A column generation algorithm for the team orienteering problem with time windows

Racha El-Hajj, Aziz Moukrim, B Chebaro, M Kobeissi

A COLUMN GENERATION ALGORITHM FOR THE TEAM ORIENTEERING PROBLEM WITH TIME WINDOWS*

R. El-Hajj[1, 2**], A. Moukrim[1], B. Chebaro[2] and M. Kobeissi[2]
[1]Sorbonne universités, Université de Technologie de Compiègne, CNRS
Heudiasyc UMR 7253, CS 60 319, 60 203 Compiègne cedex
{racha.el-hajj, aziz.moukrim}@hds.utc.fr

[2]Université Libanaise, École Doctorale des Sciences et de Technologie
Campus Hadath, Beyrouth, Liban
{bchebaro, mohamed.kobeissi}@ul.edu.lb

ABSTRACT

The Team Orienteering Problem with Time Windows (TOPTW) is a well-known variant of the Vehicle Routing Problem (VRP) whose aim is to maximize the total amount of profit collected from the visited customers while taking into consideration some resource limitations. In this variant, customers must be served at most once in their time windows while respecting the predefined maximum travel time limit of each vehicle. In this paper we propose a column generation based algorithm to solve the linear relaxation of TOPTW. A dynamic programming algorithm is used to solve the subproblems in order to generate additional columns. Experiments conducted on the benchmark of TOPTW show the effectiveness of our algorithm and the strengthen of our formulation since we were able to prove the optimality of several instances by finding their integer solutions at the root node while solving the linear relaxation of the model.

Keywords: Column Generation, Dynamic Programming, Vehicle Routing Problem with profit, Time Windows.

1 INTRODUCTION

The Team Orienteering Problem with Time windows (TOPTW) is a well-known variant of the Team Orienteering Problem (TOP). This variant derives from the Vehicle Routing Problem (VRP) in which not all the customers must be visited due to some resource limitations. In order to respect the availability of the customers, Kantor et al. [7] introduced this new variant of TOP, which is the Team Orienteering Problem with time windows. In TOPTW a fleet of vehicles is available to visit customers from a potential set and is associated with a predefined travel time limit and one particular depot from which it must start and end its path. An amount of profit is associated with each customer

* This work is carried out in the framework of the Labex MS2T and is partially supported by the Regional Council of Picardie, under TOURNEES SELECTIVES project.

** Corresponding Author

and can be collected at most once by the fleet and a service time needed to serve the customer. In addition, a time window is associated with each customer to express its availability, i.e., if a vehicle arrives before the allowed starting time, it must wait until the opening of the time window, and an arrival after the end of the time window is not accepted, where in this case the customer cannot be served. The aim of TOPTW is to select customers from the potential set and organize an itinerary of visits in order to maximize the total amount of collected profits while respecting the time windows of all the visited customers without exceeding the travel time limit of each vehicle.

TOPTW arises in many real life applications. It mainly appears in the tourist guide service as described by Vansteenwegen et al. [12]. The aim of this application is to maximize the satisfaction of the tourist while taking into consideration its interest on attractive places, the opening hours of these places and the duration of stay. Those planning problems are called Tourist Trip Design Problems (TTDPs) for which the TOPTW is considered as a simplified version. TOPTW also appears in the industrial and the transportation problems, as the routing of technicians [11] and fuel delivery problems [5].

Many heuristics had been developed to solve TOPTW. A tree heuristic was the first approximate method proposed by Kantor et al. [7] to solve this problem. Later, a Greedy Randomized Adaptive Search Procedure combined with the Evolutionary Local Search heuristic was proposed by Labadi et al. [8] and a Particle Swarm Optimization based heuristic was developed by Guibadj et al. [6]. To the best of our knowledge, only one exact method based on dynamic programming (DP) was proposed by Righini et al. [9] to solve OPTW, which is a particular case of TOPTW where only one vehicle is available for the service of customers. A state space relaxation strategies and some initialization heuristics were additionally used to speed up the dynamic programming algorithm.

In this paper we propose a column generation based algorithm to solve the linear relaxation of TOPTW. A dynamic programming algorithm was used to solve the subproblems to generate additional columns. These columns are added to the master problem at each iteration. The remainder of this paper is as follows. First, we give a formal description of TOPTW with its mathematical formulation. Then we describe our Column Generation method combined with the dynamic programming. Then, we present some computational experiments followed by some conclusions and further developments.

2 PROBLEM FORMULATION

TOPTW is modeled with a complete directed graph $G = (V; E)$, where $V = \{1, \dots, n\} \cup \{0\}$ is the set of vertices representing the customers and the depot and $E = \{(i, j) | i, j \in V\}$ the set of arcs linking the different vertices together. Vertex 0 represents the depot for each vehicle. We use V^- to denote the set of customers only. A set F of vehicles is available to visit customers where a travel time limit L is associated with each vehicle and must not be exceeded. A profit p_i ($p_0 = 0$), a service time s_i ($s_0 = 0$) and a time window $[e_i; \ell_i]$ are associated with each vertex i. Therefore, a customer cannot be served after its latest service time ℓ_i and if a vehicle arrives before the earliest service time e_i, it will incur some waiting time (for the depot the time window considered is $[0; L]$). A travel time c_{ij} is associated with each arc $(i, j) \in E$ and assumed to be satisfying the triangle inequality.

The problem can be formulated in a Mixed Integer Programming (MIP) where a polynomial number of decision variables y_{ir}, x_{ijr} and t_{ir} is used: $y_{ir} = 1$ if client i is served by vehicle r and 0 otherwise; $x_{ijr} = 1$ if arc (i, j) is used by vehicle r to serve customer i then customer j and 0 otherwise; t_{ir} represents the service time when vehicle r starts serving customer i.

$$\max \sum_{i \in V^-} \sum_{r \in F} y_{ir} p_i \tag{1}$$

$$\sum_{r \in F} y_{ir} \leq 1 \qquad \forall i \in V^- \tag{2}$$

$$\sum_{j \in V^-} x_{0jr} = \sum_{j \in V^-} x_{j0r} = 1 \qquad \forall r \in F \tag{3}$$

$$\sum_{i \in V \setminus \{k\}} x_{kir} = \sum_{j \in V \setminus \{k\}} x_{jkr} = y_{kr} \qquad \forall k \in V^-, \forall r \in F \tag{4}$$

$$e_i \leq t_{ir} \leq \ell_i \qquad \forall i \in V, \forall r \in F \tag{5}$$

$$t_{ir} - t_{jr} + (\ell_i + c_{ij} + s_i) x_{ijr} \leq \ell_i \qquad \forall i \in V, \forall j \in V, \forall r \in F \tag{6}$$

$$x_{ijr} \in \{0,1\} \qquad \forall i \in V, \forall j \in V, \forall r \in F \tag{7}$$

$$y_{ir} \in \{0,1\} \qquad \forall i \in V^-, \forall r \in F \tag{8}$$

$$t_{ir} \in \mathbb{R}^+ \qquad \forall i \in V, \forall r \in F \tag{9}$$

The objective function (1) is to maximize the sum of the collected profits. Constraints (2) force each customer to be visited at most once by the fleet of vehicles. Constraints (3) ensure that all vehicles start and end their paths at vertex 0. The connectivity of the routes is ensured by constraints (4). Constraints (5) guarantee the respect of the time windows of each customer and constraints (6) ensure that subtours, i.e. cycles excluding the depot, are forbidden. The integral requirement on variables is imposed by constraints (7) and (8), whereas the start of the service time can in general take nonnegative real values in (9).

3 COLUMN GENERATION ALGORITHM

Since TOP is an NP-Hard problem [3], its variant TOPTW is also NP-Hard. Solving its linear formulation is very difficult and requires a huge computational time. For this reason, a Column Generation (CG) algorithm would be more suitable to solve this problem. This method consists of evaluating feasible solutions in order to choose the best ones among them. Since it is impossible to add all the variables to the model, we start solving our master problem with a small number of variables, and then we enhance the search space by adding new variables at each iteration using the corresponding subproblem to generate them.

3.1 Master Problem

In order to apply the column generation algorithm, we formulate TOPTW using the classical set packing formulation. In this formulation, each variable represents a feasible route that respects all the constraints. The objective of solving this formulation is to choose the best set of routes that maximizes the profit. For this reason, we consider the set of all feasible routes $\Omega = \{r_1, r_2, \dots, r_{|\Omega|}\}$, which contains the routes that start and end at 0, and visit each customer at most once in its time window and such that the total traveled distance does not exceed L. Let P_k be the total profit collected by route r_k. We consider a constant $a_{ik} = 1$ if route r_k visits customer i and 0 otherwise. A decision variable x_k is used to indicate whether route r_k is considered and 0 otherwise. Therefore, the master problem can be stated as follows:

$$\max \sum_{r_k \in \Omega} P_k x_k \tag{10}$$

$$\sum_{r_k \in \Omega} a_{ik} x_k \leq 1 \qquad \forall i \in V^- \tag{11}$$

$$\sum_{r_k \in \Omega} x_k \leq m \tag{12}$$

$$x_k \in \{0,1\} \qquad \forall r_k \in \Omega \tag{13}$$

The objective function (10) aims to maximize the total collected profit from the considered routes. Constraints (11) ensure that each customer is visited at most once by the fleet of vehicles. Constraint (12) limits the number of available vehicles, and constraints (13) guarantee the integrity of the variables.

Since the number of variables in this linear formulation is exponential, it is impossible to quickly solve this problem. Therefore, a column generation algorithm is required, where the linear relaxation of the model (10) to (13) represents the master problem (MP). We focus in this article on solving the linear relaxation of TOPTW.

Because of the exponential size of Ω, we first consider a small subset $\Omega_1 \subset \Omega$ in the master problem. The resulting linear program is called Restricted Master Problem RMP in which only the subset Ω_1 of variables is considered. Then to enhance the subset Ω_1 a subproblem is solved to generate new variables, called columns, to add them to RMP.

3.2 SUBPROBLEM

As in Boussier et al. [2], we first initialize $RMP(\Omega_1)$ with a simple set of routes, where a single customer is visited in each route. Then, at each iteration of the algorithm, we solve $RMP(\Omega_1)$ to the optimality with the simplex method, and we get the optimal dual variables. We denote by λ_i the nonnegative dual variable associated with constraint (11) for customer i and by λ_0 the nonnegative dual variable associated with constraint (12). The subproblem is then solved to determine whether the new set of variables x_k with $r_k \in \Omega \setminus \Omega_1$ that have positive reduced costs. In other words, we determine variables that respect the following condition:

$$\sum_{i \in V^-} a_{ik} \lambda_i + \lambda_0 < P_k \tag{14}$$

At each iteration, one or several variables with positive reduced cost are generated and added to $RMP(\Omega_1)$ and the procedure reiterates. The global algorithm stops when the subproblem fails to find new routes with positive reduced cost, which means that when the generated routes do not respect the following condition:

$$\sum_{i \in r_k} (p_i - \lambda_i) + (p_0 - \lambda_0) > 0 \tag{15}$$

We use the notation $i \in r_k$ to indicate that costumer i is visited in route r_k. The objective of solving the subproblem is to determine the best feasible route that maximizes the total reduced cost while

respecting all the limitation constraints. This subproblem can then be seen as an OPTW where the profit of each customer i is set to be the reduced cost associated with the constraint corresponding to customer i, which is $(p_i - \lambda_i)$ denoted by reward ω_i. At each iteration, the new set of reduced costs is sent to the subproblem to generate new feasible solutions with a positive reduced cost to add them to the master problem.

Solving this subproblem is also NP-Hard. Thus we applied a bi-directional dynamic programming algorithm to solve the OPTW based on the one proposed by Righini et al. [9].

In this algorithm, a label $L_i = (S, \tau, W, i)$ is used to represent the ordered list of visited customers S, the total traveled distance τ, the total collected reward W and the last visited customer i in the path. To extend a label from $L_i = (S, \tau, W, i)$ to $L_j = (S', \tau', W', j)$ by adding customer j at the end of the first path, we must first make sure that customer j is not already visited in the path ($j \notin S$) and that the resulting traveled distance does not exceed the limit. To check this feasibility criterion, two strategies are followed, depending on whether it is a forward extension or a backward extension:

- For a forward extension: $\tau' = max\{\tau + c_{ij} + s_i, e_j\} \leq l_j$
- For a backward extension: $\tau' = max\{\tau + c_{ij} + s_i, L - l_j - s_j\} \leq L - e_j - s_j$

Once all the above conditions are respected, the resulting label is generated by adding customer j to the list S and its reward to the total collected reward W. The total traveled distance is then updated according to the type of extension.

To build the complete path, we join forward label with backward label, while making sure that there is no common customer between both of the paths and that the resulting traveled time does not exceed the maximum limit.

Using this dynamic programming algorithm, we generate at each iteration a set of feasible routes with positive reduced costs and add them to the master problem. This procedure is repeated until no other feasible routes with a positive reduced costs can be generated.

4 COMPUTATIONAL RESULTS

We coded our algorithm with C++, where we embedded our global Column Generation scheme in the framework SCIP 3.1.1. CPLEX 12.6 was used as the Mixed Integer Programming solver to solve the mathematical model in the master problems. We tested our algorithm on the standard benchmark of TOPTW using an AMD Opteron 2.6 GHz. In this section, we first describe the structure of the instances and their characteristics then we present a comparison between the results obtained by our column generation algorithm and those obtained by the other methods in the literature.

4.1 Benchmark of TOPTW

The benchmark of TOPTW is composed of two sets. The first set was proposed by Solomon [10] and is composed of 3 groups (R, C and RC) according to the distribution of customers, where R holds for random distribution, C for clustered one and RC for a combination between random and clustered distribution. In total, this set comprises 116 instances where the number of customers is fixed to 100 while the number of vehicles varies from 1 to 4 and the maximum travel length is between 230 and 1236. The second set was proposed by Cordeau et al. [4]. This set comprises 80 instances where the number of customers varies from 48 to 288 and the number of vehicles is between 1 and 4. The maximum travel time for this set is fixed to 1000 for all the instances. The details of the instances can be seen in Table 1.

Table 1. Benchmark of TOPTW

Set	Solomon [10]			Cordeau et al. [4]
	R	C	RC	PR
n	100	100	100	$48-288$
m	$1-4$	$1-4$	$1-4$	$1-4$
L	230	1236	240	1000
$\#\ instances$	48	36	32	80

4.2 COMPARISON WITH THE LITERATURE

To the best of our knowledge, Righini et al. [9] were the only ones who proposed an exact method for the Orienteering Problem with Time Windows, which is a particular case of TOPTW, where only one vehicle is available for the service of customers. Till now, there is no other exact method performed for the Team Orienteering Problem with Time Windows. Thus we first compare our results to those of Righini et al. [9] for the case of a single vehicle, and then we present the instances that we were able to solve for TOPTW with multiple vehicles.

Table 2 shows a comparison between our column generation algorithm and the dynamic programming algorithm proposed by Righini et al. [9] on the instances of OPTW. We note that to have a fair comparison between the two methods, we launched them both on the same computational machine and we fixed the time limit to two hours for each instance, as performed in Righini et al. [9]. We present in this table the number of optimal solutions found by each method and the average CPU time in second for each set of instances calculated for the commonly solved instances among the two methods.

Table 2. Comparison with the literature for the instances of OPTW

Set	Dynamic Programming		Column Generation	
	$\#\ solved\ instances$	$AvgCPU$	$\#\ solved\ instances$	$AvgCPU$
R	10/12	225.08	11/12	124.159
C	9/9	13.71	8/9	15.5
RC	8/8	1.736	8/8	1.35
PR	6/20	0.42	7/20	0.51

Based on these results, we notice that our column generation algorithm finds better results than the dynamic algorithm of Righini et al. [9]. Our proposed algorithm was able to find the optimal solution for all the instances being solved by the dynamic programming algorithm except for one instance in the class C, where our CG had found some difficulties when the customers are distributed in a clustered repartition form. In addition, we were able to find the optimal solution of two additional instances in the set R of Solomon [10] and the set PR of Cordeau et al. [4]. Based on the results obtained in this table, we observe that our proposed column generation algorithm was able to find, at the root, integer solutions for a large number of instances in these sets, and did not need any branching rule to reach the optimal solutions. This result shows the strengthen of the column

generation formulation and the efficiency of the subproblem solution. Furthermore, we notice that the computational times of our column generation algorithm are much smaller than those obtained with the dynamic programming, which reinforce the outperformance of the column generation algorithm in terms of quality and computational time.

For the case of TOPTW where 2 to 4 vehicles are available for the service of customers, we launched our column generation algorithm and we were able to find the optimal solution for a large number of instances from the benchmark. These instances are presented in Table 3, where the number of instances being solved to optimality in each set is shown, followed by the average computational time in seconds.

Table 3. Additional instances solved by our Column Generation algorithm

Set	Column Generation	
	$\#\ solved\ instances$	$AvgCPU$
R	17/48	291.78
C	15/36	177.47
RC	22/32	28.87
PR	11/80	620.6
$Total$	65	197.73

Based on the results presented in Table 3, we can deduce that our column generation algorithm for the linear relaxation of TOPTW is very competitive with the literature since we were able to prove the optimality of 65 new instances among all the sets of the benchmark with a considerably small computational time.

5 CONCLUSION

In this paper we treated the Team Orienteering Problem with Time Windows. This problem is one of the variants of the team orienteering problem where the availability of each customer must be respected by the vehicles. To solve the linear relaxation of this problem, we developed a column generation algorithm where we used a bi-directional dynamic programming algorithm to generate additional columns at each iteration. The computational results obtained on the benchmark of TOPTW show the competitiveness of our approach since we were able to find the optimal solutions of all the instances already solved in the literature and prove the optimality of several new instances still unsolved by any other exact method.

For future work, we aim to integrate some branching rules to reach the integer optimal solutions for more instances from the literature. We will also work on integrating new valid inequalities in order to perform a branch-cut and price algorithm for the Team Orienteering Problem with Time Windows. In addition, we will extend our research field to respond to new needs by imposing more resource limitations as the vehicle capacity. We may also introduce the synchronization between vehicles by forcing two or more vehicles to serve the same customer simultaneously [1]. These variants are not yet treated in the literature, although they might be confronted in many real life applications.

Bibliography

[1] S. Afifi, D.-C. Dang, and A. Moukrim. A simulated annealing algorithm for the vehicle routing problem with time windows and synchronization constraints. In *LION 7*, Lecture notes in computer science, pages 259--265, 2013.

[2] S. Boussier, D. Feillet, and M. Gendreau. An exact algorithm for team orienteering problems. *4OR*, 5(3):211--230, 2007.

[3] I-M. Chao, B. Golden, and E.A. Wasil. The team orienteering problem. *European Journal of Operational Research*, 88:464--474, 1996.

[4] J.-F. Cordeau, M. Gendreau, and G. Laporte. A tabu search heuristic for periodic and multi-depot vehicle routing problems. *Networks*, 30(2):105--119, 1997.

[5] B. Golden, L. Levy, and R. Vohra. The orienteering problem. *Naval Research Logistics*, 34(3):307--318, 1987.

[6] R.-N. Guibadj and A. Moukrim. A memetic algorithm with an efficient split procedure for the team orienteering problem with time windows. In *Artificial Evolution*, Lecture notes in computer science, pages 6--17, 2013.

[7] M.G. Kantor and M.B. Rosenwein. The orienteering problem with time windows. *The Journal of the Operational Research Society*, 43(6):629--635, 1992.

[8] N. Labadie, R. Mansini, J. Melechovský, and R. W. Calvo. The team orienteering problem with time windows: An lp-based granular variable neighborhood search. *European Journal of Operational Research*, 220:15--27, 2012.

[9] G. Righini and M. Salani. Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming. 36(4):1191--1203, 2009.

[10] M. M. Solomon. Algorithms for the vehicle routing and scheduling problems with time window constraints. *Operations Research*, 35(2):254--265, 1987.

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

[12] P. Vansteenwegen, W. Souffriau, G. Vanden Berghe, and D. Van Oudheusden. Metaheuristics for tourist trip planning. In *Metaheuristics in the Service Industry*, volume 624 of *Lecture Notes in Economics and Mathematical Systems*, pages 15--31. Springer Berlin Heidelberg, 2009.