universität wien

MASTERARBEIT / MASTER'S THESIS

Titel der Masterarbeit / Title of the Master's Thesis

„Branch-and-Price Algorithm for the Team Orienteering Problem with Time-Dependent Rewards“

verfasst von / submitted by

Elena Prishchepo

angestrebter akademischer Grad / in partial fulfilment of the requirements for the degree of

Master of Science (MSc)

Wien, 2017

Studienkennzahl lt. Studienblatt / degree programme code as it appears on the student record sheet:	A 066 915
Studienrichtung lt. Studienblatt / degree programme as it appears on the student record sheet:	Masterstudium Betriebswirtschaft
Betreut von / Supervisor:	o. Univ.-Prof. Dipl.-Ing. Dr. Richard Hartl

Zusammenfassung

Das Team Orienteering Problem mit zeitabhängigen Belohnungen (Team Orienteering Problem with Time-Dependent Rewards (TOPwTDR)) ist eine Erweiterung des klassischen Team Orienteering Problems, bei der der kundenbezogene Gewinn als eine Zeitfunktion dargestellt wird, z.B. linear fallende Funktion, exponentiell wachsende Funktion etc. Das Ziel des TOPwTDRs besteht darin, den Gesamtgewinn zu maximieren, so dass jeder Kunde innerhalb einer vorgeschriebenen Frist höchstens einmal besucht wird. Das TOPwTDR wurde bisher in der Literatur unter der Annahme von sinkenden Gewinnen betrachtet.

Die Dantzig-Wolfe-Zerlegung wird angewendet und das TOPwTDR wird durch einen Branch-and-Price-Algorithmus exakt gelöst. Zur Lösung des Pricing-Problems wird ein modifizierter Labelling-Algorithmus vorgeschlagen, der in der ersten Stufe für die Spaltenfindung Heuristiken verwendet, um das reduzierte Masterproblem einzugeben. Da die Dominanzregeln aufgrund der zeitabhängigen Gewinne sehr schwach sind, werden Schranken der reduzierten Kosten benutzt, um den Suchraum des Pricing-Problems deutlich zu reduzieren. Zwei Schranken werden implementiert, und zur weiteren Beschleunigung des Algorithmus werden spezielle Verfahren eingesetzt. Die Methode verwendet ein Branching Scheme, das mit dem vorgeschlagenen Spaltenerzeugungsverfahren kompatibel ist.

Für die Durchführung der Berechnungsexperimente werden die klassischen TOP-Instanzen aus der Fachliteratur verwendet, die je nach Problemart leicht modifiziert werden. Die Ergebnisse zeugen von der Wirksamkeit der Methode.

Table of Contents

List of Figures

List of Tables

Abstract

The Team Orienteering Problem with Time-Dependent Rewards (TOPwTDR) is an extension of the classical Team Orienteering Problem, where the profit associated with each customer is a function of time, e.g. linear decreasing, exponentially increasing, etc. The objective of the TOPwTDR is to maximise the total collected profit by finding vehicle routes, which visit each customer at most once within a prescribed time limit. The TOPwTDR has been considered previously in the literature under the assumption of decreasing profits.

The Dantzig-Wolfe decomposition is applied and the TOPwTDR is solved exactly with a Branch-and-Price algorithm. For tackling the Pricing problem a modified labelling algorithm is proposed, which on the first stage uses heuristics to find columns to enter the reduced master problem. As the dominance rules are very weak due to time-dependency of the profits, reduced costs bounding is used to reduce significantly the search space of the Pricing problem. Two of such bounds are implemented, and for further speeding up the algorithm a set of accelerating techniques are employed. The methodology is completed with a branching scheme compatible with the proposed Column Generation procedure.

Computational experiments are conducted on the classical TOP instances found in the literature, slightly modified according to the nature of the problem. The results demonstrate the effectiveness of the method.

1. Introduction

The Orienteering Problem (OP), also known as Selective Traveling Salesman Problem, originates from the orienteering sport, where competitors start at a specified point and try to visit as many checkpoints as possible and return to the control point within a given time frame (Chao et al. (1996a)). A certain score characterises each point and the goal of the game is to maximize the total collected score. While as many as possible points should be visited in available time span, not all the points must be visited. The shortest path must be chosen between the selected points. The OP combines the Knapsack Problem and the Travelling Salesman Problem (Vansteenwegen et al. (2011)). The OP differs from the classical Vehicle Routing Problem in the fact that in the OP not all customers have to be visited.

When more than one route is to be found the problem is called Team Orienteering Problem (TOP) and was first described by Chao et al. (1996b). This problem was also considered under the Multiple Tour Maximum Collection Problem, which was introduced by Butt and Cavalier (1994). Solving the OP is NP-Hard (Golden (1987), Laporte and Martello(1990)), and so is the TOP.

This paper considers the Team Orienteering Problem with Time-Dependent Rewards (TOPwTDR) and proposes an exact method based on the Branch-and-Price algorithm. To the best of our knowledge, the problem has been addressed before only under the assumption of decreasing rewards (Afsar and Lombardie (2013)), and in the generalised version is studied for the first time.

The paper is organized as follows. In section 2, the heuristic and exact methods, which have been proposed for solving TOP, are reviewed. Section 3 presents the problem description and its mathematical formulation. The proposed Branch-and-Price approach is described in section 4, followed by computational results and conclusion, in sections 5 and 6.

2. Literature review

The TOP has been extensively studied, and both heuristic and exact solution approaches have been proposed. Most studies have been focused on developing heuristic methods. Namely, Archetti et al. (2007) present four algorithms: two variants of Tabu Search algorithm, and two of the Variable Neighbourhood Search (VNS) algorithm.

Ke et al. (2008) propose four variations of the algorithm, which combines the Ant Colony Optimisation with the CROSS exchange procedure. Vansteenwegen et al. (2009) implement a variant of VNS, Skewed VNS, which allows accepting solutions slightly worse than current best solutions, if they are from a far enough neighbourhood. A memetic algorithm, combining Genetic Algorithm and local search procedure for the chromosome mutation, was proposed by Bouly et al. (2010). Souffriau et al. (2010) develop two versions of a combination of a Greedy Randomised Adaptive Search Procedure with a Path Relinking approach. For dealing with infeasible tours the approach uses a repairing algorithm on the phase of relinking. Tricoire et al. (2010) introduce a VNS algorithm for a generalised version of the OP and test it on the TOP instances. Another example of a population based metaheuristic is presented by Dang et al. (2013a) and combines the already mentioned memetic algorithm and the particle swarm optimisation algorithm, which in turn simulates the collective behaviour of wild animals. Giant tours, i.e. permutations of all customers, as in Bouly et al. (2010), are employed to encode indirectly particle positions. The algorithm allows exploring effectively larger neighbourhoods. Lin (2013) introduces a Multi-Start Simulated Annealing, which combines Simulated Annealing and multi-start hill climbing approaches, what helps to avoid local optimum. A modification of the Genetic Algorithm is presented by Ferreira et al. (2014). Ke at al. (2016) introduces a mimic operator, which adopt ideas of population-based metaheuristics, and generates a new solution similar to an incumbent solution and updates the population using Pareto dominance rules. A new local search operator, called swallow operator, is also introduced.

The first exact approach for solving the TOP has been developed by Butt and Ryan (1999) and is based on set covering formulation and designed for solving the specific variant of the TOP with heterogeneous fleet of vehicles. Poggi et al. (2010) introduce a Branch-and-Price-and-Cut algorithm, where two types of cuts are employed and the Pricing problem of the algorithm is tackled by dynamic programming. Later, Boussier et al. (2007) introduce a different algorithm, which tackle the Pricing problem as the Elementary Shortest Path Problem with Resource Constraints by dynamic programming, and can be easily adapted for tackling other Vehicle Routing Problems with profits. Dang et al. (2013b), convinced that potential of metaheuristics has been exhausted, propose a Branch-and-Cut algorithm, based on a linear formulation with a polynomial number of binary variables. The authors employ a set of cuts, based on valid inequalities and a set of dominance properties, which contains clique cuts, generalized sub-tour eliminations, symmetric breaking and a set of upper and lower bounds.

The most recent approaches include algorithms, proposed by Keshtkaran et al. (2016), El-Hajj et al. (2016) and Speranza et al. (2016). Keshtkaran et al. (2016) base their approach on the algorithm proposed by Boussier et al. (2007) and solve the Pricing problem using a bounded bidirectional dynamic programming, which is coupled with decremental space relaxation containing a two-staged dominance rule relaxation, and adopt the same branching scheme. A different approach is proposed by El-Hajj et al. (2016) and uses a linear formulation with a polynomial number of variables. The proposed cutting-plane based approach employs a set of the cuts, containing the cuts, which are studied by Dang et al. (2013b), and include also removal of irrelevant components and forcing mandatory customers. Speranza et al. (2016) base their approach on the alternative two-index formulation with a polynomial number of variables and constraints developed from Sequence Ordering Problem Formulation, which is strengthened by the introduction of the connectivity constraints. The authors propose a Branch-and-Cut algorithm, which similarly to Dang et al. (2013b) eliminates inaccessible customers and arcs.

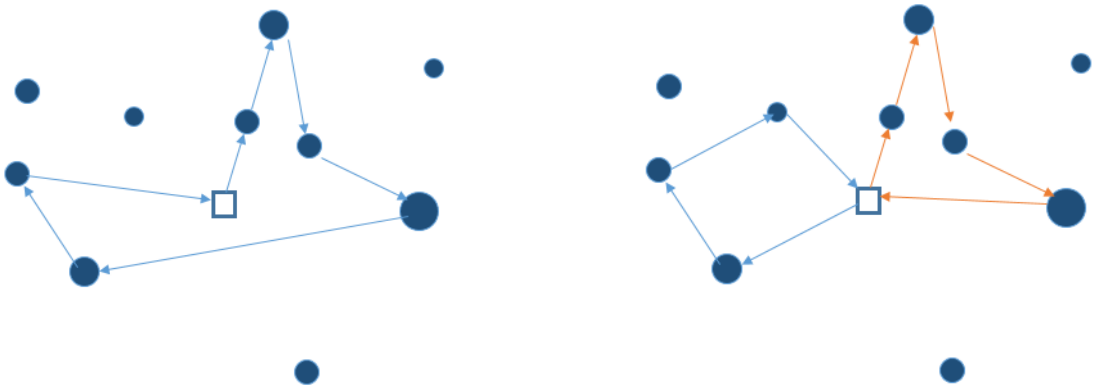
Afsar and Laporte (2013) introduce the TOP with Decreasing Profits, where the profit of each customer is a decreasing linear function of time and propose Evolutionary Local Search along with a Column Generation approach, based on the approach of Feillet et al. (2004). They test the methods on the TOP instances by Chao et al. (1996b), modified by adding variable profits.

For the recent survey on the OP, including the TOP, see Vansteenwegen et al. (2011) and Gunawan et al. (2016).

3. Problem definition and mathematical formulation

In the TOP points are visited by several sportsmen (a team), which corresponds to a fleet of vehicles in the transportation problem. *Figure 3Figure 1* depicts solutions for an orienteering problem and a TOP, where a set of customers must be chosen out of 10 possible to maximise collected profit without exceeding a given time limit.

In the TOPwTDR the profit associated with each customer considered to be visited depends on the time when it is visited, e.g. linear decreasing, exponentially increasing. The problem has been studied previously in the literature under the assumption of decreasing profits (Afsar and Labadie (2013)), where profits consist of fix and variable parts.



a. Orienteering Problem b. Team Orienteering Problem

Figure 1. Orienteering Problem and Team Orienteering Problem

Figure 2 İllustrates examples of customers' profit profiles. Thus, solution of the TOPwTDR must consider the fact that some customers are more profitable to visit at the beginning of the route, while others must be left until the last possible moment.

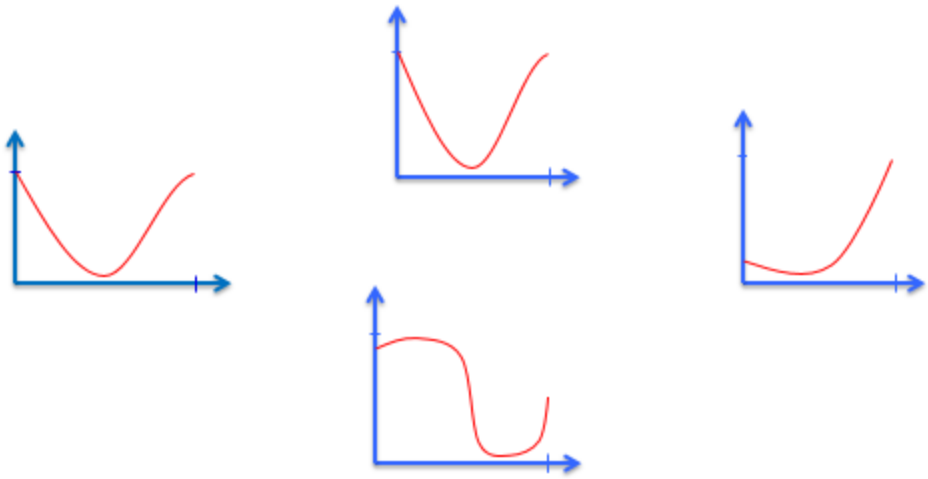


Figure 2. Examples of profit profiles for the TOPwTDR

There are different mathematical formulations of the TOP presented in the literature. The variant of the formulation is proposed in the survey of Vansteenwegen et al. (2011). Poggi et al. (2010) present additionally a less compact formulation and a decomposition for the TOP. Speranza et al. (2016) propose an alternative formulation

based on Sequence Ordering Problem Formulation. The following formulation is an extension of the formulation of Vanteenwegen et al. (2011).

Formally, the TOPwTDR, can be defined as follows. Let $G = (V, E)$ be a complete graph, where $V = \{v_0, v_1, \dots, v_{N+1}\}$ is a set of vertices and E is a set of edges. Vertices v_0 and v_{N+1} represent starting and ending depots, respectively, and vertices from v_1 to v_N represent customers. Each vertex is defined by time-dependent reward $F_i(t_i)$, where t_i is arrival time at customer v_i, note that $t_0 = 0$ and $F_0(t_0)$ and $F_{N+1}(t_{N+1})$ are equal to zero. Edges are defined by traveling time between two customers t_{ij}. Let *M* be a set of identical vehicles and $Tmax$ be a given time limit of the routes of these vehicles. When a customer is visited by a vehicle, the corresponding profit is added to the total profit of the vehicle route. The objective of the TOPwTDR is to maximise the total collected profit by finding vehicle routes, which visit each customer at most once within a prescribed time limit, such that the starting and the ending point of all routes coincide.

The problem can be stated as follows:

$$\boldsymbol{max} \sum_{m=1}^{M} \sum_{i=1}^{N} F_i(t_i) \mathrm{y}_{im} \quad (1)$$

subject to

$$\sum_{m=1}^{M} \sum_{j=1}^{N+1} x_{0jm} = \sum_{m=1}^{M} \sum_{i=0}^{N} x_{i(N+1)m} = M, \quad (2)$$

$$\sum_{m=1}^{M} y_{km} \leq 1; \quad \forall k = 1, \dots, N, \quad (3)$$

$$\sum_{i=0}^{N} x_{ikm} = \sum_{j=1}^{N+1} x_{kjm} = y_{km}; \quad \forall k = 1, \dots, N, \forall m = 1, \dots, M, \quad (4)$$

$$\sum_{i=0}^{N} \sum_{j=1}^{N+1} t_{ij} x_{ijm} \leq Tmax; \quad \forall m = 1, \dots, M, \quad (5)$$

$$1 \leq u_{im} \leq N + 1; \ \forall i = 1, \dots, N + 1; \ \forall m = 1, \dots, M \quad (6)$$

$$u_{im} - u_{jm} + 1 \leq N(1 - x_{ijm}); \quad \forall i, j = 1, \dots, N + 1; \forall m = 1, \dots, M, \quad (7)$$

$$x_{ijm}(t_j - t_i - t_{ij}) = 0; \quad \forall i, j = 0, \dots, N + 1; \ \forall m = 1, \dots, M \quad (8)$$

$$x_{ijm}, \mathrm{y}_{im} \in \{0,1\}; \quad \forall i, j = 0, \dots, N + 1; \ \forall m = 1, \dots, M \quad (9)$$

In this model, decision variable $x_{ijm} = 1$, if vehicle m visits customer v_j after customer v_i, and 0 otherwise; decision variable y_{im} = 1, if customer v_i is visited by vehicle m, and 0 otherwise; and decision variable u_{im} – position of customer v_i in the route of vehicle m. The objective function (1) calls for the maximisation of the collected reward. Constraints (2) impose that each route starts at the customer v_0 and ends at the customer v_{N+1}. Constrains (3) guarantee that every customer is visited at most once. Constrains (4) ensure the connectivity of all routes. Constraints (5) guarantee

that no route exceeds the time limit. Constraints (6) and (7) ensure absence of subtours. Constraints (8) define arrival times to each customer.

For applying the Branch-and-Price algorithm the decomposed mathematical formulation is used, which is presented in Section 4.c.

4. Branch-and-Price Algorithm

a. Motivation

Branch-and-Price is an exact algorithm for a big class of large scale integer programs, which has been used successfully for more than 30 years (Lübbecke and Desrosiers (2005)). The method has been studied broadly for scheduling and routing problems. The algorithm consists in embedding Column Generation in a Branch-and-Bound structure (Barnhart et al. (1998)). Desrochers (1988) proposes a Branch-and-Price approach to the Vehicle Routing with Time Windows based on a set partitioning reformulation, which later is improved by Boussier et al. (2007). Barnhart et al. (1998) present a generic scheme for Branch-and-Price on the examples of the Generalised Assignment Problem and the Crew Scheduling Problem. Feillet (2010) illustrates the generic methodology with the case of Vehicle Routing Problem with Time Windows. Hernandez et al. (2016) propose two Branch-and-Price algorithms for solving the Multi-Trip Vehicle Routing Problem with Time Windows based on two set covering formulations of the problem. Florio (2016) develop a methodology for the Stochastic Delivery Problem, who due to the time-dependent nature of the problem proposes using bounding techniques as an alternative to dominance rules in solving the Pricing problem.

The approach for solving the TOP has been proposed by Butt and Ryan (1999), which is based on set covering formulation and focuses on solving specific variant of the TOP with heterogeneous fleet of vehicles (the Multi-Trip Vehicle Routing Problem). The Pricing problem of the Branch-and-Price algorithm enumerates the set of nodes subsets, which could produce routes with positive reduced cost, and satisfy time restriction. Branching is executed on a nodes pairs, for 1-branch considering visiting both nodes of a pair and for 0-branch none of them.

Boussier et al. (2007) propose a different algorithm for the TOP with homogeneous fleet of vehicles. They solve the Pricing problem as the Elementary Shortest Path Problem with Resource Constraints by dynamic programming and propose two pre-processing

procedures, “label loading” and “meta extension”, which trigger finding routes with positive reduced cost, and several accelerating techniques to speed up the Pricing problem. They conduct two sorts of branchings, on a node and on an edge.

Keshtkaran et al. (2016) develop their approach based on the algorithm proposed by Boussier et al. (2007). They implement a different dynamic programming approach, where a bounded bidirectional dynamic programming is coupled with a decremental space relaxation containing a two-staged dominance rule relaxation, but apply the same branching scheme.

The approach described in this paper uses the formulation of the Master Problem proposed by Boussier et al. (2007) and adopts the ideas, presented by Florio (2016) for Stochastic Delivery Problem.

b. Description

Branch-and-Bound algorithm combines Branch-and-Bound and Column Generation; in each node of the Branch-and-Bound tree the linear relaxation of the problem is solved by Column Generation.

The approach starts from reformulating the problem in a form compatible with Column Generation procedure, which provides tighter linear programming relaxation bound. Using Dantzig-Wolfe decomposition a new mathematical formulation is obtained, where columns represent complete feasible routes. Relaxation of this formulation is called the Master Problem (MP). As the number of variables in the MP grows exponentially with the number of variables in the original formulation, the next stage of the algorithm is in defining the initial restricted sets of columns, the Restricted Master Problem (RMP). There are two ways to state initial set of variables: one introduces penalty for infeasible routes (Lübbecke and Desrosiers (2005)) and the other initiates already feasible routes, btw. *single node routes*, which go from the depot to one customer and return to the depot $\{0, v_i, N+1\}$ (Butt and Ryan (1999), Boussier et al. (2007), Keshtkaran et al. (2016)). Then, the RMP is solved using simplex method to provide dual variables values for the Pricing problem, which aims to find columns with positive reduced cost (in a maximization problem). New columns expand the RMP, which is solved again and followed by Column Generation (the Pricing problem), until no new columns are found to enter the RMP. If the found solution is fractional, branching occurs. The procedure repeats for each sub-problem. If the solution of the linear relaxation solved by Column

Generation is integer, then, the optimal solution has been found. The scheme of Branch-and-Bound Algorithm is illustrated in *Figure 3*.

All stages of the algorithm are discussed in detail in Sections 4.c-4.h.

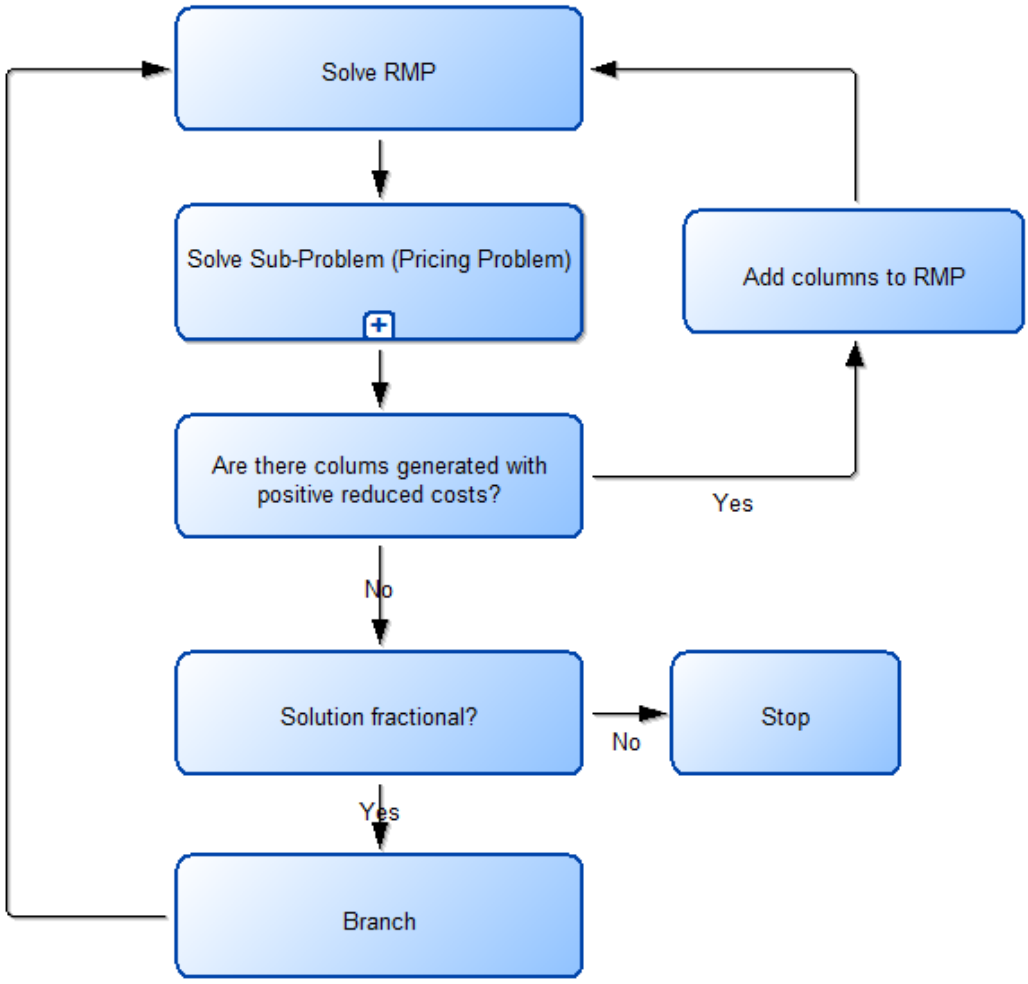


Figure 3. Branch-and-Price. Scheme

c. Master Problem, Restricted Master Problem, reduced cost

The Set-Packing formulation proposed by Boussier et al. (2007) is adopted to formulate the TOPwTDR. Let Ω= $\{r_1,...,r_{|\Omega|}\}$ be the set of all feasible routes r_k starting at the node v_0 and ending at the node v_{N+1}, with duration within a given time limit $Tmax$, and a_{ik} be the binary variables describing these routes, which is equal to 1, if the route r_k visits customer v_i, and 0 otherwise. Let p_k be the total profit generated by the route $r_k \in \Omega$, such that $p_k = \sum_{v_i \in V} a_{ik} F_i(t_i)$ (10). Then, the TOPwTDR can be stated as follows:

$$\boldsymbol{max} \sum_{r_k \in \Omega} p_k x_k \quad (11)$$

subject to

$$\sum_{r_k \in \Omega} a_{ik} x_k \leq 1 \quad (12)$$

$$\sum_{r_k \in \Omega} x_k \leq M \quad (13)$$

$$x_k \in \{0,1\} \quad (r_k \in \Omega) \quad (14)$$

Where the binary decision variable x_k is equal to 1, if the route r_k is included in the optimal solution, and 0 otherwise. The objective (11) is to maximize collected profit, ensuring that every customer is visited at most once (12), and no more than *M* vehicles are used (13).

Solving the linear relaxation of the problem (11)-(14) requires the use of Column Generation, because of the number of all possible routes $r_k \in \Omega$. The approach contains two parts: the RMP and the Pricing problem. The RMP is the MP, which consists of the restricted number of variables (columns) $\Omega_r \in \Omega$. In particular, the initial RMP is formed by all feasible *single node routes {0,* v_i, $N+1\}, \forall i = 1, \dots, N, such\ that\ t_{N+1} \leq Tmax.$ The aim of solving the RMP is to provide dual variables for the Pricing problem.

The dual of the linear relaxation of the problem has a following form:

$$\mathbf{min} \sum_{v_i \in V \setminus \{v_o \cup v_{n+1}\}} \lambda_i + m\lambda_0 \quad (15)$$

subject to

$$\sum_{v_i \in V \setminus \{v_o \cup v_{n+1}\}} a_{ik}\lambda_i + \lambda_0 \geq p_k \quad r_k \in \Omega_r, \quad (16)$$

$$\lambda_i, \lambda_0 \geq 0 \quad (17)$$

Where λ_i is the nonnegative dual variable related to the visit of customer v_i and λ_0 is the nonnegative dual variable related to the fleet size constraint.

Thus, the Pricing problem consist in finding routes with positive reduced cost:

$$p_k - \sum_{v_i \in V \setminus \{v_o \cup v_{n+1}\}} a_{ik}\lambda_i - \lambda_0 > 0;$$

or considering (10):

$$\sum_{v_i \in r_k} (F_i(t_i) - \lambda_i) - \lambda_0 > 0. \quad (18)$$

For solving the Pricing problem a modified labeling algorithm is proposed, which is described in details in Section 4.e.

d. Pseudocode

As it is not needed to solve the Pricing problem exactly until the very last iteration (Lübbecke and Desrosiers (2005)), some heuristics could be used to generate columns. As it could be seen in the

Figure 4 the algorithm starts from solving linear relaxation of the problem heuristically (*SolveHeuristic procedure*) and postpones solving it to optimality until the last iteration.

```
1: LowerBound ← 0
2: sol ← solveHeuristic
3: if intermidiateBinarySolution(IBS) exists then
4:    LowerBound ← IBS
5: end if
6: add sol in BnB node n to a pending list N
7: while N.size > 0 do
8:    next BnB node ←  max(n.sol, ∀n∈N), erase n from N
9:    filterColumns (n.edgesOn, n.edgesOff)
10:   n.sol ←  solveRMP
11:   if n.sol < lowerBound then
12:    n.sol ← solveUpper(LowerBound)
13:    if IBS exists && IBS > LowerBound then
14:          LowerBound ← IBS
15:    end if
16:    if n.sol ≤ lowerBound continue
17:   end if
18:   if n.sol is binary then
19:          n.sol ←  solveBinary
20:          if IBS exists && IBS > LowerBound then
21:          LowerBound <-  IBS
22:          end if
23:          if n.sol > lowerBound && n.sol is binary then
24:                LowerBound <- n.sol
25:                continue
26:          end if
27:    end if
28:    edgeToBranch ← edge with flow closest to 0.5
29:    add newEdgeOn to edgeToBranch.edgesOn
30:    filterColumns(newEdgesOn, n.edgesOff)
```

```
31:    sol ← solveHeuristic
32:    if IBS exists && IBS > LowerBound then
33:          LowerBound ← IBS
34:    end if
35:    add sol in BnB node n to a pending list N
36:    add newEdgesOff edgeToBranch.edgesOff
37:    sol ← solveHeuristic
38:    If IBS exists && IBS > LowerBound then
39:          LowerBound ← IBS
40:    end if
41:    add sol in Branch-and-Bound node n to a pending list N
42: end while
43: return LowerBound
```

Figure 4. Branch-and-Price. Pseudocode

The *SolveHeuristic procedure* (see

Figure 5) aims to build up faster the pool of the good columns and adds in one iteration not more than limited number of new columns `NUMHeur.` The generated columns are reused for the following sub-problems of the Branch-and-Bound problem, so is the binary solutions found, which could be considered as a lower bound of the Branch-and-Bound problem. For speeding up the Pricing problem a modified labeling algorithm (label recycling) is applied, which is discussed in Section 4.e and 4.g.

```
1: if columns.size ← 0 then
2:    RMP ← InitialRoutes // elementary routes(0,i,N+1)
3: end if
4: while fail ≤ maxFail do
4:    sol ← solveRMP
5:    if sol is binary then
6:          IBS ← sol
7:       end if
8:    pricing.solveHeuristic (find heuristically NUMHeur
           columns with positive reduced cost)
9:    add new columns to RMP
10: end while
```

Figure 5. SolveHeuristic procedure

For pruning unpromising branches of the Branch-and-Bound tree without solving the linear relaxation of the problem by Column Generation exactly the *SolveUpper procedure* is used (see

Figure 6), which returns a feasible solution with a value higher than the 'target' (lower bound), if there is such. The Pricing problem finds columns with reduced cost not just positive, but higher than the targeted value. And, finally, the linear relaxation of the problem is solved exactly (see *Figure 7*), conducting by the Pricing problem exhaustive search until solution becomes fractional, or until no more columns are found. In the last case, the algorithm returns an optimal binary solution.

```
1: target  ←  LowerBound
2: while (true) do
3:     sol ← solveRMP
4:     if (sol > target) return sol
5:     end if
6:     reducedCostTarget  ←  target-sol
7:     pricing.solveTarget (find columns with reduced cost higher
       than reducesCostTarget)
8:      if columns.size == 0 return sol
9:      end if
10:    add new columns to RMP
11: end while
```

Figure 6. SolveUpper procedure

```
1: while new columns > 0 do
2:     sol ← solveRMP
3:     pricing.solveExact (find NUMBinar columns with positive reduced cost)
4:     Add new columns to RMP
5: end while
```

Figure 7. SolveBinary

e. Pricing Problem

The Pricing problem consists in finding columns, which contribute to the increase of the solution value. A labeling algorithm is used, where labels represent feasible partial routes and at each iteration a node is selected and all its not yet handled labels are extended towards all possible successor nodes. To ensure that only feasible routes are

returned, the labels extensions cannot violate existing time restriction. Nodes are handled iteratively until no new labels are generated.

At the beginning, there is only one label at the depot with reduced cost value equal to $-\lambda_0$. The total reduced cost values of the partial routes are calculated according to (18).

For reducing the search space in the standard implementation, some dominance rules are applied to compare labels of the node and discard some of them. In the case of the TOPwTDR the dominance rules are very weak due to time-dependency of the profits and cannot be applied efficiently. For example, one label with higher total reduced cost and higher time remaining to complete the route, would usually dominate and be selected for the extension. However, as in the case of the TOPwTDR there could be nodes, which provide higher profit, if they are visited later, the potential total profit of the second label could be higher.

That is why two bounds on reduced cost are proposed instead: Knapsack Bound and Maximum Walk to the Depot Bound, which allow to discard labels, which extension will never generate routes with positive reduced cost. The bounds are discussed in the next section.

As any routes with positive reduced cost may contribute to increase of the objective value (Lübbecke and Desrosiers (2005)), the exhaustive search by solving the Pricing problem is postponed until the last stages, instead the modified labeling approach (label recycling algorithm) is used on the first stages of executing of the Branch-and-Price algorithm, which reuses the so far created labels (Florio (2016)). After every solving of the Pricing problem the labels are not discarded, but saved for the next iteration. After solving the RMP with simplex, new dual variables values are used to update the reduced cost values of the existing labels.

f. Bounds

Two bounds are used for reducing the search space of the Pricing problem.

The Knapsack Bound employs the greedy Knapsack problem solving approach, when, first, the items with higher value are packed. In the case of the TOPwTDR the items represent customers and are characterized by reduced cost and the minimum

distance to the customer from the other customers, the ratio of these two parameters are taken to measure the value of the item.

The linear relaxation of the Knapsack problem is conducted in the following way. At first, among all items, which have not been included in the partial route, all items which still could fit the Knapsack remaining capacity are selected (could be visited in the remaining time, including time, required to return to the depot). Then, the selected set of the items is sorted according to their ratio value. Starting from the highest value the Knapsack is filled by the items while the capacity allows. The last item could be taken partially. The pseudocode of the algorithm is presented in

Figure 8.

```
1: ubKb ← 0
2: cn ← last node in the route //current node
3: rt ← Tmax-t //remaining capacity(time)
4: if there is a node i among remaining nodes,
        such that dist(cn,i)+dist(i,depot) ≤ rt then
5:     items ← +i
6:     items[i].value=maxProfit(i, t, Tmax-dist(i,depot))-dual(i)
7:     items[i].cost= i == cn ? minDist(i): minDist(i, cn, rt)
8: end if
9: sort items (decreasing ratio)
10: if items[j].cost < rt  then
11:    add item j to Knapsack
12:    ubKb ← ubKb +items[j].value
13:   rt ← rt - items[j].cost
14: else
15:    ubKb ← ubKb +(rt/items[j].cost*items[j].value)
16: end if
17: return ubKb
```

Figure 8. Knapsack Bound

As an alternative the modified Knapsack Bound, a Look Ahead Knapsack Bound, is computed, using the algorithm proposed by Florio (2016), where each time two items from the sorted list are considered. If there is a pair of items such that v_i has lower value than v_{i-1} and both of them cannot be added to the Knapsack, as the remaining time is not enough to go from v_{i-1} to v_i and return to the depot, then, v_i can be skipped (see Figure 9).

The second bound, the Maximum Walk Depot Bound, considers the number of maximum walks to the depot (MWD), which could be still conducted within the remaining time, where walks are the number of edges (repetitions are permitted), which could be added to the route, to complete the route from the current node to the depot. The MWD Bound represents the maximum reduced cost, which could be added to the route from the MWD number of nodes. The procedure is adopted from the one, presented by Florio (2016) and is depicted in the *Figure 10*.

```
1: ubKbLa ← 0
2: cn ← last node in the route //current node
3: rt ← Tmax-t //remaining capacity (time)
4: sort items (decreasing ratio)
5: jump ← false
6: for all i in items do
7:     jump ← jump AND items[i-1].value ≥ items[i].value
8:   if jump AND minTimeDepot(cn, items[i-1].cn, items[i].cn) > rt then
9:         jump ← false
10:         continue
11:   end if
12:    if     items[i].cost < rt then
13:         ubKbLa ← ubKbLa+items[i].value
14:         rt ← rt -items[i].cost
15:         jump ← true
16:   else
17:        ubKbLa ← ubKbLa +(rt/items[j].cost*items[j].value)
18:   end if
19: end for
20: return ubKbLa
```

Figure 9. Knapsack Bound with Look Ahead

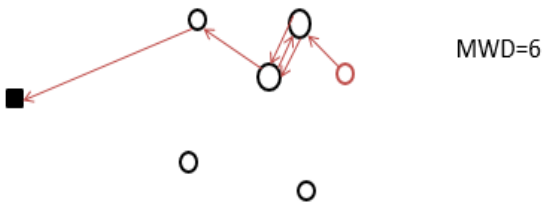


Figure 10. Maximum Walk Depot (Cf. Florio (2016), p. 17)

The algorithm starts by calculating the maximum reduced cost value for all not visited by the partial route customers, as a difference between the maximum profit, which could be collected in the remaining time, and the dual variable value λ_i. From the sorted list of the maximum reduced cost value the MWD number of the highest values is summed up to present the Maximum Walk Depot Bound.

For optimizing the calculation of the bound the values of the upper bound of the MWD are predefined for each node and each time moment, defined by number of time intervals, before starting the Pricing problem, which allow to accelerate the whole algorithm by the cost of O(N) time, where N is the number of customers.

g. Accelerating techniques

Two accelerating techniques are implemented to reduce computational time of the most time-consuming part of the algorithm, the Pricing problem. As it stated in the previous section by calculating the MWD Bound some pre-calculation are used to avoid exponentially repeated calculations by executing the Pricing problem.

Another accelerating technique consists in reusing labels already created by updating the reduced cost values from the current dual variables values (label recycling algorithm).

h. Branching

Branching is a challenging component of the Branch-and-Price algorithm and should take into consideration several aspects to be compatible with the Column Generation phase (Vanderbeck (2011)). Aiming at closing the difference between upper and lower bonds, branching scheme should guarantee providing improvement of upper bound and avoiding the significant complication of the master problem after branching (Vanderbeck (2000)), as columns need to be generated in the each node of the Branch-and-Bound tree (Barnhart et al. (1998)). A branching scheme must guarantee finiteness of the algorithm, separate the solution space, so that the desired fractional solution is eliminated, but integer solutions are retained (Lübbecke and Desrosiers (2005)).

An efficient Branch-and-Price approach for Vehicle Routing Problems conducts branching on the variables of the initial formulation, not on a fractional dual variable, and

includes branching on an edge and fleet size (Feillet (2010)), so is valid for the TOP because of the complexity of managing the prohibited routes (Boussier et al. (2007).

There are several branching approaches have been introduced for the TOP. Butt and Ryan (1999) propose to branch on a nodes pairs, for 1-branch considering visiting both nodes of a pair and for 0-branch none of them. Boussier et al. (2007) and Keshtkaran et al. (2016) combine two sorts of branching, on a node and on an edge, while Poggi et al. (2010) concentrate on branching on a node.

Branching on an edge, on a variable x_{ijm}, is investigated in this research. For branching, the edge with most fractional total flow is selected, i.e. the edge, which is transited by the amount of vehicles most close to 0.5; the 1-branch enforces visiting the edge and the 0-branch forbids visiting the edge. The next node of the Branch-and-Bound tree is selected in a greedy manner – the node with the highest solution value.

This type of constraints is easily handled in the Column Generation phase. Thus, all the columns which have been generated so far are used as initial set of columns for solving the sub-problems, after filtering among them columns which satisfy the branching condition (include the edge or not), and the Pricing problem is adjusted to generate columns considering the branching condition, namely, to ensure visiting the edge or to enforce the opposite.

For efficiently pruning unpromising branches of the Branch-and-Bound tree, bounds on reduced cost are used as it was described in Section 4.d.

5. Computational Experiments

The Branch-and-Price algorithm was implemented in C++, and CPLEX version 12.6.1 was used as the linear programming solver. The experiments were carried out on a Intel(R) Core(TM) i7-4790 @ 3.60Ghz with available memory 16GB. All the computational experiments were run with a time limit of 2 hours as in Boussier et al. (2007), Dang et al. (2013), Keshtkaran et al. (2016), Speranza et al. (2016) and El-Hajj et al. (2016).

The approach was tested on 48 TOP benchmark instanced proposed by Chao et al. (1996b). It comprises 4 sets with number of nodes $N + 1$=21, 32, 33 and 100. Each set consists of 12 instances and differs in number of vehicles, which ranges from 2 to 4, and time limit for each route $Tmax$. The set of the instances is described in the Table 1. For

transforming the TOP instances into the TOPwTDR instances four reward profiles were used (linear increasing, linear decreasing, V-shape and with fixed profit) and reward of the customers, given with the instances, considered as the maximum reward in the interval $(0, Tmax)$.

Tables 2 and 3 show the ability of the proposed Branch-and-Price algorithm to tackle the instances. 42 out of 48 instances were solved. The rest of the instances were not tackled within 2-hour time limit. In 27 cases, the optimal solutions are already found at the root node of the branch-and-bound tree with very small CPU. As predicted, the instances with smaller time limit are easier to tackle. Even though it is not seen from the tables, the Pricing problem is the most time-consuming part of the algorithm, while the number of branching does not affect drastically the computation time (e.g. p4.3.e).

Table 1. Benchmark instances

Number of nodes	21	32	33	100
Tmax	5-22	5-40	10-50	25-120
M	2/3/4	2/3/4	2/3/4	2/3/4
Number of instances	12	12	12	12

The Table 3 presents also the effect of using accelerating technique (modified Knapsack bound). Even though, for 41 instances the technique provides a positive impact, or does not affect the result, in 7 cases it concedes the basic model.

Table 2. TOPwTDR results

Set	M	N	# solved to optimality	# solved to optimality on root node
2	2	21	4	3
	3	21	4	4

	4	21	4	3
1	2	32	3	4
	3	32	4	2
	4	32	4	2
3	2	33	3	2
	3	33	4	1
	4	33	4	2
4	2	100	2	3
	3	100	3	1
	4	100	3	0
Total			**42**	**27**

Table 3. Computational results

Instance	Number of nodes	*M*	*Tmax*	CPU(s) Knapsack Bound	CPU(s) Knapsack Look Ahead Bound	# of branching
p2.2.b	21	2	10	0,01	0,01	0
p2.2.f	21	2	15	0,03	0,02	0
p2.2.j	21	2	20	0,65	0,54	7
p2.2.k	21	2	22	0,39	0,29	0
p2.3.a	21	3	5	0,00	0,00	0
p2.3.e	21	3	9	0,00	0,00	0
p2.3.f	21	3	10	0,00	0,00	0
p2.3.k	21	3	15	0,03	0,02	0

p2.4.c	21	4	5	0,00	0,00	0
p2.4.g	21	4	8	0,01	0,01	1
p2.4.j	21	4	10	0,00	0,00	0
p2.4.k	21	4	11	0,01	0,01	0
p1.2.j	32	2	25	0,20	0,19	0
p1.2.l	32	2	30	2,76	2,58	0
p1.2.q	32	2	40	771,79	635,38	0
p1.2.r	32	2	42	max	max	0
p1.3.f	32	3	10	0,01	0,01	1
p1.3.i	32	3	15	0,01	0,01	0
p1.3.p	32	3	25	0,88	0,84	5
p1.3.r	32	3	28	1,08	1,29	0
p1.4.h	32	4	10	0,00	0,00	1
p1.4.l	32	4	15	0,01	0,01	0
p1.4.q	32	4	20	0,04	0,03	0
p1.4.r	32	4	21	0,17	0,16	3
p3.2.j	33	2	30	21,47	max	5
p3.2.l	33	2	35	315,99	164,76	3
p3.2.n	33	2	40	995,70	722,82	0
p3.2.r	33	2	50	max	max	0
p3.3.d	33	3	10	0,01	0,01	0
p3.3.j	33	3	20	0,77	0,73	12
p3.3.p	33	3	30	79,38	61,43	23
p3.3.s	33	3	35	1393,35	1022,93	38
p3.4.f	33	4	10	0,00	0,01	0

p3.4.n	33	4	20	0,18	0,22	2
p3.4.r	33	4	25	6,84	10,57	11
p3.4.t	33	4	27	2,03	1,08	0
p4.2.c	100	2	35	208,44	132,46	1
p4.2.d	100	2	40	1680,22	1225,82	0
p4.2.e	100	2	45	max	max	0
p4.2.f	100	2	50	max	max	0
p4.3.b	100	3	20	0,17	0,18	1
p4.3.e	100	3	30	170,32	max	66
p4.3.h	100	3	40	4866,96	3216,64	2
p4.3.k	100	3	50	max	max	0
p4.4.b	100	4	15	0,12	0,12	7
p4.4.d	100	4	20	0,42	0,40	4
p4.4.h	100	4	30	107,72	90,67	39
p4.4.l	100	4	40	max	5492,18	15

6. Conclusion

This thesis presents the Team Orienteering Problem with Time Dependent Rewards and proposes a Branch-and-Price algorithm for finding the optimal solution for the problem. To the best of our knowledge, it is the first study of the problem. The approach applies a modified labelling algorithm for solving the Pricing problem with an implementation of two bounds on the reduced costs. A set of accelerating techniques are employed and a branching scheme compatible with the proposed Column Generation procedure is implemented. The computational results conducted on the modified classical TOP instances demonstrate the effectiveness of the proposed approach.

References

Afsar, H. M., & Labadie, N. (2013). Team orienteering problem with decreasing profits. Electronic Notes in Discrete Mathematics, 41, 285-293.

Archetti, C., Hertz, A., & Speranza, M. G. (2007). Metaheuristics for the team orienteering problem. Journal of Heuristics, 13(1), 49-76.

Barnhart, C., Johnson, E. L., Nemhauser, G. L., Savelsbergh, M. W., & Vance, P. H. (1998). Branch-and-price: Column generation for solving huge integer programs. Operations research, 46(3), 316-329.

Butt, S. E., & Cavalier, T. M. (1994). A heuristic for the multiple tour maximum collection problem. *Computers & Operations Research*, *21*(1), 101-111.

Butt, S. E., & Ryan, D. M. (1999). An optimal solution procedure for the multiple tour maximum collection problem using column generation. *Computers & Operations Research*, *26*(4), 427-441.

Boussier, S., Feillet, D., & Gendreau, M. (2007). An exact algorithm for team orienteering problems. 4OR: A Quarterly Journal of Operations Research, 5(3), 211-230.

Bouly, H., Dang, D. C., & Moukrim, A. (2010). A memetic algorithm for the team orienteering problem. 4OR: A Quarterly Journal of Operations Research, 8(1), 49-70.

Chao, I. M., Golden, B. L., & Wasil, E. A. (1996a). A fast and effective heuristic for the orienteering problem. *European journal of operational research*, *88*(3), 475-489.

Chao, I.-M., Golden, B. L., Wasil, E. A., (1996b). The team orienteering problem. European Journal of Operational Research 88 (3), 464-474.

Dang, D. C., Guibadj, R. N., & Moukrim, A. (2013a). An effective PSO-inspired algorithm for the team orienteering problem. European Journal of Operational Research, 229(2), 332-344.

Dang, D. C., El-Hajj, R., & Moukrim, A. (2013b). A branch-and-cut algorithm for solving the team orienteering problem. In International Conference on AI and OR Techniques in Constriant Programming for Combinatorial Optimization Problems (pp. 332-339). Springer Berlin Heidelberg.

Desrochers, M., Lenstra, J. K., Savelsbergh, M. W., & Soumis, F. (1988). Vehicle routing with time windows: optimization and approximation. Vehicle routing: Methods and studies, 16, 65-84.

El-Hajj, R., Dang, D. C., & Moukrim, A. (2016). Solving the team orienteering problem with cutting planes. Computers & Operations Research, 74, 21-30.

Feillet, D., Dejax, P., Gendreau, M., & Gueguen, C. (2004). An exact algorithm for the elementary shortest path problem with resource constraints: Application to some vehicle routing problems. Networks, 44(3), 216-229.

Feillet, D. (2010). A tutorial on column generation and branch-and-price for vehicle routing problems. 4OR: A Quarterly Journal of Operations Research, 8(4), 407-424.

Ferreira, D. F. (2011). Sisvar: a computer statistical analysis system. Ciência e agrotecnologia, 35(6), 1039-1042.

Florio, A. (2016), The stochastic delivery problem: introduction and solution by branch-and-price, Verolog 2016, June 05-09, Nantes.

Golden, B., Levy, L., Vohra, R. (1987). The orienteering problem. Naval Research Logistics, 34, 307–318.

Gunawan, A., Lau, H. C., & Vansteenwegen, P. (2016). Orienteering problem: A survey of recent variants, solution approaches and applications. European Journal of Operational Research, 255(2), 315-332.

Hernandez, F., Feillet, D., Giroudeau, R., & Naud, O. (2016). Branch-and-price algorithms for the solution of the multi-trip vehicle routing problem with time windows. European Journal of Operational Research, 249(2), 551-559.

Ke, L., Archetti, C., & Feng, Z. (2008). Ants can solve the team orienteering problem. Computers & Industrial Engineering, 54(3), 648-665.

Ke, L., Zhai, L., Li, J., & Chan, F. T. (2016). Pareto mimic algorithm: An approach to the team orienteering problem. Omega, 61, 155-166.

Keshtkaran, M., Ziarati, K., Bettinelli, A., & Vigo, D. (2016). Enhanced exact solution methods for the team orienteering problem. International Journal of Production Research, 54(2), 591-601.

Laporte, G., & Martello, S. (1990). The selective travelling salesman problem. *Discrete applied mathematics*, *26*(2-3), 193-207.

Lin, S. W. (2013). Solving the team orienteering problem using effective multi-start simulated annealing. Applied Soft Computing, 13(2), 1064-1073.

Lübbecke, M. E., & Desrosiers, J. (2005). Selected topics in column generation. Operations Research, 53(6), 1007-1023.

Poggi, M., Viana, H., & Uchoa, E. (2010). The team orienteering problem: Formulations and branch-cut and price. In OASIcs-OpenAccess Series in Informatics (Vol. 14). Schloss Dagstuhl-Leibniz-Zentrum fuer Informatik.

Souffriau, W., Vansteenwegen, P., Berghe, G. V., & Van Oudheusden, D. (2010). A path relinking approach for the team orienteering problem. Computers & operations research, 37(11), 1853-1859.

Speranza, M.G., Mansini, R., Bianchessi, N. (2016). A branch-and-cut algorithm for the Team Orienteering Problem. ReseachGate. https://www.researchgate.net/publication/306424574_A_branch-and-ut_algorithm_for_the_Team_Orienteering_Problem. Available from: M.Grazia Speranza, Aug 24, 2016.

Tricoire, F., Romauch, M., Doerner, K. F., & Hartl, R. F. (2010). Heuristics for the multi-period orienteering problem with multiple time windows. Computers & Operations Research, 37(2), 351-367.

Vanderbeck, F. (2000). On Dantzig-Wolfe decomposition in integer programming and ways to perform branching in a branch-and-price algorithm. Operations Research, 48(1), 111-128.

Vanderbeck, F. (2011). Branching in branch-and-price: a generic scheme. *Mathematical Programming*, *130*(2), 249-294.

Vansteenwegen, P., Souffriau, W., Berghe, G. V., & Van Oudheusden, D. (2009). Metaheuristics for tourist trip planning. In Metaheuristics in the service industry (pp. 15-31). Springer Berlin Heidelberg.

Vansteenwegen, P., Souffriau, W., & Van Oudheusden, D. (2011). The orienteering problem: A survey. European Journal of Operational Research, 209(1), 1-10.