A Greedy Randomised Adaptive Search Procedure for the Team Orienteering Problem

Wouter Souffriau*,† Pieter Vansteenwegen† Greet Vanden Berghe*,‡
Dirk Van Oudheusden†

*Information Technology, KaHo Sint-Lieven
Gebroeders Desmetstraat 1, 9000 Gent, Belgium
{Wouter.Souffriau, Greet.Vandenberghe}@kahosl.be

†Centre for Industrial Management, Katholieke Universiteit Leuven
Celestijnenlaan 300A, 3001 Leuven (Heverlee), Belgium
{Pieter.Vansteenwegen, Dirk.Vanoudheusden}@cib.kuleuven.be

‡Department of Computer Science, Katholieke Universiteit Leuven
Celestijnenlaan 200A, 3001 Leuven (Heverlee), Belgium

Abstract

This paper introduces a Greedy Randomised Adaptive Search Procedure (GRASP) for solving the Team Orienteering Problem (TOP). That is a particular routing problem in which a score is earned for visiting a location. The objective is to maximise the sum of the scores of a fixed number of routes, while not exceeding a given time budget for each route. The memoryless GRASP approach is extended by adding a long term adaptive memory component, namely a path relinking procedure. Both versions of the approach are tested using a large set of test instances from the literature.

Keywords: Team Orienteering Problem, GRASP.

1 Introduction

In the Orienteering Problem (OP) a set of n locations i is given, each with a score S_i. The starting point (location 1) and the end point (location n) are fixed. The time t_{ij} needed to travel from location i to j is known for all locations. Not all locations can be visited since the available time is limited to a given time budget T_{max}. The goal of the OP is to determine a route, limited by T_{max}, that visits some of the locations, in order to maximise the total collected score. Each location can be visited at most once. The Team Orienteering Problem (TOP) is an OP for which the goal is to determine m routes, each limited to T_{max}, that maximise the total collected score.

The OP [28] is also known as the selective travelling salesperson problem [16], the maximum collection problem [4] and the bank robber problem [2]. The OP can be formulated as a special case of the resource constrained travelling salesperson problem [20], as a travelling salesperson problem with profits [8] or as a resource constrained elementary shortest path problem [24]. Many (T)OP applications have been described in the literature: the sport game of orienteering [7], the home fuel delivery problem [13], athlete recruiting from high schools [4], routing technicians to service customers [26], etc. The application addressed in this paper, is the personalised electronic tourist guide. A device should at short notice suggest a (near) optimal combination of tourist attractions and a route between them, taking into account the weather, opening hours, crowded places and personal preferences. The problem is called the Tourist Trip Design Problem [30]. The OP is the simplest form of this problem and has been successfully used as a model for calculating personal tourist trips [25].

The remaining of this paper is structured as follows: Section 2 provides an overview of the literature on the (T)OP. Section 3 describes the GRASP approach to tackle the TOP. Section 4 elaborates on long term memory used to extend the basic GRASP approach. Section 5 presents the results of the proposed approach on different test sets available from the literature. Finally, Section 6 concludes the paper and points out directions for further research.

2 Literature Review

Several researchers propose exact algorithms to solve the OP based on a branch-and-bound [17] and branch-and-cut approach [10]. With the branch-and-cut procedure, instances up to 500 locations can be solved [10] to optimality, provided that sufficient calculation time is available. Since the OP is NP-hard [13], it is obvious that these exact algorithms are very time consuming and that most research has focused on heuristic approaches. Tsiligirides [28] proposed a stochastic (S-Algorithm) and a deterministic algorithm (D-Algorithm). Golden et al. [13] developed a centre-of-gravity heuristic and improved it later by adding learning capabilities and the randomisation idea from the S-Algorithm. The five-step heuristic of Chao et al. [6], implementing an efficient initialisation step, three improvement steps and one diversification step, clearly outperforms all above-mentioned heuristics. More algorithms [11, 14, 18, 27] were developed to tackle the OP, but none of them improved the state-of-the-art algorithms significantly.

Butt et al. [5] present an exact algorithm based on column generation to solve the TOP. They were able to solve problems with up to 100 locations, provided that the number of locations in each tour remains small. Boussier et al. [3] propose a branch-and-price approach to solve problems with up to 100 locations. Only problems in which the number of possible locations in a route is low (up to about 15 per route) can be solved in less than two hours of calculation.

The first published TOP heuristic was developed by Chao et al. [6] and is more or less the same as their five-step heuristic for the OP. Tang et al. [26] developed a tabu search heuristic embedded in an adaptive memory procedure. The best published heuristics are described in Archetti et al. [1]. They developed two variants of a tabu search heuristic and a slow and fast Variable Neighbourhood Search (VNS) algorithm. One tabu search algorithm only considers feasible solutions while the other one also accepts infeasible solutions. The

VNS applies tabu search with much fewer iterations and it only considers feasible solutions. Archetti et al. [1] present the best results among the above-mentioned algorithms for the TOP. The strength of the algorithm probably lies in the fact that all non-included locations are also grouped in feasible tours, while in other algorithms only included locations are organised in tours. Vansteenwegen et al. [31, 32] implemented a Guided Local Search (GLS) and a Skewed Variable Neighbourhood Search (SVNS) algorithm. The results of these algorithms are comparable with the results of Archetti et al. [1]. The quality is slightly worse but the computational effort is reduced significantly.

3 GRASP approach

Greedy Randomised Adaptive Search Procedure (GRASP) is a metaheuristic that was first introduced by Feo and Resende [9]. In general, GRASP performs a number of iterations that consist of a constructive procedure followed by a local search approach. The behaviour of the constructive procedure is governed by only one parameter (the so–called greediness), which prescribes a precise ratio between greediness and randomness. The different iterations are independent and the best solution found is saved and returned as the result. An overview of successful GRASP applications can by found in [23] and examples in the field of vehicle routing problems are [21, 22]. To the best of the authors' knowledge, GRASP has not been applied to the TOP.

Algorithm Listing 1 illustrates one iteration of GRASP for the TOP. Each iteration starts from an empty solution of m identical routes, which only contain the start and end location. In order to construct an initial solution, a Candidate List (CL) of feasible insert moves is generated. An insert move IM_{ilj} considers the insertion of an unvisited location l into the current solution between locations i and j. IM_{ilj} is characterised by an additional score S_l, and an increase of the duration, equal to $t_{ilj} = t_{il} + t_{lj} - t_{ij}$. Only feasible insert moves are considered, which means that the time increase of the tour has to be less than or equal to the remaining time budget. All feasible moves are collected into the CL and for each of them, a heuristic value $h_{ilj} = \frac{S_l}{t_{ilj}}$ is calculated. Based on the minimum and maximum heuristic values of the candidate list and on the greediness parameter, a threshold is computed. The original CL is filtered and only members with a heuristic value that exceeds the threshold are collected into the Restricted Candidate List (RCL). A random move from this RCL is picked and applied to the current solution. This construction process is repeated until no more feasible insertion candidates are found.

At the end of each iteration, the constructed solution is improved by a local search procedure, which consists of a sequence of best improving searches within four different neighbourhoods. First, in order to reduce the total time, a standard 2-Opt neighbourhood [19] is explored and a swap neighbourhood is introduced to exchange locations between tours. Next, replace moves try to exchange locations that belong to the current solution with locations outside. Finally, insert moves are searched again. The different neighbourhoods are explored consecutively until no further improvement is found.

```
Solution = empty solution;
CL = Generate Insert Moves;
while CL is not empty do
    Min = min heuristic value in CL;
    Max = max heuristic value in CL;
    Threshold = Min + greediness × (Max - Min);
    Restricted CandidateList (RCL) = empty;
    for Each Candidate c of CL do
        if Heuristic value of c ≥ Threshold then
            Add c to RCL;
        end
    end
    FinalCandidate = Pick random candidate from RCL;
    Solution = Apply FinalCandidate;
    CL = Generate Insert Moves;
end
Solution = localSearch(Solution);
```

Algorithm 1: One iteration of GRASP for the TOP

4 Path Relinking

A major shortcoming of the basic GRASP approach is its independence of subsequent iterations. Except for maintaining the best solution, every iteration starts again from an empty solution, regardless previously detected promising regions in the search space. In order to tackle this problem, an extension to the basic GRASP application for the TOP is introduced. A long term memory component is added to the basic approach, namely path relinking. Path relinking was originally introduced by Glover [12] and applied in combination with GRASP by Laguna and Martí [15].

Algorithm Listing 2 describes the path relinking approach. Until, during a fixed number of iterations, no further improvements are identified, the same construct and local search procedures, as described in the previous section, are carried out. Next, the actual path relinking is carried out by exploring the paths between the resulting solution and the elements of a pool of elite solutions pairwise in both directions. This is explained in further detail below. Finally, the pool of elite solutions is updated.

```
while Nr of iterations without improvement is not exceeded do
    construct;
    local search;
    link to elites;
    update elite pool;
end
Return best found;
```

Algorithm 2: Path relinking extension

Algorithm Listing 3 describes the Link to elites procedure in more detail. This procedures takes two solutions as arguments, named the starting solution and the guiding solution, and

searches the path between these for better solutions. From the guiding solution, all locations in common are substracted to end up with the "set of locations to add". Next, the locations of the set of features to add are added one by one to the starting solution by executing insert moves. In contrast to the insert moves of the previous Section, a location is also considered for insertion when it violates the time budget if the location belongs to the set of features to add and the route is not infeasible at the moment. When all m routes are infeasible, no more locations are inserted and a restore procedure restores feasibility for each route. For every location l that belongs to an infeasible route and is located between i and j, $\frac{1}{h_{ilj}} = \frac{t_{ilj}}{S_l}$ is calculated. The location l with the highest $\frac{1}{h_{ilj}}$ is removed from the route. This removal is repeated until each route is feasible again.

Set intersection = common locations of both solutions;
Set featuresToAdd = guiding solution minus intersection;
while *featuresToAdd is not empty* **do**
  Insert locations, allow infeasibility;
  Delete locations to restore feasibility;
end

Algorithm 3: Link to elites procedure

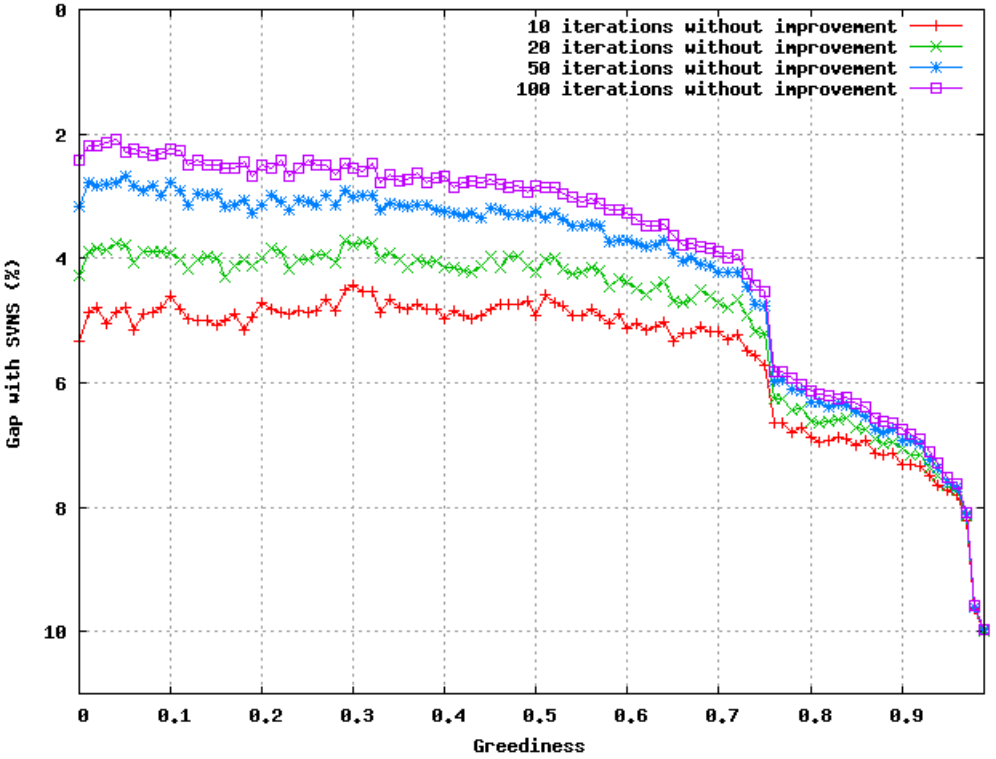
The best solution found by Link to elites is considered for insertion into the pool of elite solutions. The similarity measure between the best solution found and the other members of the elite pool are calculated. Following formula is used to calculate the similarity between solutions X and Y: $\frac{2n_{X\cap Y}}{n_X+n_Y}$, where n_X and n_Y are the number of locations of solutions X and Y, and $n_{X\cap Y}$ the number of locations that the two solutions have in common. If the elite pool has not reached its maximum number of members, and the similarity measures between the solution under consideration and all elites are below a threshold value, the solution is added to the pool. If the solution under consideration is too similar to a member of the elite pool, it is not inserted into the pool, except when the solution is better than the best elite solution. If the elite pool is full and the solution under consideration is better than the worst elite solution, the worst elite solution is replaced by the solution under consideration.

5 Experimental Results

In this section the experimental results of the GRASP algorithm are presented. The algorithm is coded using JAVA 1.6. All experiments are performed on an Intel Xeon 2.5GHz. A selection of problems of Fischetti et al. [10] and Chao et al. [7] has been addressed to validate the proposed algorithm. More details about the selected test problems can be found in [29]. As point of comparison, the SVNS approach of Vansteenwegen et al. [31] is used. The results of this algorithm deviate from the best known results by only 0.67% on average.

The basic GRASP approach is tested using different settings of the greediness parameter and the stopping criterium. The greediness parameter ranges from 0.00 to 0.99 and the stopping criterium is 10, 20, 50 and 100 iterations without improvement. Each combination of parameter settings is tested on all instances and their results are compared to the SVNS approach and averaged. Figure 1 compares the results of the basic GRASP approach to the SVNS approach in function of the greediness parameter and the stopping criterium. The

Figure 1: GRASP results



graph shows a significant detoriation of the solution quality for a greediness setting above 0.75, for all stopping criteria. In the case of 10 and 20 iterations without improvement, a greediness of 0.30 shows the best performance. In the case of 50 and 100 iterations without improvement, greediness settings close to 0 perform the best, as these settings correspond to a high diversification in the search space.

Table 1 presents the execution time for each stopping criterium, averaged over the different settings of the greediness parameters and the different test instances. The average execution time of the SVNS algorithm is 750 ms, using the same hardware configuration.

Table 1: GRASP average execution times

#iterations without improvement	10	20	40	100
time (ms)	76	145	285	674

Preliminary tests of the path relinking extension reveal an average that is 0.31% better than the SVNS approach. This is a considerable improvement as most of the test problems are solved (nearly) optimally by the SVNS algorithm. This result is achieved with a uniformly randomly sampled greediness, 100 iterations without improvement as the stopping criterium, a pool size of 10 and a maximum similarity threshold of 0.95. The execution time however equals 7900 ms, again using the same hardware configuration.

6 Conclusion and Future Research Directions

This paper presents the application of the GRASP metaheuristic to the Team Orienteering Problem. The basic GRASP approach achieves a 2% gap with a state of the art SVNS approach, in a comparable execution time. Next to the basic memoryless version, a path relinking extension is presented. Preliminary results are promising and achieve a 0.31% gap improvement over the SVNS approach.

Further investigation regarding problem characteristics is needed to determine a greediness parameter function for the basic GRASP approach, instead of a constant. The authors suspect that regression could be used to predict an appropriate greediness parameter based on the characteristics of the problem instance. The path relinking extension should be carefully tested and analysed in order to further improve the solution quality and equally important, reduce execution times.

References

[1] C. Archetti, A. Hertz, and M. Speranza. Metaheuristics for the team orienteering problem. *Journal of Heuristics*, 13:49–76, 2007.

[2] E. Arkin, J. Mitchell, and G. Narasimhan. Resource-constrained geometric network optimization. In *Proceedings of the 14th ACM Symposium on Computational Geometry*, pages 307–316, 1998.

[3] S. Boussier, D. Feillet, and M. Gendreau. An exact algorithm for the team orienteering problem. *4OR*, 5:211–230, 2007.

[4] S. Butt and T. Cavalier. A heuristic for the multiple tour maximum collection problem. *Computers and Operations Research*, 21:101–111, 1994.

[5] S. Butt and D. Ryan. An optimal solution procedure for the multiple tour maximum collection problem using column generation. *Computer and Operations Research*, 26:427–441, 1999.

[6] I. Chao, B. Golden, and E. Wasil. Theory and methodology - the team orienteering problem. *European Journal of Operational Research*, 88:464–474, 1996.

[7] I.-M. Chao, B. Golden, and E. Wasil. A fast and effective heuristic for the orienteering problem. *European Journal of Operational Research*, 88(3):475–489, 1996.

[8] D. Feillet, P. Dejax, and M. Gendreau. Traveling salesman problems with profits. *Transportation Science*, 39:188–205, 2005.

[9] T.A. Feo and M.G.C. Resende. A probabilistic heuristic for a computationally difficult set covering problem. *Operations Research Letters*, 8:67–71, 1989.

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

[11] M. Gendreau, G. Laporte, and F. Semet. A branch-and-cut algorithm for the undirected selective travelling salesman problem. Technical Report CRT-95-80, Centre de recherche sur les transports, Montreal, Canada, 1995.

[12] F. Glover. *Interfaces in Computer Science and Operations Research*, chapter Tabu search and adaptive memory programming, pages 1–75. Kluwer, 1996.

[13] B. Golden, L. Levy, and R. Vohra. The orienteering problem. *Naval Research Logistics*, 34:307–318, 1987.

[14] P. Keller. Algorithms to solve the orienteering problem: A comparison. *European Journal of Operational Research*, 41:224–231, 1989.

[15] M. Laguna and R. Martí. Grasp and path relinking for 2-layer straight line crossing minimization. *INFORMS Journal on Computing*, 11:44–52, 1999.

[16] G. Laporte. The vehicle routing problem: An overview of exact and approximate algorithms. *European Journal of Operational Research*, 59:345–358, 1992.

[17] G. Laporte and S. Martello. The selective travelling salesman problem. *Discrete applied mathematics*, 26:193–207, 1990.

[18] Y. Liang, S. Kulturel-Konak, and A. Smith. Meta heuristics for the orienteering problem. In *Proceedings of the 2002 Congress on Evolutionary Computation*, pages 384–389, Honolulu, Hawai, 2002.

[19] S. Lin. Computer solutions of the traveling salesman problem. *Bell System Technical Journal*, 44:2245–2269, 1965.

[20] J. Pekny and D. Miller. An exact parallel algorithm for the resource constrained travelling salesman problem with application to scheduling with an aggregate deadline. In *Proceedings of the ACM Annual Conference on Cooperation*, 1990.

[21] C. Prins, C. Prodhon, and R. Wolfler Calvo. Solving the capacitated location-routing problem by a grasp complemented by a learning process and a path relinking. *4OR*, 4:221–238, 2006.

[22] M. Reghioui, C. Prins, and N. Labadi. Grasp with path relinking for the capacitated arc routing problem with time windows. In *EvoWorkshops*, pages 722–731, 2007.

[23] M.G.C. Resende and C.C. Ribeiro. *Handbook of Metaheuristics*, chapter Greedy randomized adaptive search procedures, pages 219–249. Kluwer Academic Publishers, 2003.

[24] G. Righini and M. Salani. Dynamic programming for the orienteering problem with time windows. Technical Report 91, Dipartimento di Tecnologie dell Informazione, Universita degli Studi Milano, Crema, Italy, 2006.

[25] W. Souffriau, P. Vansteenwegen, J. Vertommen, G. Vanden Berghe, and D. Van Oudheusden. A personalised tourist trip design algorithm for mobile tourist guides. *Applied Artificial Intelligence*, to appear, DOI: 10.1080/08839510802379626.

[26] H. Tang and E. Miller-Hooks. A tabu search heuristic for the team orienteering problem. *Computer and Operations Research*, 32:1379–1407, 2005.

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

[28] T. Tsiligirides. Heuristic methods applied to orienteering. *J. Oper. Res. Soc.*, 35(9):797–809, 1984.

[29] P. Vansteenwegen. *Planning in tourism and public transportation.* PhD thesis, K.U. Leuven, 2008.

[30] P. Vansteenwegen and D. Van Oudheusden. The Mobile Tourist Guide: an OR Opportunity. *OR Insight*, 20(3):21–27, 2007.

[31] P. Vansteenwegen, W. Souffriau, G. Vanden Berghe, and D. Van Oudheusden. *Metaheuristics in the Service Industry*, chapter Metaheuristics for tourist trip planning. Lecture Notes in Economics and Mathematical Systems. Springer Verlag, to appear.

[32] P. Vansteenwegen, W. Souffriau, G. Vanden Berghe, and D. Van Oudheusden. A guided local search metaheuristic for the team orienteering problem. *European Journal of Operational Research*, to appear, DOI: 10.1016/j.ejor.2008.02.037.