The algorithm was tested in classic sets of benchmark instances for the Orienteering Problem and in most cases it found the best known solutions. The future steps of our research will be, initially to change the formulation of the problem and to add a stochastic variable in each node (point of interest) of the tourist where this variant will correspond to the waiting time in the specific point of interest and it will be activated only if this point of interest will be selected from the tourist, to apply this algorithm in the classic Team Orienteering Problem and in the stochastic Team Orienteering Problem and, finally, to develop a decision support system (tourist guide planner) in which the user will add, initially, his preferences and, then, the algorithm will solve either an Orienteering Problem if he would like to find one path or a Team Orienteering Problem if he would like to find multiple paths.

References

1. Chao, I.: Algorithms and solutions to multi-level vehicle routing problems. Ph.D. Dissertation, Applied Mathematics Program, University of Maryland, College Park, USA (1993)
2. Chao, I.M., Golden, B.L., Wasil, E.: A fast and effective heuristic for the Orienteering Problem. European Journal of Operational Research 88, 475–489 (1996)
3. Chao, I.M., Golden, B.L., Wasil, E.: The team orienteering problem. European Journal of Operational Research 88, 464–474 (1996)
4. Feo, T.A., Resende, M.G.C.: Greedy randomized adaptive search procedure. Journal of Global Optimization 6, 109–133 (1995)
5. Golden, B., Levy, L., Vohra, R.: The orienteering problem. Naval Research Logistics 34, 307–318 (1987)
6. Keller, C.P.: Algorithms to solve the orienteering problem: A comparison. European Journal of Operational Research 41, 224–231 (1989)

7. Montemanni, R., Gambardella, L.: Ant colony system for team orienteering problems with time windows. Foundations of Computing and Decision Sciences 34(4), 287–306 (2009)
8. Moscato, P., Cotta, C.: A gentle introduction to memetic algorithms. In: Glover, F., Kochenberger, G.A. (eds.) Handbooks of Metaheuristics, pp. 105–144. Kluwer Academic Publishers, Dordrecht (2003)
9. Righini, G., Salani, M.: Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. Computers and Operations Research 4, 1191–1203 (2009)
10. Sevkli, Z., Sevilgen, F.E.: Discrete particle swarm optimization for the orienteering problem. In: 2010 IEEE Congress on Evolutionary Computation (CEC), Barcelona, Spain (2010), doi:10.1109/CEC.2010.5586532
11. Souffriau, W., Vansteenwegen, P., Vertommen, J., Vanden Berghe, G., Van Oudheusden, D.: A personalized tourist trip design algorithm for mobile tourist guides. Applied Artificial Intelligence 22(10), 964–985 (2008)
12. Tang, H., Miller-Hooks, E.: A TABU search heuristic for the team orienteering problem. Computer and Industrial Engineering 32, 1379–1407 (2005)
13. Tsiligirides, T.: Heuristic methods applied to orienteering. Journal of Operational Research Society 35, 797–809 (1984)
14. Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D.: Metaheuristics for tourist trip planning. In: Geiger, M., Habenicht, W., Sevaux, M., Sorensen, K. (eds.) Metaheuristics in the Service Industry. Lecture Notes in Economics and Mathematical Systems, vol. 624, pp. 15–31 (2009)
15. Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D.: A guided local search metaheuristic for the team orienteering problem. European Journal of Operational Research 196, 118–127 (2009)
16. Vansteenwegen, P., Souffriau, W., Van Oudheusden, D.: The orienteering problem: A survey. European Journal of Operational Research 209, 1–10 (2011)
17. Yu, V.F., Lin, S.W., Chou, S.Y.: The museum visitor routing problem. Applied Mathematics and Computation 216, 719–729 (2010)