European Journal of Operational Research 209 (2011) 1–10

Contents lists available at ScienceDirect

European Journal of Operational Research

journal homepage: www.elsevier.com/locate/ejor

Invited Review

The orienteering problem: A survey

Pieter Vansteenwegen [a,*], Wouter Souffriau [a,b], Dirk Van Oudheusden [a]

[a] Centre for Industrial Management, Katholieke Universiteit Leuven, Celestijnenlaan 300A – bus 2422, 3001 Leuven, Belgium
[b] Information Technology, Katholieke Hogeschool Sint-Lieven, Gebroeders Desmetstraat 1, 9000 Gent, Belgium

ARTICLE INFO

Article history:
Received 14 May 2009
Accepted 30 March 2010
Available online 2 April 2010

Keywords:
Combinatorial optimisation
Orienteering problem
Survey

ABSTRACT

During the last decade, a number of challenging applications in logistics, tourism and other fields were modelled as orienteering problems (OP). In the orienteering problem, a set of vertices is given, each with a score. The goal is to determine a path, limited in length, that visits some vertices and maximises the sum of the collected scores. In this paper, the literature about the orienteering problem and its applications is reviewed. The OP is formally described and many relevant variants are presented. All published exact solution approaches and (meta) heuristics are discussed and compared. Interesting open research questions concerning the OP conclude this paper.

1. Introduction

The name "Orienteering Problem" (OP) originates from the sport game of orienteering (Chao et al., 1996b). In this game, individual competitors start at a specified control point, try to visit as many checkpoints as possible and return to the control point within a given time frame. Each checkpoint has a certain score and the objective is to maximise the total collected score. The OP is a combination of vertex selection and determining the shortest Hamiltonian path between the selected vertices. As a consequence, the OP can be seen as a combination between the Knapsack Problem (KP) and the Travelling Salesperson Problem (TSP). The OP's goal is to maximise the total score collected, while the TSP tries to minimise the travel time or distance. Furthermore, not all vertices have to be visited in the OP. Determining the shortest path between the selected vertices will be helpful to visit as many vertices as possible in the available time.

The OP is also known as the selective travelling salesperson problem (Laporte and Martello, 1990; Gendreau et al., 1998b; Thomadsen and Stidsen, 2003), the maximum collection problem (Kataoka and Morito, 1988; Butt and Cavalier, 1994) and the bank robber problem (Arkin et al., 1998). During the last decade, a number of challenging practical applications were modelled as orienteering problems and many exact and heuristic solution approaches were published.

The surveys about the TSP with profits (Feillet et al., 2005a) and the Hamiltonian and non-Hamiltonian problems (Laporte and Rodriguez Martin, 2007) clearly situate the OP between other routing problems (with and without profits) and indicate the differences. Both papers briefly discuss the OP, some solution strategies and a few extensions and variants.

Our paper focuses entirely on the OP itself, discussing the literature about its extensions and variants and many solution strategies and applications. The paper is organised as follows. Sections 2–5 define and discuss, respectively, the orienteering problem, the team orienteering problem, the orienteering problem with time windows and the team orienteering problem with time windows. Each of these sections presents a formal problem definition and a mathematical formulation, together with an overview of the applications, benchmark instances and solution approaches in the literature. Section 6 discusses several other variants of the OP. Section 7 concludes the paper and points out some interesting open research questions.

2. Orienteering problem

2.1. Problem definition and mathematical formulation

In the OP, a set of N vertices i is given, each with a score S_i. The starting point (vertex *1*) and the end point (vertex N) are fixed. The time t_{ij} needed to travel from vertex i to j is known for all vertices. Not all vertices 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 path, limited by T_{max}, that visits some of the vertices, in order to maximise the total collected score. The scores are assumed to be entirely additive and each vertex can be visited at most once.

The OP can also be defined with the aid of a graph G=(V,A) where $V=\{v_1,\ldots,v_N\}$ is the vertex set and A is the arc set. In this definition the nonnegative score S_i is associated with each vertex

* Corresponding author. Tel.: +32 16 32 25 67; fax: +32 16 32 29 86.
E-mail addresses: Pieter.Vansteenwegen@cib.kuleuven.be (P. Vansteenwegen), Wouter.Souffriau@kahosl.be (W. Souffriau), Dirk.VanOudheusden@cib.kuleuven.be (D.V. Oudheusden).

0377-2217/$ - see front matter © 2010 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2010.03.045

$v_i \in V$ and the travel time t_{ij} is associated with each arc $a_{ij} \in A$. The OP consists of determining a Hamiltonian path G' ($\subset$G) over a subset of V, including preset start (v_1) and end (v_N) vertex, and having a length not exceeding T_{max}, in order to maximise the total collected score. In most cases, the OP is defined as a path to be found between distinct vertices, rather than a circuit or tour ($v_1 \equiv v_N$). In many applications, however, v_1 does coincide with v_N. The difference between both formulations is small. It is always possible to add a dummy arc between end and start vertex to turn a path problem into a circuit problem. Mansini et al. (2006) explicitly define the "tour orienteering problem" as an OP where the start and end vertex coincide.

Making use of the notation introduced above, the OP can be formulated as an integer problem. The following decision variables are used: $x_{ij} = 1$ if a visit to vertex i is followed by a visit to vertex j – 0 otherwise; $u_i =$ the position of vertex i in the path.

$$Max \sum_{i=2}^{N-1} \sum_{j=2}^{N} S_i x_{ij}, \tag{0}$$

$$\sum_{j=2}^{N} x_{1j} = \sum_{i=1}^{N-1} x_{iN} = 1, \tag{1}$$

$$\sum_{i=1}^{N-1} x_{ik} = \sum_{j=2}^{N} x_{kj} \leqslant 1; \quad \forall k = 2, \ldots, N-1, \tag{2}$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} t_{ij} x_{ij} \leqslant T_{max}, \tag{3}$$

$$2 \leqslant u_i \leqslant N; \quad \forall i = 2, \ldots, N, \tag{4}$$

$$u_i - u_j + 1 \leqslant (N-1)(1 - x_{ij}); \quad \forall i, j = 2, \ldots, N, \tag{5}$$

$$x_{ij} \in \{0, 1\}; \quad \forall i, j = 1, \ldots, N. \tag{6}$$

The objective function (0) is to maximise the total collected score. Constraints (1) guarantee that the path starts in vertex *1* and ends in vertex *N*. Constraints (2) ensure the connectivity of the path and guarantee that every vertex is visited at most once. Constraint (3) ensures the limited time budget. Constraints (4) and (5) are necessary to prevent subtours. These subtour elimination constraints are formulated according to the Miller–Tucker–Zemlin (MTZ) formulation of the TSP (Miller et al., 1960).

Notwithstanding the fact that the given formulations are general, all applications and solution approaches in the literature assume a Euclidean metric having symmetric travel times between the vertices ($t_{ij} = t_{ji}$). This corresponds to an undirected complete graph G. Most solution approaches can be modified easily to deal with a directed and incomplete graph G.

Golden et al. (1987) prove that the OP is NP-hard; no polynomial time algorithm has been designed, or is expected to be designed, to solve this problem to optimality. This implies that exact solution algorithms are very time consuming and for practical applications heuristics will be necessary. Moreover, Gendreau et al. (1998a) give a few reasons why it is so difficult to design good heuristics for the OP. The score of a vertex and the time to reach the vertex are independent and often contradictory to each other. This makes it very difficult to select the vertices that will be part of the optimal solution. Therefore, simple construction and improvement heuristics may direct the algorithm in undesirable directions, they do not efficiently explore the whole solution space and wrong decisions are unsatisfactorily corrected. Vansteenwegen (2008) indicated that the most difficult OP instances to solve are those where the selected number of vertices is a little more than half of the total number of vertices. If the time budget allows the selection of half of the vertices, the highest number of selections will have to be evaluated by the algorithm. Moreover, determining a (shortest) path between the selected vertices becomes more complicated when the number of vertices increases.

2.2. Practical applications

The OP arises in several applications. A first application was mentioned by Tsiligirides (1984), in the case of a travelling salesperson with not enough time to visit all possible cities. He knows the number of sales to expect in each city and wants to maximise total sales, while keeping the total travel time limited to a day (or week). Golden et al. (1987) describe the home fuel delivery problem. A fleet of trucks has to deliver to a large number of customers on a daily basis. The customers' fuel inventory level should be maintained at an adequate level at all times. The forecasted inventory level can be considered as a measure of urgency. A primary goal is to select a subset of customers to be visited each day who urgently require a delivery and are clustered in such a way that efficient truck paths can be constructed. This subset selection step is modelled as an OP where the urgency of delivery at a customer is used as the score. It is the first step of the larger inventory and routing problem of home fuel delivery. Other steps include the assignment of customers to vehicles and constructing efficient paths for each truck, between the assigned customers. Another application mentioned in the literature is the single-ring design problem when building telecommunication networks (Thomadsen and Stidsen, 2003).

Another recent application is the Mobile Tourist Guide (Souffriau et al., 2008). For tourists visiting a city or region it is often impossible to visit everything they are interested in. Thus, they have to select what they believe to be the most valuable attractions. Making a feasible plan in order to visit these attractions in the available time span is often a difficult task. These planning problems are called Tourist Trip Design Problems (TTDP) (Vansteenwegen and Van Oudheusden, 2007). The orienteering problem is the simplest model of the TTDP. This application requires high quality solutions in only a few seconds of calculation time.

In tourist applications the time required to visit a certain vertex plays an important role in the selection of vertices. Nevertheless, the visiting time of a vertex is not mentioned explicitly in the mathematical formulation. The visiting time of a certain vertex can easily be modelled as part of the travel time to reach (or depart from) that vertex. Typically, half of the visiting time will be added to the travel time of all incoming arcs and half of it will be added to all outgoing arcs.

A similar tourist trip problem of selecting the most interesting combination of attractions is mentioned by Wang et al. (2008) and Schilde et al. (2009). Their model and solution approach is discussed in more detail in Section 6. Wang et al. (2008) also propose a military application. When a submarine or an unmanned aircraft is involved in surveillance activities, the length of the expedition is limited by a fuel or time constraint and the goal is to visit and photograph the best subset of all possible vertices.

2.3. Benchmark instances

Orienteering problem benchmark instances are available from Tsiligirides (1984), Chao (1993), Chao et al. (1996b) and Fischetti et al. (1998). The characteristics of these problems are presented in Table 1. For every set of instances, the corresponding reference is given, together with the number of instances and the number of vertices (N). In total, 385 instances are available and the number of vertices varies between 21 and 500.

The first set of Fischetti et al. (1998) is based on three OP instances of Tsiligirides (1984) and twelve VRP instances from the literature, each with three different values for T_{max}. The second set is based on 44 instances from the TSP literature, with three different ways to generate the score of each vertex. Set three and four are randomly generated in different ways. Set three has eleven dif-

Table 1
Benchmark OP instances.

Reference	Number of test instances	Number of vertices (N)
Tsiligirides (1984)	18	32
	11	21
	20	33
Chao (1993) and Chao et al. (1996b)	26	66
	14	64
Fischetti et al. (1998)	3*15	21–262
	3*44	47–400
	4*11	25–500
	5*15	21–301

ferent values for N and four different values for T_{max}. Set four has 15 different values for N and five different values for T_{max}.

All these benchmark instances are available via <www.mech.kuleuven.be/cib/op>.

2.4. Solution approaches

Several researchers propose exact algorithms to solve the OP. Laporte and Martello (1990) and Ramesh et al. (1992) use branch-and-bound to solve instances with less than 20 and 150 vertices, respectively. Leifer and Rosenwein (1994) add some valid inequalities to the formulation of Laporte and Martello and use a cutting plane method to obtain better upper bounds. Fischetti et al. (1998) and Gendreau et al. (1998b) introduce more valid inequalities and propose branch-and-cut algorithms. With the branch-and-cut procedure instances up to 500 vertices can be solved (Fischetti et al., 1998). Feillet et al. (2005a) present a classification of these exact algorithms and details about the applied techniques.

Many researchers propose heuristics to tackle the OP. Tsiligirides (1984) proposes a stochastic (S-Algorithm) and a deterministic (D-Algorithm) algorithm. The S-Algorithm is based on generating many paths and selecting the best. It uses a Monte-Carlo method to select the next vertex to be added to the path. The probability of selection is based on the necessary Euclidean distance and the score of the vertex under consideration. The D-Algorithm uses a variant of the vehicle routing procedure of Wren and Holliday (1972). Paths are built up in separate sectors of the geographic region according to some predefined rules. By varying the sectors, 48 paths are generated of which the best one is selected. Both algorithms are extended with a path-improvement algorithm (R-I-Algorithm).

Golden et al. (1987) develop a centre-of-gravity heuristic making use of a Euclidean metric. The first step is a path construction process which iteratively inserts vertices with a high score and reasonable duration. The second step is an improvement procedure which uses 2-Opt (Lin, 1965) and cheapest insertion. The third step is the centre-of-gravity step in which a new path is composed by ranking all vertices based on the ratio of their score over their distance to the centre-of-gravity from the previous path. Starting from an empty path, vertices are added according to the ranking, using cheapest insertion. If no more vertices can be added, steps 2 and 3 are performed again. In another heuristic, the centre of gravity idea, learning capabilities and the randomisation idea from the S-Algorithm were incorporated into an improved heuristic (Golden et al., 1988).

Ramesh and Brown (1991) introduce a four-phase heuristic. An insertion phase relaxes the time constraint and a cost improvement phase uses 2-Opt and 3-Opt. Next, a reduction of the path length is achieved by deleting and inserting one vertex. Finally, as many vertices as possible are included. The last three phases are repeated. The four phases are integrated in a framework with five control parameters to guide the search process.

The five-step heuristic of Chao et al. (1996b) only considers vertices that can be reached. In a Euclidean space, these vertices lie within an ellipse using start and end vertex as foci and T_{max} as length of the major axis. The initialisation step creates many different paths, each starting with a vertex far away from start and end vertex, and always assigns all other vertices to one of the paths using cheapest insertion. The best path is selected as the initial solution T_{op}. The non-included vertices are also assigned to feasible paths T_{nop}. The first improvement step, two-point exchange, tries to improve T_{op} by including an extra vertex from one of the T_{nop} and moving an included vertex to one of the T_{nop}, also using cheapest insertion. All paths have to remain feasible and a small decrease of the total score is allowed. The second improvement step will place one vertex from one path to another if feasible and if the total score does not decrease too much. The third improvement step involves 2-Opt. Finally, a specified number of vertices with a low score over insertion cost ratio are removed from the optimal path and the algorithm restarts. The five-step heuristic of Chao et al. clearly outperforms all above-mentioned heuristics.

Gendreau et al. (1998a) propose a tabu search heuristic that iteratively inserts clusters of vertices in the path or removes a chain of vertices. Compared to the previous methods, this heuristic reduces the chance to get trapped in a local optimum and the probability to include a high scoring vertex very far from the current path. More algorithms were developed to tackle the orienteering problem, but none of them improved significantly the state-of-the-art algorithms. Keller (1989) successfully applies his heuristic for the more general multi-objective vending problem, Wang et al. (1995) propose an artificial neural network approach, Gendreau et al. (1995) propose H1 and H2, both based on a composite heuristic for the TSP, Tasgetiren (2001) proposes the first genetic algorithm designed for the OP and Liang et al. (2002) develop and compare an ant colony optimisation approach and a tabu search algorithm.

The most recent solution approach for the OP is presented by Schilde et al. (2009). They develop a solution approach for a multi-objective variant of the OP, but at the same time their approach outperforms the five-step heuristic of Chao et al. (1996b), which deals with single objective OP instances. They developed a Pareto ant colony optimisation algorithm and a multi-objective variable neighbourhood search algorithm, both hybridised with path relinking.

3. Team orienteering problem

3.1. Problem definition and mathematical formulation

The Team Orienteering Problem (TOP) (Chao et al., 1996a; Tang and Miller-Hooks, 2005) or multiple tour maximum collection problem (MTMCP) (Butt and Cavalier, 1994) is an OP where the goal is to determine P paths, each limited by T_{max}, that maximises the total collected score. The TOP corresponds to playing the game of orienteering by teams of several persons, each collecting scores during the same time span.

The TOP can be formulated as an integer problem with these decision variables: $x_{ijp} = 1$ if, in path p, a visit to vertex i is followed by a visit to vertex $j - 0$ otherwise; $y_{ip} = 1$ if vertex i is visited in path $p - 0$ otherwise; u_{ip} = the position of vertex i in path p.

$$Max \sum_{p=1}^{P} \sum_{i=2}^{N-1} S_i y_{ip}, \tag{7}$$

$$\sum_{p=1}^{P} \sum_{j=2}^{N} x_{1jp} = \sum_{p=1}^{P} \sum_{i=1}^{N-1} x_{iNp} = P, \tag{8}$$

$$\sum_{p=1}^{P} y_{kp} \leqslant 1; \quad \forall k = 2, \ldots, N-1, \tag{9}$$

$$\sum_{i=1}^{N-1} x_{ikp} = \sum_{j=2}^{N} x_{kjp} = y_{kp}; \quad \forall k = 2, \ldots, N-1; \quad \forall p = 1, \ldots, P, \tag{10}$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} t_{ij} x_{ijp} \leqslant T_{max}; \quad \forall p = 1, \ldots, P, \tag{11}$$

$$2 \leqslant u_{ip} \leqslant N; \quad \forall i = 2, \ldots, N; \quad \forall p = 1, \ldots, P, \tag{12}$$

$$u_{ip} - u_{jp} + 1 \leqslant (N-1)(1 - x_{ijp}); \quad \forall i, j = 2, \ldots, N; \quad \forall p = 1, \ldots, P, \tag{13}$$

$$x_{ijp}, y_{ip} \in \{0, 1\}; \quad \forall i, j = 1, \ldots, N; \quad \forall p = 1, \ldots, P. \tag{14}$$

The objective function (7) is to maximise the total collected score. Constraints (8) guarantee that each path starts in vertex *1* and ends in vertex *N*. Constraints (9) ensure that every vertex is visited at most once. Constraints (10) guarantee the connectivity of each path. Constraints (11) ensure the limited time budget for each path. Constraints (12) and (13) are necessary to prevent subtours. Other mathematical formulations of the TOP are presented in Butt and Cavalier (1994), Tang and Miller-Hooks (2005) and Boussier et al. (2007).

3.2. Practical applications

Butt and Cavalier (1994) describe a TOP application of athlete recruitment from high schools. A recruiter has to visit several schools in a given number of days. He can first assign a score to each school, based on its recruiting potential. As the recruiter's available time is limited, he has to choose the schools to visit each day and try to maximise the recruiting potential.

Tang and Miller-Hooks (2005) describe a TOP application of routing technicians to service customers. Each TOP path represents a single technician who can only work a limited number of hours in a day. Thus, not all customers requiring service can be included in the technicians' daily schedules. A subset of customers will have to be selected, taking into account customer importance and task urgency.

3.3. Benchmark instances

Team orienteering problem benchmark instances are available from Chao (1993) and Chao et al. (1996a). The characteristics of these problems are presented in Table 2. The number of test instances is given, together with the number of vertices (N) and the number of paths (P). In total, 387 TOP benchmark instances are available, the number of vertices varies between 21 and 102 and the instances have two, three or four paths.

All these benchmark instances are available via <www.mech.kuleuven.be/cib/op>.

Table 2
Benchmark TOP instances.

Reference	Number of test instances	Number of vertices (N)	Number of paths (P)
Chao (1993) and Chao et al. (1996a)	3*18	32	2, 3, 4
	3*11	21	2, 3, 4
	3*20	33	2, 3, 4
	3*20	100	2, 3, 4
	3*26	66	2, 3, 4
	3*14	64	2, 3, 4
	3*20	102	2, 3, 4

3.4. Solution approaches

An exact algorithm to solve the TOP, using column generation, has been published by Butt and Ryan (1999). They were able to solve problems with up to 100 vertices, when the number of vertices in each path remains small.

More recently, Boussier et al. (2007) present an exact method to deal with TOPs and TOPs with time windows. They start with column generation and couple this with branch-and-bound to obtain a branch-and-price scheme. In order to increase the performance, different acceleration techniques are applied: "limited discrepancy search", a heuristic tree-search method from constraint programming (Harvey and Ginsberg, 1995), and "label loading" and "meta extensions", two pre-processing procedures that can be interpreted as priority rules. Problems with 100 vertices where few vertices can be selected (until around 15 per path), are solved in less than two hours of computation time.

The first published heuristic for the TOP is developed by Chao et al. (1996a) and is more or less the same as their five-step heuristic for the OP. Instead of only selecting the best path, the P best paths are selected and two reinitialisation steps are used instead of one. Tang and Miller-Hooks (2005) develop a tabu search heuristic (TMH) embedded in an Adaptive Memory Procedure (AMP). In the tabu search initialisation step, the parameters are set to explore only a small number of neighbourhood solutions. In the improvement step, random and greedy procedures generate feasible and infeasible neighbourhood solutions based on the current parameters. The parameters guarantee continuous switching between small and large neighbourhood structures. In the evaluation step, the best non-tabu solution is selected and the parameters are adjusted based on the current neighbourhood size and the solution quality. The AMP works in a similar way to genetic algorithms, and offspring can be generated from more than two parents. Single paths are stored and combined to form an initial solution for the tabu search. The result of the tabu search updates the single paths.

More recent metaheuristics for the TOP are described in Archetti et al. (2007), Ke et al. (2008), Vansteenwegen et al. (2009b,c) and Souffriau et al. (in press). Archetti et al. (2007) develop two variants of a tabu search heuristic and a Slow and Fast Variable Neighbourhood Search (SVN and FVN). All four metaheuristics start from an incumbent solution s by making a jump (described below) to an intermediate solution s'. Then, tabu search is used to try to improve s'. The new solution s" is compared with s. In the VNS strategy, s" is only accepted if it has a higher score than s. In the tabu search strategy, s" is always accepted. This process is repeated until a stopping criterion is met. As in the five-step heuristic of Chao et al., non-included vertices are also organised in paths. The tabu search uses two moves, *1-move* moves one vertex from one path to another, and the *swap-move* switches two vertices between paths. Vertices are always included using cheapest insertion. Since infeasible solutions are also accepted throughout the algorithm, different procedures are developed to reduce or remove the infeasibility of a path. Two kinds of jumps are used in these algorithms, the first one is a series of *1-moves* with non-included vertices, the second one swaps two sets of vertices between the selected paths and the paths with non-included vertices. Both tabu search algorithms only use the second jump. One tabu search algorithm only uses feasible solutions (TSF) while the other one also accepts unfeasible solutions (TSU). The VNS also uses tabu search as local search, but with far less iterations, and only considers feasible solutions. Every time the incumbent solution is improved, 2-Opt (Lin, 1965) is used to reduce the path duration. In order to compare the quality of different solutions, five functions are used, based on score, duration and feasibility. Archetti et al. present the best results among the above-mentioned algorithms for the (T)OP. The

strength of the algorithm probably lies in the fact that all non-included vertices are also grouped in feasible paths.

Ke et al. (2008) implement an Ant Colony Optimisation (ACO) approach to solve TOP instances. At each cycle, each ant constructs a feasible solution, that is improved by a local search procedure. Subsequently, pheromone trails are updated. The algorithm stops iterating when a maximum number of cycles has been performed. The local search procedure consists of shortening each path by using 2-Opt and then inserting as many vertices as possible. This procedure is iterated until a local optimum is reached. Four different methods are proposed to construct feasible solutions in the ACO framework, resulting in four different algorithms. In the sequential method (ASe) complete paths are constructed one after another. In the random-concurrent method (ARC), a path is randomly selected at every iteration to add a new vertex. In the deterministic-concurrent method (ADC), the sequence of paths to consider is fixed. In the simultaneous method (ASi), at every iteration, a vertex is added to one of the paths until all paths reach their limit length. It appears that the sequential method is an excellent compromise between solution quality and computational time. The quality of the sequential method results is at least as good as the results obtained by Archetti et al. (2007) and clearly faster.

Vansteenwegen et al. (2009b,c) were the first to focus on obtaining good TOP solutions in only a few seconds of computational time. They first implemented a Guided Local Search (GLS) framework (Vansteenwegen et al., 2009b) and later a Skewed Variable Neighbourhood Search (SVNS) framework (Vansteenwegen et al., 2009c). Both algorithms apply a combination of intensification and diversification procedures. Two diversification procedures simply remove a chain of attractions in each path. Another procedure tries to gather the available budget spread over different paths within the current solution, into a single path in the new solution. Two types of intensification procedures are designed. The first type tries to increase the score and the second type tries to decrease the travel time in a path. The intensification procedures are described in more detail below. The SVNS algorithm clearly outperforms the GLS algorithm. Furthermore, the computation time of the SVNS algorithm is lower. The success of the SVNS algorithm can be explained by a combination of factors. First of all, the SVNS framework appears suitable for this type of problem. Accepting a slightly worse intermediate solution when it is far from the incumbent, is a good strategy for selecting the vertices that will be part of the optimal solution. Additionally, the importance of a good diversification strategy is experimentally demonstrated and certain moves appear to be essential. The most important conclusion, however, is that it will always be the specific combination and sequence of different moves that determine the final quality of the algorithm.

Souffriau et al. (in press) designed two variants of a Greedy Randomised Adaptive Search Procedure (GRASP) with Path Relinking. Modifying only the stopping criterion allows changing between a slow version (SPR), with excellent results, and a fast version (FPR), with 'just' high quality results. In this GRASP algorithm, four procedures are executed in sequence, until no further improvements are identified during a fixed number of iterations. First, a construction procedure generates an initial solution. Based on a ratio between greediness and randomness, vertices are inserted one by one until all paths are full. Due to the randomness, a new initial solution is obtained during every iteration. Then, the initial solution is improved using local search neighbourhoods. The local search procedure alternates between reducing the total time of the solution and increasing its total score, until the solution is locally optimal with respect to all the neighbourhoods. Next, the path relinking introduces a pool of elite solutions as a long-term memory component. Furthermore, it considers the solutions on a virtual path in the search space between the local search solution and each elite solution. The best solution found on these paths is returned. Finally, the pool of elite solutions is updated. The algorithm keeps track of the best solution found during all iterations. The quality of the results of the slow variant is comparable to the quality obtained by the best algorithms of Archetti et al. (2007) and Ke et al. (2008).

Table 3 summarises the performance of the best TOP algorithms. This comparison is based on 157 benchmark instances from Chao et al. (1996a). Data sets 4–7 are used and instances for which the same result is obtained by all algorithms are excluded, resulting in 157 relevant instances. For each algorithm, the table presents the number of times the best known solution is found and the average gap to the best known solution. For each algorithm, only the best result (over different runs) is taken into account.

The last column presents, for each algorithm, the average CPU time in seconds. Since not all details are published on the execution times of the different approaches, the CPU times in Table 3 are based on averages reported by the authors. For the ant colony approaches of Ke et al. (2008) and the GRASP algorithm of Souffriau et al. (in press) the average execution times of one run are multiplied by ten, which equals the number of runs it took to find the best solution. Similarly, the average execution times reported by

Table 3
Summary of the best-performing TOP algorithms.

Reference	Computer specifications	Technique	Algorithm	# best	Avg gap (%)	Avg CPU (seconds)
Tang and Miller-Hooks (2005)	DEC Alpha XP1000, 1 GB RAM, 1.5 GB swap	Tabu search	TMH	34	1.32	336.6
Archetti et al. (2007)	Intel Pentium 4, 1 GB RAM, 2.8 GHz	Tabu search	TSF	94	0.20	531.5
			TSU	69	0.49	318.0
		Variable neighbourhood search	SVN	**128**	**0.05**	906.1
			FVN	97	0.18	**63.6**
Ke et al. (2008)	PC, 3.0 GHz	Ant colony optimisation	ASe	**130**	**0.08**	252.3
			ARC	81	0.40	204.8
			ADC	80	0.35	213.8
			ASi	84	0.32	215.0
Vansteenwegen et al. (2009c)	Intel Pentium 4, 1 GB RAM, 2.8 GHz	Variable neighbourhood search	SVNS	44	0.97	**3.8**
Souffriau et al. (in press)	Intel Xeon, 4 GB RAM, 2.5 GHz	Greedy randomised adaptive search procedure with path relinking	FPR	78	0.39	**5.0**
			SPR	**131**	**0.04**	212.4

Archetti et al. (2007) are multiplied by three. The three best results for each performance criteria are indicated in bold.

In order to give some guidance in the development of future algorithms, it is interesting to take a closer look at the commonly used local search moves. First, five moves are explained that increase the total score of the solution: *Insert*, *TwoInsert*, *Replace*, *TwoReplace* and *Change*. Next, two moves are described that reduce the travel time between the selected vertices: *2-Opt* and *Swap*. Efficient TOP algorithms switch between score increasing moves and travel time decreasing moves.

1. The *Insert* move tries to include one vertex extra in any of the paths, using cheapest insertion.
2. The *TwoInsert* move tries to include two extra vertices. For each combination of two non-included vertices, the least time consuming position for each vertex is considered.
3. For the *Replace* move, all non-included vertices are considered for insertion. If enough time budget is available for insertion, the vertex is inserted. If the remaining budget is insufficient, an included vertex with a lower score is excluded to make the insertion feasible.
4. The *TwoReplace* move considers all combinations of two non-included vertices for insertion. If necessary, two successive included vertices are considered for exclusion to make the insertion feasible.
5. The *Change* move generates neighbours in an opposite way to *Replace* and *TwoReplace*. Change first removes five successive vertices in a path and then tries to insert non-included vertices one by one, until no more vertices can be inserted. Only if the removal and the insertion result in a higher total score, the new solution is accepted as a neighbour.
6. A very popular move to reduce the travel time between two vertices is *2-Opt* (Lin, 1965). *2-Opt* removes two edges from the path to replace them with two new edges not previously included in the path. The path has to remain closed and the total travel time should be reduced.
7. The *Swap* move exchanges two vertices belonging to different paths in order to save travel time.

Table 4 indicates which heuristic applies which local search moves. The local search moves of Archetti et al. (2007) are not exactly the same as the moves defined above, since Archetti et al. arrange the non-included vertices in paths as well. However, the results of the moves are similar: the *1-move* corresponds with *Insert*, and the *swap-move* corresponds with *Swap*.

All algorithms use *Insert* to increase the score of a solution. Tang and Miller-Hooks (2005) are the only ones not applying *2-Opt* and Ke et al. (2007) are the only ones not using *Swap*. It would be interesting to measure the efficiency of these local search moves and to determine to what extend each move contributes to obtain a high quality solution in limited computational time. However, comparing Tables 3 and 4 does not allow drawing relevant conclusions. Until now, almost no research has been done on the efficiency of local search moves; only Vansteenwegen et al. (2009a) conducted a first attempt.

4. Orienteering problem with time windows

4.1. Problem definition and mathematical formulation

Recently, the Orienteering Problem with Time Windows (OPTW) and the Team Orienteering Problem with Time Windows (TOPTW), discussed in the next section, received a lot of attention (Boussier et al., 2007; Righini and Salani, 2009; Montemanni and Gambardella, 2009; Vansteenwegen et al., 2009d; Tricoire et al., 2010). The main reason is that instances with time windows should be solved in a very different way than instances without time windows. For instance, the well-known *2-Opt* move is indispensable to obtain high quality results for the OP, but due to the time windows, it cannot be applied to efficiently solve the OPTW. Moreover, reducing the travel time by changing the order of the visits, is no longer appropriate due to the time windows. However, solution approaches for the (T)OPTW can be applied to deal with the (T)OP as well. This is shown explicitly by Tricoire et al. (2010), who slightly modify their solution approach for problems with time windows in order to obtain good results on (T)OP benchmark instances. The work of Tricoire et al. (2010) is discussed in more detail in the TOPTW section.

In the OPTW, each vertex is assigned a time window $[O_i, C_i]$ and a visit to a vertex can only start during this time window. Based on the notation introduced earlier, the OPTW can be formulated as a mixed integer problem with the following decision variables: $x_{ij} = 1$ if a visit to vertex i is followed by a visit to vertex j – 0 otherwise; $y_i = 1$ if vertex i is visited – 0 otherwise; $s_i =$ the start of the service at vertex i; M a large constant.

$$Max \sum_{i=2}^{N-1} \sum_{j=2}^{N} S_i x_{ij}, \tag{15}$$

$$\sum_{j=2}^{N} x_{1j} = \sum_{i=1}^{N-1} x_{iN} = 1, \tag{16}$$

$$\sum_{i=1}^{N-1} x_{ik} = \sum_{j=2}^{N} x_{kj} \leqslant 1; \quad \forall k = 2, \ldots, N-1, \tag{17}$$

$$s_i + t_{ij} - s_j \leqslant M(1 - x_{ij}); \quad \forall i,j = 1, \ldots, N, \tag{18}$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} t_{ij} x_{ij} \leqslant T_{max}, \tag{19}$$

$$O_i \leqslant s_i; \quad \forall i = 1, \ldots, N, \tag{20}$$

$$s_i \leqslant C_i; \quad \forall i = 1, \ldots, N, \tag{21}$$

$$x_{ij} \in \{0, 1\}; \quad \forall i,j = 1, \ldots, N. \tag{22}$$

The objective function (15) maximises the total collected score. Constraints (16) guarantee that the path starts in vertex *1* and ends in vertex *N*. Constraints (17) determine the connectivity and ensure that every vertex is visited at most once. Constraints (18) ensure the timeline of the path and constraint (19) limits the time budget. Constraints (20) and (21) restrict the start of the service to the time window. The time window of the end vertex *N* may replace constraint (19).

Table 4
Local search moves in TOP heuristics.

	Insert	*TwoInsert*	*Replace*	*TwoReplace*	*Change*	*2-Opt*	*Swap*
Tang and Miller-Hooks (2005)	✓				✓		✓
Archetti et al. (2007)	✓					✓	✓
Ke et al. (2008)	✓					✓	
Vansteenwegen et al. (2009b)	✓		✓			✓	✓
Vansteenwegen et al. (2009c)	✓	✓	✓	✓	✓	✓	✓
Souffriau et al. (in press)	✓		✓			✓	✓

Not many specific OPTW applications are mentioned in the literature, but most of the (T)OP applications mentioned before also require time windows and, as a consequence, the (T)OPTW model can be applied to many real life situations. For example, the planning problems for tourist applications will have to take the opening hours of the tourist attractions into account. The routing of technicians or fuel delivery problems will also face opening hours in practice.

4.2. Benchmark instances

Table 5 presents an overview of the available OPTW test instances. For every set of instances, the corresponding reference is given, together with the name of the original instances the set is based on. The number of instances and the number of vertices (N) of the instances are presented as well.

Righini and Salani (2006) designed 68 test instances for the OPTW using Solomon's data set (1987) of vehicle routing problems with time windows (c10*, r10* and rc10*) and using 10 multi-depot vehicle routing problems of Cordeau et al. (pr1–pr10) (1997). Montemanni and Gambardella (2009) added 27 extra instances based on Solomon (c20*, r20* and rc20*) and 10 instances based on Cordeau et al. (pr11–pr20). In total, 105 OPTW instances are available and the number of vertices varies between 48 and 288.

All these benchmark instances are available via <www.mech.kuleuven.be/cib/op>.

4.3. Solution approaches

Kantor and Rosenwein (1992) were the first to solve the OPTW. They first describe a straightforward insertion heuristic. The vertex with the highest ratio "score over insertion time" is inserted into the path, without violating time windows. Secondly, a depth-first search algorithm is proposed that constructs partial paths, using the insertion heuristic and beginning in the start vertex. Partial paths are abandoned if they are infeasible or if they are unlikely to yield the best total score. Righini and Salani (2006, 2009) designed an exact algorithm, bi-directional dynamic programming, to solve OPTW instances to optimality. Starting forward from the start vertex and backwards from the end vertex, current states are extended by adding an extra vertex at the end. Forward and backward states are matched if feasible and dominance tests are applied to record only non-dominated states. Decremental state space relaxation (Righini and Salani, 2008) is used to reduce the number of states to be explored. Mansini et al. (2006) develop a simple constructive heuristic and a granular variable neighbourhood search for a special case of the OPTW in which the starting and end vertex are the same. The granular VNS improves a VNS algorithm by reducing the size of the analysed neighbourhoods by preventing the insertion of non-promising arcs. Bar-Yehuda et al. (2005) also mention the OPTW, but only for the special cases where all vertices have the same score and are on a line or in the Euclidean plane. They do not consider time limits on the path duration.

Table 5
Benchmark OPTW instances.

Reference	Based on	Number of test instances	Number of vertices (N)
Righini and Salani (2006)	Solomon (c10*, r10* and rc10*)	29	50
Righini and Salani (2008)	Solomon (c10*, r10* and rc10*)	29	100
	Cordeau et al. (pr1-pr10)	10	48–288
Montemanni and Gambardella (2009)	Solomon (c20*, r20* and rc20*)	27	100
	Cordeau et al. (pr11-pr20)	10	48–288

5. Team orienteering problem with time windows

5.1. Problem definition and mathematical formulation

Based on the notation introduced earlier, the TOPTW can be formulated as a mixed integer problem with the following decision variables: $x_{ijp} = 1$ if, in path p, a visit to vertex i is followed by a visit to vertex j – 0 otherwise; $y_{ip} = 1$ if vertex i is visited in path p – 0 otherwise; s_{ip} = the start of the service at vertex i in path p; M a large constant.

$$Max \sum_{p=1}^{P} \sum_{i=2}^{N-1} S_i y_{ip}, \tag{23}$$

$$\sum_{p=1}^{P} \sum_{j=2}^{N} x_{1jp} = \sum_{p=1}^{P} \sum_{i=1}^{N-1} x_{iNp} = P, \tag{24}$$

$$\sum_{i=1}^{N-1} x_{ikp} = \sum_{j=2}^{N} x_{kjp} = y_{kp}; \quad \forall k = 2, \ldots, N-1; \quad \forall p = 1, \ldots, P, \tag{25}$$

$$s_{ip} + t_{ij} - s_{jp} \leqslant M(1 - x_{ijp}); \quad \forall i, j = 1, \ldots, N; \quad \forall p = 1, \ldots, P, \tag{26}$$

$$\sum_{p=1}^{P} y_{kp} \leqslant 1; \quad \forall k = 2, \ldots, N-1, \tag{27}$$

$$\sum_{i=1}^{N-1} \sum_{j=2}^{N} t_{ij} x_{ijp} \leqslant T_{max}; \quad \forall p = 1, \ldots, P, \tag{28}$$

$$O_i \leqslant s_{ip}; \quad \forall i = 1, \ldots, N; \quad \forall p = 1, \ldots, P, \tag{29}$$

$$s_{ip} \leqslant C_i; \quad \forall i = 1, \ldots, N; \quad \forall p = 1, \ldots, P, \tag{30}$$

$$x_{ijp}, y_{ip} \in \{0, 1\}; \quad \forall i, j = 1, \ldots, N; \quad \forall p = 1, \ldots p. \tag{31}$$

The objective function (23) maximises the total collected score. Constraints (24) guarantee that all paths start in vertex *1* and end in vertex *N*. Constraints (25) and (26) determine the connectivity and timeline of each path. Constraints (27) ensure that every vertex is visited at most once and constraints (28) limit the time budget. Constraints (29) and (30) restrict the start of the service to the time window. Again, the time window of the end vertex N can replace constraints (28).

5.2. Benchmark instances

Table 6 presents an overview of the available TOPTW test instances. For every set of instances, the corresponding reference is given, together with the name of the original instances the set is based on. The number of instances, the number of vertices (N) and the number of paths (P) are presented as well.

Montemanni and Gambardella (2009) designed new TOPTW instances based on their own OPTW instances and the OPTW instances from Righini and Salani (2006). Instead of using only one path, all aforementioned OPTW instances are also considered with two, three and four paths. Vansteenwegen et al. (2009d) constructed a new data set of TOPTW instances, with more difficult instances, but nonetheless, for all these instances the optimal solution is known. Indeed, this data set uses the original instances from Solomon (1987) and Cordeau et al. (1997), with the number of paths equal to the number of vehicles. With this number of paths it should be feasible to visit every vertex and hence the optimal result equals the sum of all scores. In total, 144 TOPTW instances are available, the number of vertices varies between 48 and 288 and the number of paths varies between 2 and 20.

Table 6
Benchmark TOPTW instances.

Reference	Based on	Number of test instances	Number of vertices (N)	Number of paths (P)
Montemanni and Gambardella (2009)	Solomon (c10*, r10* and rc10*)	29	50	2, 3, 4
	Solomon (c10*, r10* and rc10*)	29	100	2, 3, 4
	Cordeau et al. (pr1–pr10)	10	48–288	2, 3, 4
	Solomon (c20*, r20* and rc20*)	27	100	2, 3, 4
	Cordeau et al. (pr11–pr20)	10	48–288	2, 3, 4
Vansteenwegen et al. (2009)	Solomon (c10*, r10* and rc10*)	29	100	9–19
	Cordeau et al. (pr1-pr10)	10	48–288	3–20

All these benchmark instances are available via www.mech.kuleuven.be/cib/op.

5.3. Solution approaches

Montemanni and Gambardella (2009) based their algorithm to solve TOPTW instances on ant colony optimisation. The method is based on the solution of a hierarchic generalisation of the TOPTW. This algorithm clearly outperforms the algorithm of Mansini et al. (2006) on all considered OPTW instances. Tricoire et al. (2010) solve the Multi-Period Orienteering Problem with Multiple Time Windows, a generalisation of the TOPTW. In this problem, every vertex can have more than one time window on a given day and time windows can be different on different days. They propose a Variable Neighbourhood Search (VNS) algorithm and embed an exact algorithm to deal with a path feasibility subproblem. Extensive experimental results show that they obtain high quality solutions for problems with 100 vertices and 2 paths in around one minute of computation time. The application described by Tricoire et al. (2010) is about facilitating the planning of future working days by field workers and sales representatives. Vansteenwegen et al. (2009d) designed a very fast Iterated Local Search (ILS) metaheuristic to deal with TOPTW instances. This algorithm tackles the above-mentioned tourist trip design problem (Vansteenwegen and Van Oudheusden, 2007). In this case, personalised tourist trips are planned, taking into account opening hours. This application requires a TOPTW solution within a few seconds of computation. The quality of their results is worse (around 2% on average) than the results of Montemanni and Gambardella (2009) on the same test sets, but the computational time is reduced significantly, with a factor of more than 100, to around one second.

Vansteenwegen et al. (2009d) did their experiments on a PC Intel Core 2 with 2.5 GHz processor and 3.45 GB RAM. Montemanni and Gambardella (2009) used a comparable computer with a Dual AMD Opteron 250 2.4 GHz processor with 4 GB RAM. Tricoire et al. (2010) also used a comparable computer with 2.4 GHz and 4 GB RAM. It is impossible to give a detailed comparison of the TOPTW solution approaches, since different authors have used (slightly) different benchmark instances. Nevertheless, it can be concluded that the ILS approach of Vansteenwegen et al. (2009d) has the advantage of being very quick and the approaches of Montemanni and Gambardella (2009) and Tricoire et al. (2010) have the advantage of obtaining high quality solutions.

The Selective Vehicle Routing Problem with Time Windows (SVRPTW) generalises the TOPTW by adding two constraints to the TOPTW. The first constraint limits the vehicle capacity, while each customer has a certain demand. The second constraint limits the travel distance of the paths. It should be noted that imposing this limit on the travel distance, is not always the same as imposing the time limit T_{max}. In most practical cases, visiting a vertex will require some visiting (or service) time. If that is the case, the time limit T_{max} will not only limit the travel time, but also the visiting time. The visiting time is not considered when the travel distance is limited. The SVRPTW corresponds to a VRPTW in which not all customers can be visited for some logistic reason. Boussier et al. (2007) easily modified their exact branch-and-price algorithm for the TOP to solve SVRPTW instances with 100 vertices and up to 10 paths.

6. Variants of the orienteering problem

The OP can be formulated as a TSP With Profits (TSPWP) (Feillet et al., 2005a; Bérubé et al., 2009) or as a special case of the Resource Constrained TSP (RCTSP) (Pekny and Miller, 1990). A TSPWP can be seen as a bicriteria TSP with two opposite objectives: collecting profits by travelling around and minimising travel costs. When the travel cost objective is stated as a constraint, this problem corresponds to an OP. A TSPWP classification and an extensive literature survey about TSPWP applications, modelling approaches and solution techniques is presented by Feillet et al. (2005a). In the RCTSP a set of vertices is given and travelling from one vertex to another incurs a cost and the consumption of a resource. The objective is to find a minimum cost path to visit all vertices, while the resource consumption is constrained by a given value. This problem can be converted to an orienteering problem, if not all vertices need to be visited, the travelling cost between two vertices is replaced by a (negative) vertex score and the travelling cost is considered as the resource consumption.

Other variants of the OP are the Generalised Orienteering Problem (GOP) (Wang et al., 1996; Zong et al., 2005; Wang et al., 2008), the multi-objective OP (Schilde et al., 2009), the OP with stochastic profits (Ilhan et al., 2008), the OP with compulsory vertices (Gendreau et al., 1998b), the time-dependent OP (Fomin and Lingas, 2002) and the capacitated TOP (Archetti et al., 2009). The difference between the OP and the generalised OP is the objective function. In the OP, the scores associated with each vertex are added to obtain the total score, while in the GOP, the total score is a more complicated (nonlinear) function of the vertices visited. In this case a certain combination of vertices could produce a higher (or lower) score than the sum of individual scores. An example from the tourist sector could be that attractions are variations on a certain theme and in order to really appreciate the series, they should preferably be visited all. Another possibility is that a low valued attraction becomes more appealing when visited in combination with another one. For instance, a craft market could be more appreciated after a visit to a related folk museum. Schilde et al. (2009) model the different interests a tourist will have in different categories of attractions, as a multi-objective orienteering problem. Furthermore, they present a solution approach to deal with the bi-objective orienteering problem.

In the Orienteering Problem with Stochastic Profits (OPSP), normally distributed scores are associated with the vertices. The goal of the OPSP is to maximise, within a time limit, the probability of collecting more than a prespecified target level (Ilhan et al., 2008). Ilhan et al. (2008) developed an exact solution approach and a bi-objective genetic algorithm to deal with the OPSP. They also describe an application at a major US car company.

Compulsory vertices for the OP were first mentioned by Gendreau et al. (1998b). In the OP with compulsory vertices, a subset of all vertices has to be visited. In the case of a company that uses the OP to make the daily planning, these compulsory vertex could be customers that have to be scheduled today and cannot be postponed. In tourist trip planning, these compulsory vertices could be top attractions that should be included in all personalised tours. Gendreau et al. developed different classes of valid inequalities and use branch-and-cut to solve to optimality problems with up to 100 vertices, of which some are compulsory.

Fomin and Lingas (2002) consider a generalisation of the OP, the time-dependent OP. In this case, the travel time between two vertices depends on the leaving time of the first vertex. One application they discuss is about a robot that needs to intercept as many moving targets as possible, within a given time. At any time moment, the location of each target is known and thus also the (time-dependent) travel times.

Archetti et al. (2009) introduce the capacitated team orienteering problem. Additional to the above defined regular TOP, a non-negative demand is associated with each vertex and the total demand in each path may not exceed the given capacity. In many real-life applications, the capacity of each vehicle is an issue to take into account.

A few papers consider arc routing variants of the OP where a profit is associated to each arc. Gendreau et al. (1995) present the Ring Network Design Problem (RNDP) with an application in telecommunications. Feillet et al. (2005b) proposed a branch-and-price approach for the "Profitable Arc Tour Problem" (PATP). The application they consider is a freight transportation planning problem in the car industry. Aráoz et al. (2009) present a similar problem, named the "Prize-Collecting Rural Postman Problem" (PCRPP). They present the application of collecting recycling bins. When this collection is organised by the public administration, all arcs in an area must be serviced. A private company, however, would only select to serve the streets with the highest profits.

In the three above-mentioned problems, the objective is to maximise the difference between the collected profit and the total travel time. This objective is the arc routing objective of the TSP with profits and it should be noted that in the regular OP, the travel time is not a part of the objective, but it is limited by a constraint. In the RNDP and the PCRPP the profit of each arc can be collected at most once. In the PATP, the profit of each arc can be collected at most a fixed number of times.

Archetti et al. (in press) describe the "undirected capacitated arc routing problem with profits" which corresponds to a capacitated arc TOP. The objective is to determine a path for each available vehicle in order to maximise the total collected profit, without violating the capacity and time limit of each vehicle. They consider an application where carriers can select potential customers for transporting their goods. Another potential application is the creation of personal bicycle trips. Based on the biker's personal interests, starting and ending point and the available time, a personal trip can be composed using the selection of arcs that match the most with the cyclist's profile.

The combination of the well-known Vehicle Routing Problem (VRP) and Arc Routing Problem (ARP) is defined as the "general routing problem" (Muyldermans et al., 2005). However, the name Generalised Orienteering Problem is already assigned (Wang et al., 1996; Zong et al., 2005; Wang et al., 2008). Therefore, we propose here to name the combination of the orienteering problem and the arc routing problem with profits, the "Mixed Orienteering Problem" (MOP), when scores are associated to vertices as well as to arcs. For the above-mentioned tourist applications, the MOP needs to be solved when not all attractions are at specific locations, but when a walk along a river or through a beautiful street can also be considered as an attraction.

7. Conclusions and possible future research lines

A number of challenging practical applications can be modelled as an orienteering problem or a variant of the orienteering problem. The routing of technicians, athlete recruitment or military applications can benefit from (team) orienteering algorithms. Tourism applications, for instance, require very fast and effective solution approaches. It can be expected that the orienteering problem will play a prominent role in future developments in tourist planning problems.

The performance of many metaheuristic approaches for (T)OP instances is compared. Three algorithms obtain the same result quality and it appears to be very difficult to obtain a higher quality on the same set of instances. New and larger (T)OP test instances should be developed to be able to better distinguish the performance of different and future approaches. Probably some existing approaches will be too time consuming for larger instances and the quality of other approaches will significantly decrease when instances grow.

Based on the best performing algorithms, new algorithms should probably group non-included vertices in feasible paths and allow infeasible solutions during the search procedure. Additionally, more research effort should be dedicated towards the efficiency of different local search moves in order to facilitate the future development of new solution approaches for the orienteering problem.

Until now, not much attention is given to the arc routing problem with profits, which is the "arc routing" equivalent of the orienteering problem. This line of research can benefit from the recent developments in the field of the orienteering problem. Furthermore, combining the orienteering problem and the arc routing problem with profits leads to a new problem, defined here as the mixed orienteering problem. To the best of our knowledge, no research has been done about the mixed orienteering problem. This new problem offers many research opportunities. Furthermore, many practical applications, for instance in tourism, will benefit from solution algorithms for these more complicated problems.

Almost all OP papers assume uncapacitated vehicles; only a few papers deal with capacitated vehicles (Boussier et al., 2007; Archetti et al., 2009, in press). Since in many practical cases capacity constraints are present, this is certainly a relevant topic for further research.

Acknowledgement

Dr. P. Vansteenwegen is a post-doctoral research fellow of the "Fonds Wetenschappelijk Onderzoek – Vlaanderen (FWO)".

References

Aráoz, J., Fernández, E., Meza, O., 2009. Solving the prize-collecting rural postman problem. European Journal of Operational Research 196, 886–896.

Archetti, C., Feillet, D., Hertz, A., Speranza, M., 2009. The capacitated team orienteering and profitable tour problems. Journal of the Operational Research Society 60, 831–842.

Archetti, C., Feillet, D., Hertz, A., Speranza, M. in press. The undirected capacitated arc routing problem with profits. Computers and Operations Research. doi: 10.1016/j.cor.2009.05.005.

Archetti, C., Hertz, A., Speranza, M., 2007. Metaheuristics for the team orienteering problem. Journal of Heuristics 13, 49–76.

Arkin, E., Mitchell, J., Narasimhan, G. 1998. Resource-constrained geometric network optimisation. In: Proc. 14th ACM Symposium on Computational Geometry, June, pp. 307–316.

Bar-Yehuda, R., Even, G., Shahar, S., 2005. On approximating a geometric prize-collecting travelling salesman problem with time windows. Journal of Algorithms 55 (1), 76–92.

Bérubé, J.F., Gendreau, M., Potvin, J.Y., 2009. An exact ε-constraint method for bi-objective combinatorial optimization problems: Application to the traveling salesman problem with profits. European Journal of Operational Research 194, 39–50.

Boussier, S., Feillet, D., Gendreau, M., 2007. An exact algorithm for the team orienteering problem. 4OR 5, 211–230.
Butt, S., Cavalier, T., 1994. A heuristic for the multiple tour maximum collection problem. Computers and Operations Research 21, 101–111.
Butt, S., Ryan, D., 1999. An optimal solution procedure for the multiple tour maximum collection problem using column generation. Computers and Operations Research 26, 427–441.
Chao, I. 1993. Algorithms and solutions to multi-level vehicle routing problems. Ph.D. Dissertation, Applied Mathematics Program, University of Maryland, College Park, USA.
Chao, I., Golden, B., Wasil, E., 1996a. Theory and methodology – the team orienteering problem. European Journal of Operational Research 88, 464–474.
Chao, I., Golden, B., Wasil, E., 1996b. Theory and methodology – a fast and effective heuristic for the orienteering problem. European Journal of Operational Research 88, 475–489.
Cordeau, J.-F., Gendreau, M., Laporte, G., 1997. A tabu search heuristic for periodic and multi-depot vehicle routing problems. Networks 30, 105–119.
Feillet, D., Dejax, P., Gendreau, M., 2005a. Travelling salesman problems with profits. Transportation Science 39, 188–205.
Feillet, D., Dejax, P., Gendreau, M., 2005b. The profitable arc tour problem: Solution with a branch-and-price algorithm. Transportation Science 39, 539–552.
Fischetti, M., Salazar, J., Toth, P., 1998. Solving the orienteering problem through branch-and-cut. INFORMS Journal on Computing 10, 133–148.
Fomin, F., Lingas, A., 2002. Approximation algorithms for time-dependent orienteering. Information Processing Letters 83, 57–62.
Gendreau, M., Laporte, G., Semet, F. 1995. A branch-and-cut algorithm for the undirected selective travelling salesman problem. Publication CRT-95-80, Centre de recherche sur les transports, Montreal, Canada.
Gendreau, M., Laporte, G., Semet, F., 1998a. A tabu search heuristic for the undirected selective travelling salesman problem. European Journal of Operational Research 106, 539–545.
Gendreau, M., Laporte, G., Semet, F., 1998b. A branch-and-cut algorithm for the undirected selective travelling salesman problem. Networks 32, 263–273.
Golden, B., Levy, L., Vohra, R., 1987. The orienteering problem. Naval Research Logistics 34, 307–318.
Golden, B., Wang, Q., Liu, L., 1988. A multifaceted heuristic for the orienteering problem. Naval Research Logistics 35, 359–366.
Harvey, W., Ginsberg, M. 1995. Limited discrepancy search. In: Proceedings of the 14th International Joint Conference on Artificial Intelligence (IJCAI-95), Morgan Kaufman, Montréal, pp. 607–615.
Ilhan, T., Iravani, S., Daskin, M., 2008. The orienteering problem with stochastic profits. IIE Transactions 40, 406–421.
Kantor, M., Rosenwein, M., 1992. The orienteering problem with time windows. The Journal of the Operational Research Society 43 (6), 629–635.
Kataoka, S., Morito, S., 1988. An algorithm for the single constraint maximum collection problem. Journal of the Operations Research Society of Japan 31 (4), 515–530.
Ke, L., Archetti, C., Feng, Z., 2008. Ants can solve the team orienteering problem. Computers and Industrial Engineering 54, 648–665.
Keller, C., 1989. Algorithms to solve the orienteering problem: A comparison. European Journal of Operational Research 41, 224–231.
Laporte, G., Martello, S., 1990. The selective travelling salesman problem. Discrete Applied Mathematics 26, 193–207.
Laporte, G., Rodriguez Martin, I., 2007. Locating a cycle in a transportation or a telecommunications network. Networks, 92–108.
Leifer, A., Rosenwein, M., 1994. Strong linear programming relaxations for the orienteering problem. European Journal of Operational Research 73, 517–523.
Liang, Y., Kulturel-Konak, S., Smith, A. 2002. Meta heuristics for the orienteering problem. In: Proceedings of the 2002 Congress on Evolutionary Computation. Hawaii, Honolulu, pp. 384–389.
Lin, S., 1965. Computer solutions of the traveling salesman problem. Bell System Technical Journal 44, 2245-–2269.
Mansini, R., Pelizzari, M., Wolfer, R. 2006. A granular variable neighbourhood search heuristic for the tour orienteering problem with time windows. Technical Report R.T 2006-02-52, University of Brescia, Italy.
Miller, C., Tucker, A., Zemlin, R., 1960. Integer programming formulations and travelling salesman problems. Journal of the ACM 7, 326–329.
Montemanni, R., Gambardella, L., 2009. Ant colony system for team orienteering problems with time windows. Foundations of computing and Decision Sciences 34 (4), 287–306.
Muyldermans, L., Beullens, P., Cattrysse, D., Van Oudheusden, D., 2005. Exploring variants of 2- and 3-opt for the general routing problem. Operations Research 53 (6), 982–995.
Pekny, J., Miller, D. 1990. An exact parallel algorithm for the resource constrained travelling salesman problem with application to scheduling with an aggregate deadline. In: Proceedings of the 1990 ACM Annual Conference on Cooperation, Washington, DC, USA, pp. 208–214.
Ramesh, R., Brown, K., 1991. An efficient four-phase heuristic for the generalized orienteering problem. Computers and Operations Research 18, 151–165.
Ramesh, R., Yoon, Y., Karwan, M., 1992. An optimal algorithm for the orienteering tour problem. ORSA Journal on Computing 4, 155–165.
Righini, G., Salani, M. 2006. Dynamic programming for the orienteering problem with time windows. Technical Report 91 2006, Dipartimento di Tecnologie dell'Informazione, Universita degli Studi Milano, Crema, Italy.
Righini, G., Salani, M., 2008. New dynamic programming algorithms for the resource constrained elementary shortest path. Networks 51 (3), 155–170.
Righini, G., Salani, M., 2009. Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. Computers & Operations Research 4, 1191–1203.
Schilde, M., Doerner, K., Hartl, R., Kiechle, G., 2009. Metaheuristics for the bi-objective orienteering problem. Swarm Intelligence 3, 179–201.
Solomon, M., 1987. Algorithms for the vehicle routing and scheduling problem with time window constraints. Operations Research 35, 254–265.
Souffriau, W., Vansteenwegen, P., Vanden Berghe, G., Van Oudheusden, G. in press. A path relinking approach for the team orienteering problem. Computers and Operations Research, doi: 10.1016/j.cor.2009.05.002.
Souffriau, W., Vansteenwegen, P., Vertommen, J., Vanden Berghe, G., Van Oudheusden, D., 2008. A personalised tourist trip design algorithm for mobile tourist guides. Applied Artificial Intelligence 22 (10), 964–985.
Tang, H., Miller-Hooks, E., 2005. A tabu search heuristic for the team orienteering problem. Computer and Operations Research 32, 1379–1407.
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.
Thomadsen, T., Stidsen, T. 2003. The quadratic selective travelling salesman problem. Informatics and Mathematical Modelling Technical Report 2003-17, Technical University of Denmark.
Tricoire, F., Romauch, M., Doerner, K., Hartl, R., 2010. Heuristics for the multi-period orienteering problem with multiple time windows. Computers and Operations Research 37 (2), 351–367.
Tsiligirides, T., 1984. Heuristic methods applied to orienteering. Journal of the Operational Research Society 35 (9), 797–809.
Vansteenwegen, P. 2008. Planning in tourism and public transportation – attraction selection by means of a personalised electronic tourist guide and train transfer scheduling. Ph.D. Dissertation, Katholieke Universiteit Leuven, Centre for Industrial Management, Belgium, ISBN: 978-90-5682-949-0.
Vansteenwegen, P., Souffriau, W., Van Oudheusden, D., 2009a. A detailed analysis of two metaheuristics for the team orienteering problem. In: Stützle, T., Birattari, M., Hoos, H. (Eds.), Engineering Stochastic Local Search, Lecture Notes in Computer Science, vol. 5752. Springer, Berlin, pp. 110–114.
Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D., 2009b. A guided local search metaheuristic for the team orienteering problem. European Journal of Operational Research 196 (1), 118–127.
Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D., 2009c. Metaheuristics for tourist trip planning. In: Geiger, M., Habenicht, W., Sevaux, M., Sörensen, K. (Eds.), Metaheuristics in the Service Industry, Lecture Notes in Economics and Mathematical Systems, vol. 624. Springer-Verlag, pp. 15–31.
Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D., 2009d. Iterated local search for the team orienteering problem with time windows. Computers and Operations Research 36 (12), 3281–3290.
Vansteenwegen, P., Van Oudheusden, D., 2007. The mobile tourist guide: An or opportunity. OR Insights 20 (3), 21–27.
Wang, Q., Sun, X., Golden, B., Jia, J., 1995. Using artificial neural networks to solve the orienteering problem. Annals of Operations Research 61, 111–120.
Wang, Q., Sun, X., Golden, B., 1996. Using artificial neural networks to solve generalized orienteering problems. In: Dagli, C., Akay, M., Chen, C., Fernández, B., Ghosh, J. (Eds.), Intelligent Engineering Systems Through Artificial Neural Networks, vol. 6. ASME Press, New York, pp. 063–1068.
Wang, X., Golden, B., Wasil, E. 2008. Using a genetic algorithm to solve the generalized orienteering problem. In: Golden, B., Raghavan, S., Wasil, E. (Eds.), The Vehicle Routing Problem: Latest Advances and New Challenges, pp. 263–274.
Wren, A., Holliday, A., 1972. Computer scheduling of vehicles from one or more depots to a number of delivery points. Operational Research Quarterly 23, 333–344.
Zong, G., Tseng, C.-L., Park, Y., 2005. Harmony search for generalized orienteering problem: Best touring in China. Advances in Natural Computation, 741–750.