The multi-constraint team orienteering problem with time windows in the context of distribution problems: A variable neighborhood search algorithm

Brahim Aghezzaf
Laboratoire d'Informatique et Aide à la Décision
Faculté des Sciences Aïn Chock (FSAC)
Université Hassan II de Casablanca
Email: b.aghezzaf@fsac.ac.ma

Hassan EL Fahim
Laboratoire d'Informatique et Aide à la Décision
Faculté des Sciences Aïn Chock (FSAC)
Université Hassan II de Casablanca
Email: elfahimhassan@gmail.com

***Abstract*—In this paper we present a new model for the multi-constraint team orienteering problem with time windows (MC-TOPTW) in the context of distribution problems, the resulting problem is called the capacitated team orienteering problem with time windows abbreviated as CTOPTW. In this problem, a set of vertices is given each with a known demand, a profit, a service time and a time window. The objective is to create a set of a predetermined number of routes that maximizes the total collected profit such that the time restriction and the vehicle capacity constraints are respected on each route. We propose a new variable neighborhood search algorithm to solve this problem. Computational results on the test instances available in the literature show that the proposed algorithm is competitive with other existing approaches in the literature and it is able to achieve a new best known solutions.**

***Index Terms*—Transportation, capacitated team orienteering problem, time window, variable neighborhood search**

I. INTRODUCTION

The vehicle routing problem (VRP) is one of the most studied combinatorial optimization problems. The first version was introduced in 1959 by [1] as a problem of satisfying a set of vertices with known demands with an objective of minimizing the total travel cost.

The vehicle routing problem with profits (VRPP) is a variant of the VRP in which visiting all the vertices is not possible for some logistics reasons. When the number of vehicles used to satisfy the vertices is limited to 1, the problem is known as the traveling salesman problem with profits (TSPP). This TSP is by nature a bi-objective optimization problem in which two objectives are in conflict, the first objective is the minimization of the total travel cost and the second objective is the maximization of the total collected profit. Despite this characteristic, mono-objective versions have received more attention in the literature. For review of the class of traveling salesman problems with profits the reader is referred to the very good survey of [2].

The orienteering problem (OP) is a very interesting mono-objective version of the TSPP since it arises in many-real life transportation problems. In the OP, a set of vertices is given each with a known profit. The objective is to design a feasible route that maximizes the total collected profit such that the total travel time is limited by a given time budget. The OP is also known in the literature as the selective traveling salesman problem, the maximum collection problem and the bank robber problem.

Exact methods were proposed to solve the OP to optimality. [3] developed an exact algorithm based on branch-and-bound method to solve instances with less than 20 vertices. A branch-and-cut algorithm was developed by [4] which is able to solve instances with up to 500 vertices. Heuristics and metaheuristics were proposed to solve the OP. [5] developed a five-step heuristic. A genetic algorithm was developed by [6].

An extension of the OP is its version with multiple routes called the team orienteering problem (TOP). The objective is to design a set of m routes in order to maximize the total collected profit. Exact algorithms were proposed to solve the TOP to optimality. [7] and [8] developed an exact algorithm based on branch-and-price and column generation methods respectively. Approach methods have drawn the attention of researchers. [9] developed two variants of a tabu search algorithm and a variable neighborhood search algorithm. Another tabu search was developed by [10]. A guided local search was developed by [11]. A multi-start simulated annealing algorithm was developed by [12].

A variant of the TOP is its constrained version called the capacitated team orienteering problem (CTOP). In this problem, a set of homogeneous vehicle of limited capacity is considered to satisfy the vertices. The problem was introduced by [13], the authors developed an exact algorithm based on branch-and-price method and they proposed 3 metaheuristics inspired from those proposed in [9] for the TOP. Recently, [14] proposed a bi-level algorithm based on a filter-and-fan procedure and a variable neighborhood search to solve the CTOP.

A variant of the OP is its time windows version called the orienteering problem with time windows. The problem was introduced by [15], the authors developed a tree phase heuristic to solve it. To the best of our knowledge, only one exact algorithm was developed to solve the OPTW to optimality. [16] proposed an exact algorithm based on dynamic programming. Approach methods were proposed for the OPTW and its multiple routes version called the team orienteering problem with time windows (TOPTW). [17] were among the first to proposed an approach method to solve both the OPTW and TOPTW. They developed and ant colony system metaheuristic which is able to achieve very good results on the TOPTW test instances. [18] formulated the TOPTW as a version of the tourist trip design problem (TTDP), they developed an iterated local search algorithm to solve both the OPTW and TOPTW. [19] developed a variable neighborhood search algorithm and an hybrid approach which combines a variable neighborhood search algorithm with a granular search. [20] developed two versions of a simulated annealing heuristic. For review of the OP, the reader is referred to the very good survey of [21].

A variant of the TOPTW is its constrained version called the multi-constrained team orienteering problem with time windows (MCTOPTW). This problem was introduced by [22], they formulated the problem as a version of the TTDP. [23] developed an hybrid algorithm called GRILS which combines an iterated local search (ILS) algorithm and a greedy randomized adaptive search procedure (GRASP).

In this paper, we propose a new model for the MCTOPTW in the context of distribution problems, the resulting problem is called the capacitated team orienteering problem with time windows (CTOPTW). In the CTOPTW, a set of n vertices is given each with a known demand, a profit, a service time and a time window. A set of homogeneous vehicles each of capacity Q is based at a depot to satisfy the vertices. The objective is to design a set of m feasible routes that maximizes the total collected profit such that the total travel time of each route is limited by a given time budget and the vehicle capacity constraint is respected on each route. We develop a variable neighborhood search algorithm to solve the CTOPTW. Experimental results on the test instances show that the proposed algorithm is competitive with other existing algorithms in the literature and it is able to achieve a new best known solutions on a small computational times.

The remainder of this paper is described as follows: after formulating the CTOPTW mathematically in section 2, a representation of a solution is presented in section 3. In the fourth section, the proposed algorithm is presented. In the fifth section experimental results are discussed. The last section deals with the conclusions.

II. Mathematical formulation

The CTOPTW is defined on a complete undirected graph $G = (V, E)$ where $V = \{0, 1, \dots, n, n+1\}$ is the set of vertices and E the set of edges. The set $C = \{1, \cdots, n\}$ stands for the customers. There is only one central depot that is represented by vertices 0 and $n+1$. At each vertex i is associated a known profit p_i, a demand d_i, a service time T_i and a time window $[e_i, l_i]$ ($p_0 = d_0 = T_0 = 0$). A nonnegative cost t_{ij} is associated with each edge $(i,j) \in E$ which represents the time required to travel from vertex i to vertex j. A set $K = \{1, \dots, m\}$ of m homogeneous vehicles each of capacity Q is stationed at the depot to satisfy the customers.

The CTOPTW aims to design a set of m feasible routes that maximizes the total collected profit such that:

- routes must start and end at the depot
- routes cannot start before e_0 and cannot end after l_0
- each customer is visited at most once and within its time window
- the total travel time of each route is limited by a given time budget T_{max}
- the total demand of each route cannot exceed a given vehicle capacity Q

Define the following decision variables:

$$x_{ijk} = \begin{cases} 1 & \text{if in route } k \text{ a visit to } i \text{ is followed by a visit to } j \\ 0 & \text{otherwise} \end{cases}$$

$$y_{ik} = \begin{cases} 1 & \text{if customer } i \text{ is visited in route } k \\ 0 & \text{otherwise} \end{cases}$$

π_{ik} = beginning of service at vertex i visited in route k

M: a large value (constant)

The CTOPTW can be stated mathematically as the following mixed integer problem.

$$\text{The objective function} \quad \underset{S}{\text{maximize}} \quad f(S) = \sum_{k \in K} \sum_{i \in C} p_i y_{ik}$$

subject to

$$\sum_{k \in K} \sum_{j \in C} x_{0jk} = \sum_{k \in K} \sum_{i \in C} x_{in+1k} = m$$

$$\sum_{i \in C \cup \{0\}} x_{ilk} = \sum_{j \in C \cup \{n+1\}} x_{ljk} = y_{lk}, \forall l \in C, \forall k \in K$$

$$\pi_{ik} + T_i + t_{ij} - \pi_{jk} \leq M(1 - x_{ijk}), \forall i, j \in V, \forall k \in K$$

$$\sum_{k \in K} y_{hk} \leq 1, \forall h \in C$$

$$\sum_{i \in C} d_i y_{ik} \leq Q, \forall k \in K$$

$$\sum_{i \in C \cup \{0\}} (T_i y_{ik} + \sum_{j \in C \cup \{n+1\}} t_{ij} x_{ijk}) \leq T_{max}, \forall k \in K$$

$$e_i \leq \pi_{ik} \leq l_i, \forall k \in K, \forall i \in V$$

$$x_{ijk}, y_{ik} \in \{0, 1\}, \pi_{ik} \geq 0, \forall i, j \in V, \forall k \in K$$

The objective function (line 1) is the maximization of the total collected profit. Constraint 1 (line 2) ensure that the number of routes to be designed is equal to m. Constraints 2 (line 3) ensure that if a vertex is included in a route it is preceded and followed exactly by one other vertex. Constraints 3 (line 4) guarantee the continuity of each route. Constraints 4 (line 5) ensure that each customer is visited at most once. Constraints 5 (line 6) are the vehicle capacity constraints. Constraints 6 (line 7) ensure the limited time budget of each route. Constraints 7

TABLE I
INSTANCE WITH 20 CUSTOMERS

j	x_j	y_j	p_j	d_j	e_j	l_j	T_j
0	35	35	0	0	0	230	0
1	41	49	10	5	161	171	10
2	35	17	7	5	50	60	10
3	55	45	13	5	116	126	10
4	55	20	19	5	149	159	10
5	15	30	26	5	34	44	10
6	25	30	3	10	99	109	10
7	20	50	5	10	81	91	10
8	10	43	9	10	95	105	10
9	55	60	16	10	97	107	10
10	30	60	16	10	124	134	10
11	20	65	12	15	67	77	10
12	50	35	19	15	63	73	10
13	30	25	23	15	159	169	10
14	15	10	20	15	32	42	10
15	30	5	8	15	61	71	10
16	10	20	19	5	75	85	10
17	5	30	2	5	157	167	10
18	20	40	12	5	87	97	10
19	15	60	17	5	76	86	10
20	45	65	9	5	126	136	10

TABLE II
A SOLUTION OF THE GIVEN INSTANCE

0	5	16	6	13	0	0	12	9	3	4	0

(line 8) ensure that a visit to each customer must start within its time window.

III. Solution representation

An instance of the CTOPTW is represented by a matrix. Each line represents a vertex. At each vertex j is associated a coordinate (x_j, y_j), a profit p_j, a demand d_j, a service time T_j and a time window $[e_j, l_j]$. Vertex 0 represents the depot. Table I gives an instance with 20 customers.

A solution of the CTOPTW is represented by a string of numbers each represents a customer and $2m$ zeros which represent starting and ending points (depot). A solution on the instance represented in Table I is represented in Table II. In this solution we have $m = 2$, $T_{max} = 230$ and $Q = 50$.

The first vehicle starts by servicing customer 5 and then servicing customers 16, 6 and 13. The vehicle returns to the depot after servicing customer 13. The second vehicle starts by servicing customer 12 and then servicing customers 9, 3 and returns to the depot after servicing customer 4. Table III gives the visit's details on each vertex. The following notations were considered:

a_j: corresponds to the arrival time at j
w_j: corresponds to the waiting time at j
π_j: corresponds to the beginning of service time at j
ϵ_j: corresponds to the end of service time at j

TABLE III
VISIT'S DETAILS ON EACH VERTEX

j	p_j	a_j	w_j	π_j	ϵ_j
0	0	-	-	-	-
5	26	20.6	13.4	34	44
16	19	55.1	19.9	75	85
6	3	103	0	103	113
13	23	120	39	159	169
0	0	180.1	-	-	-
0	0	-	-	-	-
12	19	15	48	63	73
9	16	98.4	0	98.4	108.4
3	13	123.4	0	123.4	133.4
4	19	158.4	0	158.4	168.4
0	0	193.4	-	-	-

IV. The proposed approach

The CTOPTW is a NP-hard problem since it generalizes the TSP which is known to be NP-hard. Then, In this paper we propose an approach method based on the variable neighborhood search concept.

Note that the variable neighborhood search (VNS) is a metaheuristic that was introduced in 1997 by [24]. The basic idea behind this metaheuristic is a systematic change of neighborhood structure during the search process. This idea seems very simple but it will be relevant to improve the search since a local optima with respect to one neighborhood structure is not necessary a local optima with respect to another neighborhood structure.

In order to do so, let N_l $(l = 1, \ldots l_{max})$ be a finite set of neighborhood structures and $N_l(S)$ be the N_l neighborhood of the solution S.

A VNS algorithm typically starts with an initial solution that is randomly generated or using a constructive heuristic. While the stopping condition is not met, which may be the maximum number of iterations allowed and/or the maximum computational time (CPU) allowed, the search continues. At each iteration a solution $\acute{S}$ is generated from the first neighborhood of the incumbent solution S. If $\acute{S}$ is better than S, the search continues with the same neighborhood structure. Otherwise, the next neighborhood structure replaces the incumbent one. The inner loop is stopped when all the neighborhood structures were used without any improvement. The pseudocode of a basic VNS algorithm is given by Algorithm 1.

The following subsections will discuss the initial solution and the neighborhood structures procedures used in the proposed VNS algorithm.

A. Initial solution procedure

For generating the initial solution for the proposed algorithm 2 constructive heuristics are developed. These heuristics are based on the sequential best insertion concept and they differ in the way how the best insertion is chosen at each iteration. In order to do so, let $R = \{i_0, i_1, \ldots, i_{q-1}, i_q\}$ be a partial route such that $i_0 = i_q = 0$.

Algorithm 1 VNS algorithm

procedure VNS(S)
  $S \leftarrow$ InitialSolution()
  determine a set of neighborhood structures
$N_l, l = 1, ..., l_{max}$
  while stopping condition is not met **do**
    $l \leftarrow 1$
    while $(l \leq l_{max})$ **do**
      $\acute{S} \leftarrow$ generate a solution from $N_l(S)$
      if $\acute{S}$ is better than S **then**
        $S \leftarrow \acute{S}$
        $l \leftarrow 1$
      else
        $l \leftarrow l + 1$
      end if
    end while
  end while
  return S ▷ the best known solution
end procedure

Each constructive heuristic inserts the unvisited customer v for which $C_1(i_{l-1}, v, i_l)$ is minimized where (i_{l-1}, i_l) is a feasible position of v in R. The cost function $C_1(i_{l-1}, v, i_l)$ is expressed for the first constructive heuristic as follows: $C_1(i_{l-1}, v, i_l) =$

$$\frac{\alpha_1.(\overbrace{t_{i_{l-1}v} + t_{vi_l} - t_{i_{l-1}i_l}}^{\theta_v} + \beta.T_v) + \alpha_2.(\pi^a_{i_l} - \pi_{i_l})}{p_v^{\lambda}} \tag{1}$$

θ_v corresponds to the travel time saved by inserting v between the two adjacent vertices i_{l-1} and i_l.
$\pi^a_{i_l}$ corresponds to the beginning of service time at i_l given that v is inserted between i_{l-1} and i_l.
For the second constructive heuristic, the cost function $C_1(i_{l-1}, v, i_l)$ is expressed as follows:
$C_1(i_{l-1}, v, i_l) =$

$$\frac{\alpha_1.(\overbrace{t_{i_{l-1}v} + t_{vi_l} - t_{i_{l-1}i_l}}^{\theta_v} + \beta.T_v) + \alpha_2.(\sum_{i_h \in R} w_{i_h} + w_v)}{p_v^{\lambda}} \tag{2}$$

$\sum_{i_h \in R} w_{i_h}$ corresponds to the total waiting time on the current route before the insertion of v.
w_v corresponds to the waiting time on v given that v is inserted between i_{l-1} and i_l.
For each constructive heuristic, the procedure continues until no more vertex with feasible insertion can be inserted in R. Then the process starts the construction of a new route. The process continues until the construction of m routes.
Note that, at each value of the 4-tuple (α_1, α_2, β, λ) is associated two feasible solutions each one is generated using one of the constructive heuristics. The following values were considered during the initial solution procedure:
$(\alpha_1, \alpha_2) \in \{(0.9, 0.1), (0.7, 0.3), (0.5, 0.5)\}$.
$\beta \in \{0.9, 0.7, 0.5, 0.3, 0.1\}$.
$\lambda \in \{2, 3, 4\}$.
The idea behind this initial solution procedure is to create a set of feasible solutions and then select the best solution (solution with the highest objective function value) as the initial solution for the proposed VNS algorithm.
An illustration of this initial solution procedure is given by Algorithm IV.1.

Algorithm IV.1: INITIAL SOLUTION $(\alpha_1, \alpha_2, \beta, \lambda)$

$List \leftarrow \emptyset$
for each $(\alpha_1, \alpha_2, \beta, \lambda)$
  do
  generate a solution S_1 based on the first heuristic
  add S_1 to $List$
  generate a solution S_2 based on the second heuristic
  add S_2 to $List$
output (the best solution of $List$)

B. Neighborhood structures procedure

In the proposed VNS algorithm, 3 neighborhood structures are used. A neighbor of a given solution S is generated using a shaking-insertion procedure. In the shaking phase, a set of vertices (customers) is removed from each route of S. The obtained solution is then completed in the insertion phase. The neighborhood structures used differ in the shaking phase. Let S be a feasible solution and $R = \{i_0, i_1, \cdots, i_{q-1}, i_q\}$ be the first route of this solution. Let $\phi(R)$ be the number of visited customers in this route. The first shaking operator called **2-positions** performs by randomly selecting two positions $i, j \in [1, \phi(R)]$ and removing the customers associated to these positions from R. The second shaking operator called **Interval** performs by randomly selecting two positions $i, j \in [1, \phi(R)]$ and removing the customer associated to each position $l \in [i, j]$ from R. The third shaking operator called **Restart** performs by randomly selecting one position $l^{'} \in [1, \phi(R)]$ and removing the customer associated to each position $l \in [1, l^{'}]$ from R. Note that, after a removal all vertices followed the removed ones are shifted towards the beginning of the route in order to ensure its continuity. For each shaking operator, the procedure is applied for each route of the other $m-1$ routes of S.
An illustration of these shaking operators on the solution S given by Table IV is given by Tables V, VI and VII. By applying the 2-positions shaking operator to S, customers 8 and 2 have been removed. By applying the Interval operator to S, customers 15, 11, 9, 6 and 4 have been removed. By applying the Restart shaking operator to S, customers 13, 7, 8, 15, 11, 9 and 6 have been removed from S. Note that, the output of each shaking operator on a given solution S is an incomplete solution S_a that needs to be completed. It is easy to see that if the insertion of each unvisited customer in S_a is evaluated using the cost function given by Equation (1) or (2), the set of customers just removed in the shaking phase

TABLE IV
SOLUTION WITH $m = 1$

0	13	7	8	15	11	9	6	4	2	1	0

TABLE V
AN ILLUSTRATION OF THE 2-POSITIONS OPERATOR

0	13	7	15	11	9	6	4	1	0

TABLE VI
AN ILLUSTRATION OF THE INTERVAL OPERATOR

0	13	7	8	2	1	0

TABLE VII
AN ILLUSTRATION OF THE RESTART OPERATOR

0	4	2	1	0

will be inserted which means that the search will be stuck in S. One way to do so is to insert each unvisited customer at its first feasible place in S_a. This idea will give chance to the unvisited customers in S to be part of S_a in order to improve the search.

The stopping condition used is the maximum number of iterations between two improvements which is set to 50. The value of this parameter was determined using a set of experiments on a subset of test instances which is randomly selected. The value 50 realizes the best trade-off between the quality of the obtained solutions and the amount of computational time needed to achieve these solutions. An illustration of the proposed VNS algorithm is given by Algorithm 2. The idea behind the proposed VNS algorithm is to slightly perturb the incumbent solution by applying the 2-positions shaking operator. If this shaking operator is not able to help the search to escape a local optima, the Interval shaking operator is then applied. If this perturbation is not sufficient, the incumbent solution is then strongly perturbed by applying the Restart operator. Note that, if the position $l^{'} = \phi(R)$, the insertion phase on the shaking's output will behave like a new construction.

Algorithm 2 The proposed VNS algorithm

```
procedure VNS(S)
    N1 2-positions neighborhood structure
    N2 Interval neighborhood structure
    N3 Restart neighborhood structure
    S ← initial solution ()
    I ← 0
    while I < 50 do
        p ← 1
        while (p ≤ 3) do
            Ś ← generate a solution from Np(S)
            if f(Ś) > f(S) then
                S ← Ś
                p ← 1
            else
                p ← p + 1
            end if
        end while
        I ← I + 1
    end while
    return S                          ▷ the best known solution
end procedure
```

V. COMPUTATIONAL RESULTS

The proposed VNS algorithm was coded in Java and run on a personal computer Intel(R) with 2.1 GHz processor and 4GB of RAM memory. The performance of the proposed algorithm is compared with that of GRILS algorithm of [23] on the test instances created by [22]. These test instances are inspired from those created by [16] for the OPTW. These test instances are themselves inspired from the test instances c100, r100 and rc100 created by [25] for the VRPTW and the test instances pr01-pr10 created by [26] for the periodic multi-depot vehicle routing problem. The number of customers on the Solomon's test instances (c100, r100 and rc100) is 100 and the number of customers on the Cordeau's test instances (pr01-pr10) ranges from 48 to 288.

Tables VIII to XI compare the profits achieved by GRILS of [23] and the proposed VNS algorithm. The first group of columns denotes the instance's name, the time budget and the vehicle capacity respectively. The fourth column and the fifth column are the average profit of ten runs achieved by GRILS and the average computational time respectively. Columns six and seven are the profit achieved by the proposed VNS and its computational time respectively. All computational times are measured in seconds (s). Bold numbers indicate that the profit achieved by of the proposed VNS algorithm is equal or greater than that achieved by GRILS algorithm.

It can be seen from tables VIII to XI that:

On the Solomon's test instances with $m = 1$, the average gap of VNS to GRILS on the c100, r100 and rc100 test instances is respectively 0.4%, 0.5% and 2.6%. The proposed VNS algorithm outperforms GRILS on 8 test instances over 29. The average CPU of VNS and GRILS is the same (0.1 seconds). The average gap of VNS to GRILS on the Cordeau's instances (pr01-pr08 and pr10) is 0.6% and the average gap of VNS to GRILS on the Cordeau's instances (pr01-pr10) is 2.5%. The average CPU of VNS and GRILS is 1.4 seconds and 0.4 seconds respectively.

With $m = 2$, the proposed VNS performs on average less than GRILS on the Solomon's test instances. The proposed VNS algorithm outperforms GRILS on 8 test instances over 29. On the Cordeau's instances, the proposed VNS dominates GRILS. The average gap of VNS to GRILS on the Cordeau's test instances (pr01-pr08 and pr10) is -2.0% and the average gap of VNS to GRILS on the Cordeau's test instances (pr01-pr10) is -1.7%. The average CPU of VNS and GRILS is 2.6 seconds and 1.2 seconds respectively. The proposed VNS algorithm performs better on the large test instances pr04, pr05, pr06 and pr10 with a number of 192, 240, 288 and 240 customers respectively. The proposed algorithm provides new best known solutions on these test instances. As a consequence we can conclude that the proposed VNS algorithm can be able to efficiently solve real-life CTOPTW test instances.

VI. Conclusions

In this paper we have presented a new model for the multi-constraint team orienteering problem with time windows in the context of distribution problems, the resulting problem is called the capacitated team orienteering problem with time windows. This problem is a variant of the well known CTOP. As a consequence it can serve as a model for many real-world distribution problems (i.e grocery distribution, gasoline delivery truck, etc). We have developed a variable neighborhood search algorithm to solve the CTOPTW. Computational results on the test instances have shown that the proposed algorithm is competitive with GRILS algorithm and it is able to provide new best known solutions on a large and hard test instances.

Acknowledgment

The authors would like to thank the two anonymous referees who helped them to improve a first version of this paper.

References

[1] G. Dantzig and J. Ramser, "The truck dispatching problem," *Management science*, vol. 6, pp. 80–91, 1959.

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

[3] G. Laporte and S. Martello, "The selective travelling salesman problem," *Discrete Applied Mathematics*, vol. 26, pp. 193–207, 1990.

[4] M. Fischetti, J. S. Gonzales, and P. Toth, "Solving the orienteering problem through branch-and-cut," *INFORMS Journal on Computing*, vol. 10, no. 2, pp. 133–148, 1998.

[5] I. Chao, B. Golden, and E. Wassil, "A fast and effective heuristic for the orienteering problem," *European Journal of Operational Research*, vol. 88, pp. 475–489, 1996b.

[6] M. Tasgetiren, "A genetic algorithm with an adaptive penalty function for the orienteering problem," *Journal of Economic and Social Research*, vol. 4, no. 2, pp. 1–26, 2001.

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

[8] S. Butt and D. Ryan, "An optimal solution procedure for the multiple tour maximum collection problem using column generation," *Computers and Operations Research*, vol. 26, no. 4, pp. 427–441, 2006.

[9] C. Archetti, A. Hertz, and M. G. Speranza, "Metaheuristics for the team orienteering problem," *Journal of Heuristics*, vol. 13, no. 1, pp. 49–76, 2007.

[10] H. Tang and E. Hooks, "A tabu search heuristic for the team orienteering problem," *Computers and Operations Research*, vol. 32, no. 6, pp. 1379–1407, 2005.

[11] P. Vansteenwegen, W. Souffriau, G. V. Berghe, and D. V. Oudheusden, "A guided local search for the team orienteering problem," *European Journal of Operational Research*, vol. 196, pp. 118–127, 2009.

[12] S.-W. Lin, "Solving the team orienteering problem using effective multi-start simulated annealing," *Applied Soft Computing*, vol. 13, no. 2, pp. 1064–1073, 2013.

[13] C. Archetti, D. Feillet, A. Hertz, and M. G. Speranza, "The capacitated team orienteering problem and profitable tour problem," *Journal of the Operational Research Society*, vol. 60, no. 6, pp. 831–842, 2009.

[14] C. Tarantilis, F. Stavropoulou, and P. Repoussis, "The capacitated team orienteering problem: A bi-level filter-and-fan method," *European Journal of Operational Research*, vol. 224, no. 1, pp. 65–78, 2013.

[15] M. Kantor and M. Rosenwein, "The orienteering problem with time windows," *Journal of Operational Research Society*, vol. 43, pp. 629–635, 1992.

[16] G. Righini and M. Salani, "Decremental state space relaxation strategies and initialization heuristics for solving the orienteering problem with time windows with dynamic programming," *Computers and Operations Research*, vol. 36, no. 4, pp. 1191–1203, 2009.

[17] R. Montemanni and L. M. Gambardella, "An ant colony system for team orienteering problem with time windows," *Foundations of Computing and Decision Sciences*, vol. 34, no. 4, pp. 287–306, 2009.

[18] P. Vansteenwegen, W. Souffriau, G. V. Berghe, and D. V. Oudheusden, "Iterated local search for the team orienteering problem with time windows," *Computers and Operations Research*, vol. 36, no. 12, pp. 3281–3290, 2009.

[19] N. Labadie, R. Mansini, J. Melechovsky, and R. Calvo, "The team orienteering problem with time windows: An lp-based granular variable neighborhood search," *European Journal of Operational Research*, vol. 220, no. 1, pp. 15–27, 2012.

[20] S.-W. Lin and V. F. Yu, "A simulated annealing heuristic for the team orienteering problem with time windows," *European Journal of Operational Research*, vol. 217, no. 1, pp. 94–107, 2012.

[21] P. Vansteenwegen, W. Souffriau, and D. V. Oudheusden, "The orienteering problem: A survey," *European Journal of Operational Research*, vol. 209, no. 1, pp. 1–10, 2011.

[22] A. Garcia, P. Vansteenwegen, W. Souffriau, D. Arbelaitz, and M. Linaza, "Solving multi constrained team orienteering problems to generate tourist routes," 2009.

[23] W. Souffriau, P. Vansteenwegen, G. V. Berghe, and D. V. Oudheusden, "The multiconstraint team orienteering problem with multiple time windows," *Transportation Science*, vol. 47, no. 1, pp. 53–63, 2013.

[24] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers and Operations Research*, vol. 24, pp. 1097–1100, 1997.

[25] M. Solomon, "Algorithms for the vehicle routing and scheduling problem with time window constraints," *Operations Research*, vol. 35, pp. 254–265, 1987.

[26] J. Cordeau, M. Gendreau, and G. Laporte, "A tabu search heuristic for periodic and multi-depot vehicle routing problems," *Networks*, vol. 30, pp. 105–119, 1997.

TABLE VIII
RESULTS ON THE SOLOMON'S TEST INSTANCES WITH $m = 1$

Instance	T_{max}	Q	GRILS	CPU(s)	VNS	CPU(s)
c101	1236	100	320.0	0.1	**320**	0.1
c102	1236	100	360.0	0.2	**360**	0.1
c103	1236	100	393.0	0.3	390	0.1
c104	1236	105	409.0	0.3	**420**	0.0
c105	1236	100	340.0	0.1	**340**	0.1
c106	1236	105	340.0	0.1	**340**	0.0
c107	1236	115	370.0	0.1	360	0.0
c108	1236	115	370.0	0.2	360	0.0
c109	1236	105	380.0	0.2	**380**	0.0
r101	230	105	198.0	0.1	**198**	0.0
r102	230	90	283.8	0.1	**286**	0.0
r103	230	95	289.8	0.2	286	0.0
r104	230	90	295.2	0.2	**297**	0.0
r105	230	100	247.0	0.1	**247**	0.0
r106	230	95	290.6	0.2	**293**	0.0
r107	230	140	292.2	0.2	288	0.0
r108	230	140	300.5	0.2	**303**	0.1
r109	230	120	277.0	0.1	276	0.0
r110	230	135	278.7	0.1	**281**	0.0
r111	230	110	296.4	0.2	283	0.0
r112	230	115	292.9	0.2	287	0.0
rc101	240	75	219.0	0.1	216	0.0
rc102	240	80	260.4	0.1	259	0.0
rc103	240	80	259.2	0.1	236	0.0
rc104	240	100	295.5	0.1	271	0.0
rc105	240	90	240.0	0.1	**244**	0.0
rc106	240	95	247.2	0.1	244	0.0
rc107	240	100	272.3	0.1	266	0.1
rc108	240	100	274.9	0.1	**276**	0.1

TABLE IX
RESULTS ON THE CORDEAU'S TEST INSTANCES WITH $m = 1$

Instance	T_{max}	Q	GRILS	CPU(s)	VNS	CPU(s)
pr01	1000	195	306.8	0.1	280	0.1
pr02	1000	230	381.5	0.2	380	0.5
pr03	1000	180	385.1	0.3	377	0.7
pr04	1000	220	442.2	0.5	**481**	1.3
pr05	1000	300	522.5	0.7	521	2.9
pr06	1000	260	502.1	0.7	498	2.9
pr07	1000	165	296.0	0.1	285	0.2
pr08	1000	215	444.1	0.3	**455**	1.1
pr09	1000	300	448.0	0.5	364	1.3
pr10	1000	235	509.8	0.8	503	2.8

TABLE X
RESULTS ON THE SOLOMON'S TEST INSTANCES WITH $m = 2$

Instance	T_{max}	Q	GRILS	CPU(s)	VNS	CPU(s)
c101	1236	100	590.0	0.4	580	0.5
c102	1236	100	645.0	0.5	**650**	0.4
c103	1236	105	690.0	0.7	680	0.4
c104	1236	110	727.0	0.8	**750**	0.5
c105	1236	105	640.0	0.4	**640**	0.2
c106	1236	100	620.0	0.4	**620**	0.2
c107	1236	90	668.0	0.5	640	0.2
c108	1236	110	675.0	0.5	**680**	0.5
c109	1236	100	699.0	0.6	**700**	0.5
r101	230	70	341.7	0.3	324	0.0
r102	230	90	500.8	0.4	481	0.2
r103	230	130	508.6	0.5	**513**	0.4
r104	230	90	521.7	0.5	512	0.5
r105	230	75	437.3	0.3	411	0.2
r106	230	95	514.4	0.4	503	0.2
r107	230	75	519.9	0.5	478	0.2
r108	230	90	530.5	0.5	508	0.3
r109	230	105	486.3	0.4	474	0.3
r110	230	125	494.0	0.4	**500**	0.5
r111	230	125	532.4	0.5	523	0.5
r112	230	120	512.1	0.4	511	0.5
rc101	240	60	425.5	0.3	385	0.2
rc102	240	70	490.4	0.4	453	0.2
rc103	240	90	500.8	0.4	**507**	0.3
rc104	240	120	536.6	0.4	**551**	0.5
rc105	240	90	465.1	0.3	454	0.2
rc106	240	85	466.5	0.3	443	0.3
rc107	240	80	497.4	0.3	485	0.2
rc108	240	85	514.4	0.4	514	0.2

TABLE XI
RESULTS ON THE CORDEAU'S TEST INSTANCES WITH $m = 2$

Instance	T_{max}	Q	GRILS	CPU(s)	VNS	CPU(s)
pr01	1000	195	483.4	0.3	456	0.3
pr02	1000	230	654.9	0.6	651	0.9
pr03	1000	180	686.2	0.8	676	1.4
pr04	1000	220	827.4	1.3	**847**	2.7
pr05	1000	300	938.0	2.0	**1040**	4.7
pr06	1000	260	868.3	2.1	**976**	4.7
pr07	1000	165	549.4	0.4	546	4.1
pr08	1000	215	769.5	0.9	**771**	1.6
pr09	1000	300	786.4	1.4	763	2.9
pr10	1000	235	925.3	2.3	**951**	2.8