European Journal of Operational Research 220 (2012) 15–27

Contents lists available at SciVerse ScienceDirect

European Journal of Operational Research

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

Discrete Optimization

The Team Orienteering Problem with Time Windows: An LP-based Granular Variable Neighborhood Search

Nacima Labadie [b], Renata Mansini [a], Jan Melechovský [b,d], Roberto Wolfler Calvo [c,*]

[a] University of Brescia, Department of Information Engineering, Italy
[b] University of Technology of Troyes, ICD-LOSI, France
[c] LIPN (UMR 7030), 99, av. Jean-Baptiste Cement, 93430 Villetaneuse, France
[d] University of Economics, Prague, Department of Econometrics, Czech Republic

ARTICLE INFO

Article history:
Received 4 January 2011
Accepted 17 January 2012
Available online 25 January 2012

Keywords:
Team Orienteering Problem
Time windows
Variable Neighborhood Search
LP-based granularity

ABSTRACT

The Team Orienteering Problem (TOP) is a known NP-hard problem that typically arises in vehicle routing and production scheduling contexts. In this paper we introduce a new solution method to solve the TOP with hard Time Window constraints (TOPTW). We propose a Variable Neighborhood Search (VNS) procedure based on the idea of exploring, most of the time, granular instead of complete neighborhoods in order to improve the algorithm's efficiency without loosing effectiveness. The method provides a general way to deal with granularity for those routing problems based on profits and complicated by time constraints. Extensive computational results are reported on standard benchmark instances. Performance of the proposed algorithm is compared to optimal solution values, when available, or to best known solution values obtained by state-of-the-art algorithms. The method comes out to be, on average, quite effective allowing to improve the best know values for 25 test instances.

1. Introduction

Orienteering is a sport in which a competitor has to determine a path from a starting point to a final destination, visiting control points along the path. A score (profit) is associated to each control point, and for each pair of control points a travel cost is specified. The problem looks for a set of control points to be visited, so that the total score/profit is maximized subject to a constraint on the total travel cost. The problem was first introduced as Orienteering Problem (OP) by Tsiligrides [31] and then found different practical applications in vehicle routing and production scheduling (see [10,14]). When starting point and final destination are the same, the problem looks for a subtour (instead of a path) and is called Selective Traveling Salesman Problem (TSP) [18].

Many other routing problems deal with a positive score/profit associated to each visiting point. This is the case of the *quota* TSP that differs from the OP since the objective function minimizes the traveling costs while guaranteeing a minimum threshold on the total score of the visited points (target quota) [9]. The OP also differs from the Prize-Collecting TSP which is the same as the *quota* TSP, but with the additional complication of visiting points having penalties for not being on the tour. Interested readers are referred to Feillet et al. [8] and to Vansteenwegen et al. [35] for a survey on the traveling salesman problem with profits and on the Orienteering Problem, respectively.

Several variants of the OP are dealt with in the literature, one of them takes time window constraints into account (OPTW). Time window constraints are motivated by different practical situations and typically arise in those routing problems where each customer/location has to be visited within a predefined time interval specified by an earliest and a latest time into which the service has to start. Examples of applications are bank and postal deliveries, industrial refuse collection, dial-a-ride services, and school bus routing (see Solomon and Desrosiers [25] for a survey on time constrained routing problems). Very few contributions can be found to the solution of the OPTW. The first one is due to Kantor and Rosenwein [12] who propose a heuristic algorithm based on an exhaustive exploration of the feasible solutions space. In the technical report by Mansini et al. [19], the authors analyze a Granular Variable Neighborhood Search, and provide an upper bound for the problem based on the solution of a 0–1 knapsack problem. The proposed method represent a very initial version of the Granular Variable Neighborhood Search discussed in the present paper. More recently, Righini and Salani [24] present an algorithm based on dynamic programming and provide optimal solutions for short scheduling horizon instances.

Another extension of the OP consider a fixed number m ($m > 1$) of tours each one satisfying a given time (cost) threshold. This problem was first formulated by Chao et al. [5] as the Team Orienteering Problem (TOP). The authors apply a two phase heuristic algorithm

* Corresponding author.
E-mail addresses: nacima.labadie@utt.fr (N. Labadie), rmansini@ing.unibs.it (R. Mansini), jan.melechovsky@utt.fr, jan.melechov@seznam.cz (J. Melechovský), roberto.wolfler@lipn.univ-paris13.fr (R. Wolfler Calvo).

0377-2217/$ - see front matter © 2012 Elsevier B.V. All rights reserved.
doi:10.1016/j.ejor.2012.01.030

that they originally designed for the OP [4]. First a set of feasible paths is determined within a feasible ellipse encompassing all paths that satisfy the traveling cost limit. In the second phase, the solution is improved by performing nodes insertion and nodes exchange moves. Other approaches to the TOP presented in the literature are represented by meta-heuristic methods. Tang and Miller-Hooks [27] solve the problem using a tabu search algorithm. An ant colony optimization approach is proposed by Ke et al. [13]. The authors develop four methods working within an ant colony framework. A memetic algorithm is presented by Bouly et al. [1]. Souffriau et al. [26] develop a path relinking algorithm combined with a greedy randomized adaptive search procedure. The same authors also present a guided local search and a skewed variable neighborhood search for the TOP [32,34].

Recently, some attention has been devoted to the Team Orienteering Problem with Time Windows (TOPTW). An ant colony algorithm (ACS) is proposed by Montemanni and Gambardella [22] and an iterated local search (ILS) is developed by Vansteenwegen et al. [33]. An exact approach (a branch and price) is presented by Boussier et al. [2] for the TOP and the selective vehicle routing problem with time windows (SVRPTW). The latter differs from the TOPTW in some additional constraints. In the SVRPTW, each tour must satisfy a capacity constraint, a given time limit and the time window of each customer. Tricoire et al. [29] introduce the multi-period orienteering problem with multiple time windows (MuPOPTW). The problem generalizes the TOPTW considering a multi-period horizon. Hence, each customer is associated with a fixed number of services along the horizon and with different time windows. Finally, a hybridized evolutionary local search algorithm for the TOPTW is proposed by Labadie et al. [16,17]. More recently, Montemanni et al. [23] provided new results for the TOPTW.

In all problems with profits, time windows (when included) are assumed to be hard constraints. This means that early arrivals lead to waiting times, whereas late arrivals cause infeasibility. On the other hand, soft time windows allow to violate the constraints but with additional penalty costs. This paper studies the Team OP with *hard* Time Window constraints (TOPTW). The problem arises in different application contexts from distribution logistics to tourism. A Variable Neighborhood Search (VNS) scheme hybridized with the idea of exploring, most of the time, granular instead of complete neighborhoods is developed. More specifically, the algorithm uses information provided by the optimal solution of a related LP-problem to construct granular neighborhoods. The aim is to increase the algorithm's efficiency without loosing its effectiveness. Granularity concept was first introduced by Toth and Vigo [28] associated to a Tabu Search algorithm for the VRP. The resulting approach called Granular Tabu Search is based on the idea that longer edges usually have a lower likelihood to belong to optimal solutions in routing problems. By removing *a priori* all edges with length exceeding a given threshold, the algorithm can avoid to visit several non promising solutions in the search process. The authors suggest to adjust dynamically the threshold according to the objective function evolution during the search.

In this paper we introduce a more general concept of granularity that includes time constraints and profits in addition to pure distances. In time constrained routing problems, the selection of shorter traveling distances may be misleading in constructing good quality solutions. Indeed, points located very close may have time windows with strongly different starting times so that visiting them sequentially may be a penalizing choice. Moreover, the solutions quality for the TOP depends on profits. Thus, to define granularity, we use the dual optimal solutions of a related LP-problem that takes into account all the described problem features. More specifically, the reduced costs of the optimal solution of an assignment problem are used. Arc costs of the assignment problem are defined using distances as well as times and profits. The lower the reduced cost associated to an arc, the higher the likelihood it will belong to a good quality solution. We apply this granularity concept to the local search neighborhoods of a VNS approach. Nevertheless, the idea is general enough and can be easily applied also to other meta-heuristic approaches. The proposed granular VNS (GVNS) has been tested on known standard instances for the TOPTW. The method has shown a good average performance in a reasonable amount of time and has allowed to improve the best known values of 25 benchmark instances.

The remainder of this paper is organized as follows. In Section 2 the mathematical programming formulation of the Team Orienteering Problem with Time Windows is provided. Section 3 is devoted to the solution algorithms. We start with the description of a simple constructive heuristic used to provide an initial feasible solution. Then we introduce the Variable Neighborhood Search approach proposed to solve the problem. Finally, we describe its granular variant. In Section 4 the proposed algorithm is tested on different sets of standard benchmark instances. Its performance is compared to optimal solution values (when available) or to best known solution values found by state-of-the-art algorithms. Section 5 draws some conclusions and states some possible future developments.

2. The mathematical model

The Team Orienteering Problem with Time Windows (TOPTW) can be described as follows. Let us consider a set of visiting points $N = \{1,2,\ldots,n\}$ plus a depot indexed by 0. For model formulation, let us define a copy of the depot indexed $n+1$. Let $G = (N \cup \{0,n+1\},A)$ be a directed graph where $N \cup \{0,n+1\}$ is the set of nodes and A is the set of arcs. A positive integer score (profit) p_i is associated with each node $i \in N$, whereas $p_0 = p_{n+1} = 0$. Each node $i \in N$ has a time window $[e_i,l_i]$, where e_i is the earliest time and l_i the latest time allowed for starting service at node i, whereas we indicate as e_0 and l_0 the earliest leaving time and the latest arrival time of each vehicle to the depot and the dummy node $n+1$. We define as T the maximum total travel time allowed to complete a tour. W.l.o.g. we assume that $e_{n+1} = e_0 = 0$ and $l_{n+1} = l_0 = T$.

Let t_{ij} be the nonnegative travel time associated to each arc $(i,j) \in A$. We assume that the service time at node i is included in the time value t_{ij} for all $i \in N \cup \{0\}$. If, for some $i \in N$ and $j \in N \cup \{n+1\}$, $i \neq j$, $e_i + t_{ij} > l_j$ we assume $t_{ij} = \infty$. We set also $t_{i,n+1} = t_{i0}$ for all $i \in N$ and $t_{0,n+1} = \infty$. The problem looks for m tours with maximum total profit. Each tour starts at time e_0 at the origin depot 0, visits a subset of nodes exactly once within their time windows and ends at the dummy depot $n+1$ before the time limit $l_{n+1} = l_0$ (i.e. total travel time of each tour does not exceed T).

Let y_i^v, $i \in N \cup \{0,n+1\}$, be a binary variable set to 1 if node i is selected in the tour v and to 0 otherwise. We define as x_{ij}^v, $(i,j) \in A$, the set of binary variables such that x_{ij}^v is equal to 1 if arc (i,j) is selected in the tour v and 0 otherwise. Let a_i^v, $i \in N \cup \{0\}$, be a time variable specifying when the service starts at node i in tour v, whereas variable a_{n+1}^v provides the arrival time of vehicle v at the node $n+1$. The TOPTW mathematical formulation is as follows:

$$(TOPTW) \quad \max \sum_{v=1}^{m} \sum_{i \in N} p_i y_i^v \tag{1}$$

$$\sum_{j \in N \cup \{n+1\}} x_{ij}^v = y_i^v \qquad i \in N,\ v = 1,\ldots,m \tag{2}$$

$$\sum_{i \in N \cup \{0\}} x_{ij}^v - \sum_{i \in N \cup \{n+1\}} x_{ji}^v = 0 \quad j \in N,\ v = 1,\ldots,m \tag{3}$$

$$\sum_{v=1}^{m} \sum_{j \in N} x_{0j}^v = \sum_{v=1}^{m} \sum_{j \in N} x_{j,n+1}^v = \sum_{v=1}^{m} y_0^v = \sum_{v=1}^{m} y_{n+1}^v = m \tag{4}$$

$$\sum_{v=1}^{m} y_i^v \leqslant 1 \quad i \in N, \tag{5}$$

$$x_{ij}^v \left(a_i^v + t_{ij} - a_j^v\right) \leqslant 0 \quad (i,j) \in A, \ v = 1, \ldots, m \tag{6}$$

$$e_i y_i^v \leqslant a_i^v \leqslant l_i y_i^v \quad i \in N \cup \{0, n+1\}, \ v = 1, \ldots, m \tag{7}$$

$$x_{ij}^v \in \{0,1\} \quad (i,j) \in A, \ v = 1, \ldots, m \tag{8}$$

$$y_i^v \in \{0,1\} \quad i \in N, \ v = 1, \ldots, m \tag{9}$$

$$a_i^v \in R_+ \quad i \in N \cup \{0, n+1\}, \ v = 1, \ldots, m \tag{10}$$

The objective function (1) establishes the maximization of the total profit of the selected nodes. Constraints (2) and (3) define the assignment constraints on the subset of selected nodes. Constraint (4) guarantees that all tours start and end at the depot. Constraints (5) determine that each location is visited at most once, whereas constraints (6) and (7) ensure feasibility of the time schedule. In particular, constraints (6) imply that if the arc (i,j) is visited, then the starting service time at node j has to be greater than or equal to the time the service started at node i plus the travel time from node i to node j. Constraints (7) represent time windows constraints, implying that, for each node i, $i \in N \cup \{0, n+1\}$, selected in the solution, the service has to start between its earliest and its latest time, e_i and l_i, respectively. Since constraints (7) include conditions on the depot for which $l_0 = l_{n+1} = T$, the TOPTW formulation does not require the introduction of a constraint on the total travel time T. Moreover, subtours are eliminated by constraints (6). By means of a large constant M, such constraints can be linearized as follows $a_i^v + t_{ij} - a_j^v \leqslant M\left(1 - x_{ij}^v\right)$. They represent a generalization of the classic TSP subtour elimination constraints proposed by Miller, Tucker and Zemlin [21]. Since $a_i^v + t_{ij} - a_j^v \leqslant l_i + t_{ij} - e_j \leqslant l_i + t_{ij}$, M can be replaced by $M_{ij} = l_i + t_{ij}$ for each $(i,j) \in A$ such that $e_i + t_{ij} \leqslant l_j$. This choice is consistent with setting M as small as possible. Finally, constraints (8) and (9) are binary conditions, whereas the start of the service time a_i^v can in general take nonnegative real values (10). Similar mathematical formulations of the TOPTW can be found in [22,33].

Denoting $P = \sum_{i \in N} p_i$ and introducing variables $x_{ii}^v = 1 - y_i^v$, $i \in N$, $v = 1, \ldots, m$, as self-loop arcs, the objective function and the constraints (2) and (3) in formulation (TOPTW) can be rewritten as an assignment problem as follows:

$$(TOPTW1) \quad \max \quad \left(P - \sum_{v=1}^{m} \sum_{i \in N} p_i x_{ii}^v\right) \tag{11}$$

$$\sum_{j \in N \cup \{n+1\}} x_{ij}^v = 1 \quad i \in N, \ v = 1, \ldots, m \tag{12}$$

$$\sum_{i \in N \cup \{0\}} x_{ij}^v = 1 \quad j \in N, \ v = 1, \ldots, m \tag{13}$$

If the loop arc x_{ii} is selected, then node i is not visited in the optimal solution. We refer to this formulation as to model (TOPTW1).

3. Solution algorithms

The present section is devoted to the description of the main steps of implemented algorithms. We start by describing a simple constructive heuristic used to build an initial feasible solution to the problem (*insertion heuristic*). Then Variable Neighborhood Search (VNS) approach and its local search procedure are described. Finally, the VNS variant using the LP-based granularity (Granular VNS) is discussed.

3.1. Insertion heuristic

The insertion heuristic described in [16] is used to build an initial feasible solution. In every iteration, the procedure evaluates all feasible insertions of unvisited nodes and selects the node representing the best insertion. An insertion is evaluated with the following criterion. Let i be some node in a tour and let r be a node candidate for the insertion. Let S_r and g_r denote the set of nodes reachable from node r (i.e. the set of nodes s such that $a_r + t_{rs} \leqslant l_s$) and its cardinality, respectively. Then, the best insertion is determined by the pair (i,r) for which ρ_{ir} is maximum:

$$\rho_{ir} = g_r \cdot \frac{p_r}{\Delta_{ir}} \tag{14}$$

where p_r is the profit of node r and $\Delta_{ir} = t_{ir} + w_r + t_{rj} - t_{ij}$ measures the possible delay if node r is inserted between node i and its successor j in the tour, with w_r denoting the waiting time in r.

At the beginning m tours are initialized with the first m nodes for which ρ_{0r} is maximum. Then, the remaining unvisited nodes are considered to complete the solution. The procedure stops when no more feasible move can be found.

3.2. The Variable Neighborhood Search

VNS is a simple approach based on the idea of changing neighborhood every time the algorithm reaches a local optimum. It is mainly composed of a local search tool, a sequence of neighborhoods and a set of rules defining the selection of the current neighborhood according to the search status. A detailed description of the VNS is out of the scope of this paper. Interested readers are referred to [20,11]. In the following, we first describe the basic VNS approach developed for the TOPTW and then we provide details on its granular variant.

Let $N_k(\cdot)$, $k = 1, \ldots, k_{max}$, be a finite sequence of neighborhoods depending on parameter k. Given an incumbent solution x^*, its k–neighborhood $N_k(x^*)$ consists of all the solutions which can be obtained by randomly removing k consecutive nodes from x^*. The procedure starts by setting $k = 1$. A solution $x' \in N_k(x^*)$ is then obtained using the procedure *Random*(x^*,k) by removing a random sequence of k consecutive nodes from solution x^* and replacing it by a sequence of unvisited nodes. This sequence of nodes is determined using a simple heuristic. Let i and j denote the nodes between which the sequence should be inserted and u the current predecessor of the node that will be inserted in the tour. First, the procedure initializes $u = i$. In the next step, it searches for a node v (among all unvisited nodes), for which the insertion between u and j is feasible and the ratio $\rho' = g_v p_v/(w_v + t_{u,v})$ is maximum. Recall that g_v denotes the number of nodes reachable from v, p_v is the profit of v, $t_{u,v}$ is the travel time from u to v and w_v denotes the waiting time in v when traveling from u. The node v is inserted into the sequence and the search starts again with $u = v$. The search continues until the sequence cannot be further extended. Note that the routine removes a sequence of k nodes and replaces it with a new sequence of no predefined length.

A local search algorithm is then applied to x' (procedure *LocalSearch*). If the resulting solution x'' is better than x^*, it becomes the new incumbent solution and the search is restarted with $k = 1$. Otherwise, the value of k is increased and a different neighborhood is analyzed. The VNS approach stops as soon as the number of iterations meets a predefined value *MaxIter* or no solution value improvement has been obtained after *ItImp* iterations (external **while loop**). Algorithm 1 provides the pseudo-code description of the proposed VNS approach.

Algorithm 1. Pseudo-code of the implemented VNS algorithm.

```
     Input: an initial solution $x_I$ with value $z(x_I)$
     Output: a feasible solution $x^*$ with value $z(x^*)$
     Parameters $MaxIter$, $ItImp$, $k_{max}$, $q_{max}$
1    begin
2        $x^* := x_I$, $z(x^*) := z(x_I)$;
3        $it := imp := 0$;
4        $k := 1$;
5        while ($it < MaxIter$) and ($imp < ItImp$) do
6           $x' := Random(x^*, k)$;
7           $x'' := LocalSearch(x', q_{max})$;
8           if $z(x'') > z(x^*)$ then
9              $x^* := x''$ and $z(x^*) := z(x'')$;
10             $imp := 0$ and $k := 1$;
11          else
12             $k := \min(k+1, k_{max}+1)$;
13             if $k > k_{max}$ then
14                $k := 1$;
15             end if
16             $imp := imp + 1$;
17          end if
18          $it := it + 1$
19       end while
20   end
```

The definition of a suitable neighborhood structure is one of the most critical issues when designing local search mechanisms. In the following, we provide the description of the procedure *LocalSearch*( ), the definition of the neighborhoods it makes use of and how they are explored.

3.2.1. Local search procedure

The local search procedure explores two different neighborhoods. The first neighborhood contains classical routing moves:

1. 2-opt move: it removes and replaces two arcs in a tour and reorders nodes,
2. Or-opt move: it relocates one node,
3. 2-opt* move: it interchanges two sub-paths between two tours,
4. Swap move: it swaps two nodes.

The moves are considered subsequently and the first feasible move is performed. The search stops when no improving move can be found. The time window constraints require a feasibility check before a move is performed. However, the feasibility of each move can be checked in $O(1)$ time, if the maximum feasible delay of each tour is kept in memory. See Kindervater and Savelsbergh [15] for details. The aim of the exploration of this neighborhood is to reduce the total travel time of the solution so to possibly gain some space for the insertion of other nodes.

The second neighborhood tries to fulfill the possibly available space in the solution by replacing short chain of nodes. It is defined by removing q consecutive nodes from a tour and replacing them by a sequence of nodes that are not yet in the solution. The determination of the optimal entering sequence is done exactly using dynamic programming. The problem of finding such an optimal sequence is *NP*-hard being equivalent to the problem of finding a single tour OPTW. However, the presence of time windows reduces the number of feasible sequences so that the exact evaluation of the entering sequence can be implemented in a reasonable amount of time. The way the second neighborhood is explored is illustrated in Algorithm 2. Starting with $q = 1$, the procedure considers all sequences X_{Out} of q consecutive nodes and determines a sequence X_{In} of entering nodes that fits the available time slot. If the sum of profits in X_{In} is greater than the sum of profits in X_{Out} then the move is admissible. Finally, the move with largest difference between the inserted and the removed profit is performed. The procedure stops when no more admissible move can be found. Due to computational efficiency reasons, the number q of removed nodes is limited by a maximum value q_{max}.

Given a starting solution x', the procedure *LocalSearch* first tries to reduce the total travel time of each tour in x' using the first neighborhood. Then it searches for moves increasing the total collected profit by exploring the second neighborhood. If an improvement is found, it switches to the first neighborhood again. The procedure stops when no improving move can be found.

Algorithm 2. Exploration of the second neighborhood of the local search.

```
     Input: a solution $x$ with value $z(x)$
     Output: a solution $x'$ with value $z(x')$
     Parameters: $q_{max}$
1    begin
2        repeat
3           $Found := False$;
4           $Best := 0$;
5           $q := 1$;
6           while $q \leqslant q_{max}$ do
7              for $v = 1$ to $m$ do
8                 foreach sequence $X_{Out}$ of $q$ consecutive nodes
                  in tour $v$ do
9                    $X_{In} := Find\_Insertion\_Sequence$;
10                   if $z(X_{In}) - z(X_{Out}) > Best$ then
11                      $BX_{In} := X_{In}$;
12                      $BX_{Out} := X_{Out}$;
13                      $Best := z(X_{In}) - z(X_{Out})$;
14                      $Found := True$;
15                   end if
16                end foreach
17             end for
18             $q := q + 1$;
19          end while
20          if $Found = True$ then
21             $x' := x - BX_{Out} + BX_{In}$ and
               $z(x') := z(x) - z(X_{Out}) + z(X_{In})$;
22             $x := x'$ and $z(x) := z(x')$;
23          end if
24       until not $Found$;
25   end
```

3.3. The Granular VNS

The second neighborhood of the local search procedure is quite large and the determination of the entering sequence may result to be a cumbersome task. The granular variant aims at reducing the size of the analyzed neighborhoods excluding non promising arcs during the node sequences construction in the local search procedure.

As a preliminary step, the algorithm formulates and optimally solves an assignment problem on graph $G(N \cup \{0\}, A)$ using an ad hoc objective function (see model TOPTW1). Then, dual information (reduced costs) is used to identify more promising arcs. Specifically, the cost for each arc $(i,j) \in A$ in the assignment problem is set equal to $(t_{ij} + w^U_{ij})/(p_i + p_j)$, where w^U_{ij} is the maximum possible waiting time at node j when the service at node i is assumed to start at time e_i.

Since arc costs take into account traveling and waiting times as well as profits, each reduced cost value can be seen as a measure of the likelihood the corresponding arc will belong to high quality solutions: the larger the reduced cost of the variable associated to an arc, the lower the likelihood that the insertion of such an arc in the tour will provide a high profit requiring a low traveling time. Notice that arcs selected by the assignment optimal solution have null reduced costs, and thus they are among the most promising arcs considered by the GVNS for the construction of a solution.

Given a solution x' and a parameter l_{max}, we introduce a set of neighborhoods $H_l(x')$, $l = 1,\ldots,l_{max}$, where l is the granularity parameter defining the subset of arcs which can be taken into account when constructing the entering sequence of nodes in the local search procedure. After optimally solving the assignment problem, the arcs are sorted in non decreasing order of their reduced costs value (set $\overline{A}$). Intervals of granularity are then created by partitioning $\overline{A}$ into l_{max} buckets. The first bucket $\overline{A}_1$ contains all arcs with null reduced cost. The remaining buckets, denoted $\overline{A}_2,\overline{A}_3,\ldots,\overline{A}_{l_{max}}$, are obtained by dividing the set $\overline{A}\setminus\overline{A}_1$ into $l_{max}-1$ subsets with the same cardinality (but possibly the last one). Starting with $l = 1$, GVNS allows the *Local Search* to consider only arcs whose reduced costs are lower than or equal to the worst reduced cost in the subset $\overline{A}_l$ (i.e. for $l > 1$, only arcs contained in the union of buckets $\overline{A}_1,\overline{A}_2,\ldots,\overline{A}_l$ are considered). If an improvement solution is found, l is set to 1, otherwise l is incremented and a larger subset of arcs is made available for the search. This allows the GVNS to diversify the search if a local optimum within the current neighborhood is reached, and on the other hand to intensify the search considering only the most promising arcs when a potentially good region is explored. The GVNS pseudo-code is outlined in Algorithm 3.

Algorithm 3. Pseudo-code of the Granular VNS algorithm.

```
     Input: an initial solution $x_I$ with value $z(x_I)$
     Output: a feasible solution $x^*$ with value $z(x^*)$
     Parameters: $ItMAX$, $ItIMP$, $k_{max}$, $l_{max}$, $q_{max}$
1    begin
2        $x^* := x_I$, $z(x^*) := z(x_I)$;
3        $it = imp := 0$;
4        $l := 1$;
5        $k := 1$;
6        while ($it < ItMAX$) and ($imp < ItIMP$) do
7          $x' := Random(x^*,k)$;
8          $x'' := LocalSearch(x',q_{max},l)$;
9          if $z(x'') > z(x^*)$ then
10           $x^* := x''$ and $z(x^*) := z(x'')$;
11           $imp := 0$ and $k := 1$ and $l := 1$;
12         else
13           $k := \min(k+1, k_{max}+1)$;
14           if $k > k_{max}$ then
15             $l := \min(l+1, l_{max})$;
16             $k := 1$;
17           end if
18           $imp := imp + 1$
19         end if
20         $it := it + 1$;
21       end while
22     end
```

4. Experimental analysis

All the described heuristics have been coded in *Embarcadero Delphi* 2010 and run on a PC Pentium (R) IV with 3 GHz processor. We first define the testing environment. Then the parameters setting and the computational results of the proposed VNS approach and of its granular variant GVNS are analyzed.

4.1. Testing environment

The test problems for OPTW and TOPTW proposed in the literature are based on Solomon and Derosiers's instances (see [25]) and on Cordeau et al.'s ones (see [6]) originally designed for the VRPTW and the Multi Depot Periodic VRPTW (MDPVRPTW), respectively. From now on, for sake of simplicity we will refer to such data sets as to Solomon's and Cordeau's instances, respectively. In all instances the demand of a customer becomes the node profit, whereas all customers are considered as active the same day in the case of Cordeau's instances. The number of nodes in Solomon's data set equals 100, whereas the number of nodes in Cordeau's data set varies from 48 to 288. In [24], the authors used 29 Solomon's instances of series c100, r100 and rc100 and the first 10 Cordeau's instances. The remaining 27 Solomon's instances of series c200, r200 and rc200 and the instances pr11–pr20 of Cordeau were additionally considered in [22,33]. In both papers the authors considered a number of tours $m = 1,\ldots,4$. All these instances are further referred to as *basic instances*. Another data set was proposed in [19] by considering only the first 50 nodes of the first 29 Solomon's instances. Results on this data set are not included, since problems are rather easy to solve. Finally, Vansteenwegen et al. [33] proposed new data sets based on Solomon's instances and the first 10 Cordeau's instances. The number of tours is set equal to the number of vehicles needed to visit all customers. Thus, these new data sets provide instances relatively difficult to solve, for which the optimal solutions are known. We will refer to such instances as to *new instances*. We remark that in all solved instances we computed traveling times rounding distances to the second digit as in [22,24].

A short remark shall be dedicated to the computation of reduced costs. In order to correctly compute the optimal solution of an assignment problem, we have introduced $\bar{v}-1$ copies of the depot, where $\bar{v}$ is a lower bound on the number of tours necessary for visiting all the nodes of a problem. $\bar{v}$ has been computed as the size of the maximal clique in the graph $\widehat{G} = (N,\widehat{A})$, where N is the set of nodes and $\widehat{A}$ is the set of arcs defined as $\widehat{A} = \{(i,j) : e_i + t_{ij} > l_j,\ i,j \in N\}$. The algorithm used to compute the maximal clique is that proposed by Carraghan and Pardalos in [3]. Once determined $\bar{v}$ the original matrix $O((|N|+1)^2)$ of the assignment problem is orlated with $\bar{v}-1$ new rows and columns corresponding to copies of the depot. For each pair of nodes i and j, $i \neq j$, representing copies of the depot we have set $t_{ij} = \infty$.

4.2. Initial solution heuristic

The results obtained with the insertion heuristic on Solomon's and Cordeau's instances for $m = 1,\ldots,4$ are summarized in Table 1. The table reports the average percentage gap between the value of initial solutions and the value of best known solutions (*BK*) over

Table 1
Average percentage gap between initial solution values and *BK*.

Instance set	m			
	1	2	3	4
c100	5.9	8.1	6.6	8.2
r100	9.0	11.1	10.1	10.6
rc100	15.4	16.9	13.6	18.4
c200	5.9	3.9	5.4	0.7
r200	13.2	10.1	2.2	0.2
rc200	21.2	13.8	4.9	0.6
pr01–pr10	19.8	21.9	18.5	16.7
pr11–pr20	16.6	18.7	17.1	16.1

each data set and m. The method performs better on clustered data sets (c100 and c200), whereas the average gap for the random and random-clustered instances is rather high. Very good solutions were obtained for the test problems c200–rc200 and $m = 4$, since the initial heuristic was able to find a solution covering all nodes in some cases. Cordeau's instances seem to be much harder to solve with a simple heuristic. The average gap reported for the initial heuristic is between 16.1% and 21.9%.

4.3. The Granular VNS

4.3.1. Parameters setting

The standard setting of parameters was determined performing several experiments on a small subset of instances. The subset was composed of 6 Solomon's data files and 4 Cordeau's data files. We considered $m = 1,\ldots,4$ and five runs were executed for each instance. Based on these preliminary experiments, the following values were finally selected for the standard setting of parameters: the maximum size k_{max} of the neighborhood explored was set to 7, whereas the parameter q_{max} of the local search procedure was set to 3. For the creation of granular neighborhoods we have partitioned the sorted set into $l_{max} = 10$ buckets. As stopping rule we set the maximum number of iterations *ItMAX* to 500 and the maximum number of iterations allowed without improvement of best solution value *ItIMP* to 100. The algorithm also stopped whenever a solution visiting all customers was found.

With this standard parameters setting, the average gap to the best known solution value (BK) achieved by GVNS on the subset of instances varies from 0.9% to 2.0%. The average computational time increases with m from 22.3 seconds ($m = 1$) up to 124.1 seconds ($m = 4$). These results represent the best tradeoff between the computational time and the quality of obtained solutions. For example, increasing *ItIMP* to 200 leads to a greater average computational time (33.7 seconds up to 187.3 seconds) without a significant change in the average gap (0.8% and 1.9%, respectively). Contrarily, a lower value of *ItIMP* causes a decrease in the computational time, but the average gap increases adequately. With *ItIMP* = 50, the average gap varies from 1.4% to 2.7%. Regarding the maximum number of iterations *ItMAX*, larger values of the parameter do not seem to have a significant impact on GVNS performance. The remaining parameters have been analyzed in a similar way. When k_{max} is set to a lower value than 7, the perturbation of the solution provided by the procedure *Random* becomes too weak and the results get worse. Note that the extension of the granular bucket l depends on the current value of k: l increases whenever $k > k_{max}$ and k is then reset to 1. Hence, a lower value for k_{max} causes the granular bucket to be extended after few iterations, which leads to greater computational times. The parameter q_{max} has probably the most critical impact on the computational time. When q_{max} is set to a greater value than 3, the dynamic programming procedure of the local search consumes much more computational time. The effect is remarkable already for $q_{max} = 4$, since the observed average computational time varies from 39.5 seconds to 234.2 seconds. Finally, the value of l_{max} was selected so that the algorithm can possibly explore the whole neighborhood before the stopping rule is applied. This might not be guaranteed with larger values of l_{max}. On the other hand, the experiments have shown that if l_{max} is set to a smaller value than 10, the average computational time increases but the obtained solutions are even worse in some cases. For example, with $l_{max} = 5$, the average gap varies from 1.0% to 2.1% and the computational time increases from 29.4 seconds to 156.2 seconds.

4.3.2. Computational results on basic instances

The performances of the proposed VNS and of its variant GVNS are compared to the state-of-the-art methods, notably GRASP-ELS [17], ILS [33], ACS [22] and VNS* introduced by Tricoire et al. in [29] (this algorithm is marked with a "$*$" in order to differentiate it from our VNS). Note that some results of VNS*, not reported in [29], have been made available on-line (see [30]). The results are summarized in Table 2. The table reports the average percentage gap to best known solution values *BK* and the average computational time in seconds for each instance set and $m = 1,\ldots,4$. The results of GVNS, GRASP-ELS and ACS were obtained with 5 runs of the algorithm on each instance and the average over the runs was considered in the comparison. The same applies to VNS* except that the algorithm was run 10 times per instance. ILS is a deterministic algorithm. The comparison of VNS and GVNS proves the contribution of granular neighborhoods. The average computational times reported for VNS and GVNS are 81.4 seconds and 54.9 seconds, respectively. With much less computational effort, GVNS achieves only slightly worse average gap 1.3% compared to 1.2% of VNS. Among the state-of-the-art algorithms only the GRASP-ELS gets a little better average gap (1.1%) than GVNS. GRASP-ELS also spends less computational time (22.7 seconds). However, we cannot conclude that GRASP-ELS is strictly better than GVNS, since the latter provides better results on most of the Cordeau's instances. Note that the comparison of GVNS and GRASP-ELS is unbiased since both algorithms were coded in Delphi and the experiments were carried out on the same computer. Regarding the other algorithms, ILS was run on a machine approximately two times faster than ours as we have deduced from the comparison of the performance of various computers in [7]. ACS and VNS* were run on a machine comparable to our PC. ILS is definitely the fastest algorithm, but its average gap 3.2% is definitely greater than that of the other algorithms. ACS requires a large amount of computational time (1373.5 seconds on average) and the achieved gap is relatively high (2.1%). The average gap reported for VNS* is 1.2% as for our VNS. The average computational time spent by VNS* is 372.8 seconds. This is significantly greater than the time required by our GVNS or even by VNS.

GVNS and VNS found 25 new best known solution values for Cordeau's instances. Table 3 reports the new *BKs* obtained with the standard setting of parameters. The first column shows the instance name, the number m of tours is indicated in the second column and the last column reports the new solution value.

ILS is a method that aims at providing good solutions very quickly, whereas our method looks for near-optimal solutions at the cost of larger computational times. In order to compare the performance of GVNS and ILS correctly, we run GVNS with a stopping rule based on the solution value obtained by ILS. Specifically, GVNS was stopped as soon as it found an equal or better solution value. It should be noted that in few cases, GVNS terminates without finding a solution with value at least equal to ILS solution value. This mainly happens in Solomon's instances of series c100, r100, rc100. However, as shown in Table 4, GVNS provides again, on average, better results than ILS. Minimum and maximum average gap reported for GVNS are 2.5% and 3.3%, respectively, whereas values for ILS are 2.7% and 3.5%. The average computational time spent by GVNS varies from 4.5 seconds to 15.8 seconds. The same values for ILS are 1.2 seconds and 4.7 seconds. The greater computational times required by GVNS are always due to instances c100, r100, rc100.

Our VNS consists of several components that may contribute differently to the overall performance of GVNS. One of the most critical aspects is the local search procedure and the role played by its two neighborhoods. In order to better analyze the contribution of the first neighborhood in the local search procedure, we ran some additional experiments. We recall that the first neighborhood of the local search is defined by the common routing moves 2-opt, Or-opt, Swap and 2-opt*. The analysis was performed on Cordeau's instances for $m = 1,\ldots,4$ using the standard setting of parameters. We selected

Table 2
Comparison of VNS and its granular variant GVNS to the state-of-the-art methods.

Instance set	VNS		GVNS		GRASP-ELS		ILS		ACS		VNS*	
	Gap (%)	Time (seconds)	Gap	Time	Gap	Time	Gap	Time	Gap	Time	Gap	Time
m = 1												
c100	0.3	67.4	0.3	46.0	0.0	22.6	1.1	0.3	0.0	6.3	0.1	98.4
r100	0.9	23.4	1.1	16.5	0.2	3.5	1.9	0.2	0.2	383.4	0.0	89.1
rc100	0.7	14.9	0.5	10.8	0.4	2.0	2.5	0.2	0.0	143.2	0.0	65.2
c200	1.1	101.3	1.0	65.0	0.6	32.2	2.3	1.7	0.6	342.6	0.2	560.2
r200	2.7	33.7	2.7	19.7	1.6	11.2	2.9	1.7	3.2	1 556.7	1.1	1 065.8
rc200	3.5	17.0	3.7	12.1	2.2	8.2	3.4	1.6	2.0	1 544.5	1.3	869.4
pr01–10	1.6	16.7	1.6	12.4	1.4	5.0	4.7	1.8	1.2	1 626.6	1.1	822.1
pr11–20	3.4	30.6	3.1	24.2	2.2	7.9	8.4	2.0	10.7	887.6	2.2	1 045.9
m = 2												
c100	0.8	192.8	0.7	139.5	0.1	70.9	0.9	1.1	0.3	818.0	0.3	88.0
r100	1.7	63.1	1.6	60.3	1.5	8.0	2.2	0.9	0.6	1 559.4	1.2	63.5
rc100	1.6	28.2	2.8	20.3	2.3	4.7	2.5	0.7	1.2	1 375.8	1.5	55.2
c200	0.7	56.5	0.6	33.8	0.3	29.3	2.5	3.5	1.8	1 398.1	0.9	545.6
r200	1.1	13.4	1.3	14.7	0.6	17.6	2.7	2.3	3.7	2 735.1	0.7	1 015.1
rc200	2.1	13.2	2.5	12.8	1.1	17.1	4.1	2.2	3.7	2 342.7	1.4	804.8
pr01–10	1.5	56.7	1.5	39.1	2.0	19.5	6.0	4.8	3.3	1 889.7	3.6	524.8
pr11–20	1.5	123.1	1.4	82.4	2.8	28.8	7.1	5.2	5.4	2 384.8	2.9	618.8
m = 3												
c100	1.0	221.1	0.8	165.0	0.4	86.7	2.4	1.5	0.7	1 043.2	0.6	85.5
r100	2.0	90.1	2.1	73.9	1.5	13.9	1.7	1.7	1.4	1 668.9	1.4	61.9
rc100	2.2	43.0	2.3	33.7	2.8	8.7	1.7	1.1	2.1	1 476.8	1.4	60.6
c200	0.3	3.0	0.2	7.7	0.0	0.1	1.9	2.2	1.6	1 320.4	0.0	241.6
r200	0.2	2.1	0.2	7.0	0.0	2.5	0.3	1.4	0.2	1 171.6	0.1	321.7
rc200	0.4	4.7	0.4	7.4	0.2	8.3	1.4	1.7	0.8	1 509.6	0.2	404.0
pr01–10	1.0	117.9	0.9	85.9	1.7	40.6	6.4	9.2	3.8	2 163.8	3.4	473.2
pr11–20	1.1	294.0	1.1	150.7	2.5	42.9	8.5	9.7	5.9	2 228.2	2.6	517.5
m = 4												
c100	1.2	243.2	1.4	133.2	0.8	84.6	3.0	2.4	1.4	1 056.1	1.0	81.9
r100	1.8	139.5	2.1	84.7	1.5	24.2	3.2	2.6	1.7	1 652.5	1.4	61.2
rc100	1.6	59.1	1.6	36.9	2.1	13.4	2.1	2.0	1.9	1 854.0	2.2	58.5
c200	0.0	0.3	0.0	0.6	0.0	0.0	0.0	1.0	0.1	7.7	0.0	104.8
r200	0.0	0.3	0.0	0.3	0.0	0.0	0.0	0.9	0.0	126.5	0.0	150.7
rc200	0.0	0.8	0.0	0.9	0.0	0.0	0.0	1.1	0.0	646.7	0.0	164.6
pr01–10	1.3	161.0	1.1	127.3	2.0	45.8	6.5	14.1	3.0	2 447.7	3.0	403.2
pr11–20	0.7	372.4	1.2	232.6	1.8	65.3	6.9	13.7	4.8	2 583.5	1.9	408.0
Average	1.2	81.4	1.3	54.9	1.1	22.7	3.2	3.0	2.1	1 373.5	1.2	372.8

Table 3
New best known solution values found by VNS or GVNS.

Instance	*m*	New *BK*	Instance	*m*	New *BK*
pr05	2	1090	pr17	3	835
pr08	2	834	pr18	3	1281
pr09	2	905	pr19	3	1417
pr15	2	1219	pr20	3	1684
pr16	2	1231	pr06	4	1860
pr19	2	1034	pr07	4	876
pr20	2	1232	pr09	4	1619
pr04	3	1294	pr10	4	1939
pr10	3	1573	pr14	4	1670
pr13	3	1132	pr16	4	2065
pr14	3	1372	pr18	4	1525
pr15	3	1650	pr19	4	1723
pr16	3	1668			

these instances for the analysis because such test problems are reputed as hard to solve. When all these routing moves were applied, the average gaps to BK achieved on Cordeau's instances pr01–pr10 and pr11–pr20 were 1.3% and 1.7%, respectively. The average computational times were 66.2 seconds and 122.5 seconds. When all the routing moves were disabled, the results were as follows: 2.9% and 57.6 seconds, 3.8% and 110.7 seconds, respectively. Obviously, the average computational time decreased but the average gap increased significantly. The results show that the first neighborhood plays an important role in the effective exploration of the solution space. Furthermore, we ran the algorithm without performing the 2-opt move on a single tour. The average results were then 1.8% and 63.1 seconds, 2.7% and 117.5 seconds, respectively. Hence, we can conclude that exploring the 2-opt neighborhood is still beneficial since it contributes to the quality of obtained solution requiring only a little bit more computational effort.

Finally, detailed results obtained by GVNS on basic instances are reported in Tables 5–12. Each table consists of two parts of identical structure. The first column contains the instance name, the second column reports the best known solution value *BK*. The following three columns show minimum, average and maximum solution value obtained within five runs of the algorithm. Column "Gap" provides the percentage gap with respect to *BK*. The last three columns show minimum, average and maximum computational time. The optimal value, when know, is reported in italic in the BK columns. The same in Tables 13–15.

4.3.3. Computational results on new instances

Tables 13–15 report the results obtained on the new data sets, for which there always exists a solution visiting all customers. Each table reports for each instance the number of tours and the optimal value (which is the sum of the profits of all customers). For each algorithm included in the comparison, the average solution value, the average gap to the optimum and the average computational time are reported in that order. The number of runs executed is the same as for the basic instances.

On the first set of Solomon's instances, the average gap and the average computational time for GVNS are 1.1% and 29.7 seconds,

Table 4
Average performance of GVNS when ILS solution value is used as stopping criterion.

Instance	m	GVNS		ILS		m	GVNS		ILS	
		Gap (%)	Time (seconds)	Gap (%)	Time (seconds)		Gap (%)	Time (seconds)	Gap (%)	Time (seconds)
c100	1	1.1	0.7	1.1	0.3	3	2.4	37.0	2.4	1.5
r100		1.5	5.4	1.9	0.2		2.3	54.5	1.7	1.7
rc100		2.6	2.0	2.5	0.2		3.0	13.6	1.7	1.1
c200		1.8	3.6	2.3	1.7		1.2	0.6	1.9	2.2
r200		3.4	9.3	2.9	1.7		0.3	1.2	0.3	1.4
rc200		4.3	8.8	3.4	1.6		1.2	1.3	1.4	1.7
pr01–10		4.2	3.7	4.7	1.8		5.7	8.8	6.4	9.2
pr11–20		6.8	2.8	8.4	2.0		7.5	9.3	8.5	9.7
Average		3.2	4.5	3.4	1.2		3.0	15.8	3.0	3.6
c100	2	1.0	40.3	0.9	1.1	4	2.6	21.8	3.0	2.4
r100		2.5	30.7	2.2	0.9		3.1	50.0	3.2	2.6
rc100		3.4	15.4	2.5	0.7		2.8	11.1	2.1	2.0
c200		2.1	0.6	2.5	3.5		0.0	0.6	0.0	1.0
r200		2.5	4.6	2.7	2.3		0.0	0.3	0.0	0.9
rc200		3.8	5.6	4.1	2.2		0.0	0.9	0.0	1.1
pr01–10		5.4	8.8	6.0	4.8		5.8	6.0	6.5	14.1
pr11–20		6.0	10.7	7.1	5.2		6.0	7.7	6.9	13.7
Average		3.3	14.6	3.5	2.6		2.5	12.3	2.7	4.7

Table 5
Detailed results of GVNS on Solomon's instances with $m = 1$.

Instance	BK	GVNS			Gap (%)	Time (seconds)			Instance	BK	GVNS			Gap (%)	Time (seconds)		
		Min	Avg	Max		Min	Avg	Max			Min	Avg	Max		Min	Avg	Max
c101	*320*	320	320.0	320	0.0	0.2	0.2	0.3	c201	870	850	850.0	850	2.3	0.1	0.1	0.1
c102	*360*	360	360.0	360	0.0	38.1	67.4	107.2	c202	930	890	916.0	930	1.5	64.3	78.7	91.1
c103	*400*	390	396.0	400	1.0	208.9	382.4	713.5	c203	960	950	956.0	960	0.4	222.9	329.0	382.4
c104	*420*	410	410.0	410	2.4	367.8	1005.9	2051.4	c204	980	960	966.0	970	1.4	706.1	974.0	1322.3
c105	*340*	340	340.0	340	0.0	0.2	3.3	9.0	c205	910	890	898.0	900	1.3	7.1	12.5	17.7
c106	*340*	340	340.0	340	0.0	0.9	6.5	26.4	c206	930	920	922.0	930	0.9	30.0	34.4	43.8
c107	*370*	350	358.0	360	3.2	0.2	0.7	1.2	c207	930	920	928.0	930	0.2	11.8	36.3	50.9
c108	*370*	350	354.0	370	4.3	0.4	1.1	2.9	c208	950	940	942.0	950	0.8	65.3	74.2	89.4
c109	*380*	380	380.0	380	0.0	19.9	30.6	45.0									
r101	*198*	197	197.0	197	0.5	0.2	0.2	0.3	r201	797	765	775.6	785	2.7	2.3	6.7	11.7
r102	*286*	269	274.8	281	3.9	3.9	13.4	25.3	r202	929	875	881.4	890	5.1	6.5	13.4	21.4
r103	*293*	284	286.0	288	2.4	16.3	33.8	52.4	r203	1021	978	992.2	1002	2.8	18.1	34.7	54.1
r104	*303*	295	298.6	301	1.5	41.0	72.7	150.2	r204	1086	1065	1073.8	1077	1.1	58.1	77.6	95.8
r105	*247*	223	230.6	236	6.6	0.2	0.3	0.3	r205	953	881	905.8	919	5.0	9.3	15.3	21.7
r106	*293*	276	280.4	283	4.3	5.0	19.1	39.8	r206	1029	950	966.6	982	6.1	18.3	34.9	74.8
r107	*299*	284	287.2	291	3.9	23.1	50.3	103.8	r207	1072	1017	1022.0	1027	4.7	31.2	46.6	81.6
r108	*308*	297	301.4	308	2.1	12.1	50.1	115.2	r208	1112	1082	1083.6	1086	2.6	41.0	52.9	68.2
r109	*277*	275	276.4	277	0.2	1.8	10.4	35.4	r209	950	920	926.0	933	2.5	28.7	37.3	48.6
r110	*284*	278	279.2	281	1.7	4.0	16.8	32.5	r210	987	950	961.4	975	2.6	15.1	30.1	47.7
r111	*297*	289	290.6	292	2.2	38.9	54.2	87.4	r211	1046	1015	1025.6	1038	2.0	17.1	22.5	26.9
r112	*298*	289	289.6	290	2.8	18.0	31.9	42.5									
rc101	*219*	219	219.0	219	0.0	1.6	2.1	2.6	rc201	795	778	784.0	788	1.4	2.0	3.7	5.2
rc102	*266*	236	246.0	253	7.5	4.2	5.8	8.6	rc202	936	885	890.6	900	4.9	7.5	12.1	17.8
rc103	*266*	249	253.2	256	4.8	9.4	23.3	42.1	rc203	1003	936	954.4	969	4.8	13.2	22.5	32.9
rc104	*301*	301	301.0	301	0.0	10.8	16.2	22.4	rc204	1136	1075	1101.8	1137	3.0	14.0	31.9	55.8
rc105	*244*	217	227.0	231	7.0	1.6	2.7	4.0	rc205	859	835	843.8	853	1.8	3.5	7.0	9.9
rc106	*252*	252	252.0	252	0.0	1.5	2.2	2.6	rc206	895	859	866.6	879	3.2	8.9	11.6	19.7
rc107	*277*	253	261.2	274	5.7	8.1	15.7	31.2	rc207	983	901	911.4	927	7.3	9.5	14.7	19.1
rc108	*298*	277	288.8	298	3.1	4.9	10.4	15.4	rc208	1053	992	1000.2	1014	5.0	8.7	24.6	60.5

respectively. GRASP-ELS provides better results (average gap equal to 0.5%), but the average computational time is greater (65.7 seconds). VNS* is faster (8.5 seconds) than GVNS and GRASP-ELS and also the average gap 0.4% is better. ILS requires only 3.2 seconds on average, but the reported average gap is equal to 1.8% thus greater than that provided by the other algorithms. The second set of Solomon's instances seems to be quite easy to solve. The average gap for GVNS, GRASP-ELS and ILS is respectively 0.1%, 0.0% and 0.4% and the average computational time is 3.2 seconds, 3.5 seconds and 1.5 seconds, respectively. The results of VNS* are not available for this data set. The most difficult to solve among the new instances seem to be those of Cordeau. All algorithms require on average a relatively large amount of computational time. The best average gap 0.9% is achieved by GRASP-ELS with an average of 71.5 seconds. GVNS obtains an average gap equal to 1.2% and requires 51.3 seconds on average. VNS* spends 71.7 seconds on average and achieves an average gap of 1.3%. It is interesting to note that even for ILS the average computational time increased significantly (30.4 seconds), whereas the average gap equal to 2.3% is still greater compared to the gap of other algorithms. Thus, on average, we can conclude that GVNS provides the best tradeoff between quality of solutions and computational time for this

Table 6
Detailed results of GVNS on Solomon's instances with $m = 2$.

Instance	BK	GVNS			Gap (%)	Time (seconds)			Instance	BK	GVNS			Gap (%)	Time (seconds)		
		Min	Avg	Max		Min	Avg	Max			Min	Avg	Max		Min	Avg	Max
c101	590	590	590.0	590	0.0	8.4	9.3	11.4	c201	1460	1450	1450.0	1450	0.7	4.6	5.8	7.8
c102	660	640	648.0	650	1.8	41.3	47.5	63.6	c202	1470	1460	1464.0	1470	0.4	19.3	21.1	24.0
c103	720	690	704.0	710	2.2	160.9	271.6	368.0	c203	1480	1460	1464.0	1470	1.1	36.0	46.1	52.5
c104	760	740	746.0	750	1.8	486.3	616.4	1012.7	c204	1480	1470	1472.0	1480	0.5	107.9	135.5	182.1
c105	640	640	640.0	640	0.0	30.0	35.0	38.6	c205	1470	1460	1466.0	1470	0.3	7.8	10.8	13.9
c106	620	620	620.0	620	0.0	35.1	44.4	51.7	c206	1480	1470	1472.0	1480	0.5	13.1	14.1	16.0
c107	670	670	670.0	670	0.0	31.2	34.8	37.4	c207	1490	1480	1484.0	1490	0.4	15.4	18.1	22.3
c108	680	680	680.0	680	0.0	60.1	65.7	78.8	c208	1490	1480	1480.0	1480	0.7	16.9	18.8	20.5
c109	720	710	716.0	720	0.6	110.1	131.1	157.8									
r101	349	349	349.0	349	0.0	3.7	4.0	4.3	r201	1250	1209	1225.8	1235	1.9	12.2	15.0	16.6
r102	508	482	492.8	499	3.0	13.4	17.4	24.3	r202	1347	1284	1306.2	1327	3.0	12.9	15.4	20.3
r103	522	504	506.6	508	3.0	29.3	37.4	50.1	r203	1414	1382	1392.6	1407	1.5	13.9	17.7	21.6
r104	549	537	539.2	541	1.8	82.9	93.7	99.4	r204	*1458*	1449	1452.6	1458	0.4	5.7	8.9	12.3
r105	453	442	442.6	445	2.3	4.8	5.3	5.6	r205	1379	1341	1351.6	1362	2.0	13.3	20.1	23.3
r106	529	517	524.4	529	0.9	21.2	31.2	47.6	r206	1440	1423	1429.8	1435	0.7	14.4	17.1	19.6
r107	535	521	522.2	527	2.4	44.5	56.5	72.5	r207	*1458*	1444	1449.8	1454	0.6	8.2	12.4	17.2
r108	557	541	545.6	551	2.0	101.5	179.5	289.6	r208	*1458*	1458	1458.0	1458	0.0	7.1	8.3	9.8
r109	506	497	501.6	506	0.9	18.1	19.7	21.7	r209	1405	1375	1389.6	1402	1.1	10.8	16.3	23.3
r110	525	523	523.0	523	0.4	43.5	65.5	89.9	r210	1423	1373	1391.0	1404	2.2	11.9	16.0	19.5
r111	538	525	530.6	533	1.4	54.6	115.1	223.5	r211	*1458*	1448	1452.6	1455	0.4	9.7	14.8	21.2
r112	543	525	536.6	542	1.2	53.1	98.8	190.0									
rc101	427	419	420.6	427	1.5	2.4	2.8	3.5	rc201	1377	1348	1359.6	1369	1.3	5.9	7.3	9.8
rc102	505	473	485.4	504	3.9	6.8	8.4	9.8	rc202	1509	1446	1465.0	1476	2.9	12.8	16.8	20.3
rc103	524	493	502.0	514	4.2	20.8	26.8	39.9	rc203	1632	1550	1561.6	1570	4.3	7.7	11.8	19.7
rc104	575	535	554.8	562	3.5	28.0	43.6	74.9	rc204	1716	1681	1699.2	1713	1.0	6.8	8.8	11.4
rc105	480	447	456.8	478	4.8	5.3	7.1	12.0	rc205	1455	1389	1396.0	1407	4.1	5.3	8.5	13.8
rc106	483	477	480.4	483	0.5	6.1	11.0	18.1	rc206	1546	1470	1507.4	1531	2.5	9.7	18.1	24.1
rc107	534	520	524.0	529	1.9	17.9	26.0	36.5	rc207	1587	1526	1546.6	1563	2.5	10.3	18.3	24.7
rc108	556	538	544.6	553	2.1	24.1	36.8	48.4	rc208	1691	1656	1669.6	1687	1.3	6.4	12.5	19.5

Table 7
Detailed results of GVNS on Solomon's instances with $m = 3$.

Instance	BK	GVNS			Gap (%)	Time (seconds)			Instance	BK	GVNS			Gap (%)	Time (seconds)		
		Min	Avg	Max		Min	Avg	Max			Min	Avg	Max		Min	Avg	Max
c101	810	800	808.0	810	0.2	14.3	18.0	24.9	c201	*1810*	1800	1800.0	1800	0.6	3.6	3.7	3.8
c102	920	910	916.0	920	0.4	58.5	94.1	137.7	c202	*1810*	1790	1806.0	1810	0.2	6.3	8.8	12.4
c103	980	960	964.0	970	1.6	237.5	332.3	404.1	c203	*1810*	1780	1804.0	1810	0.3	8.6	10.9	12.1
c104	1030	1010	1012.0	1020	1.7	414.9	604.0	736.3	c204	*1810*	1790	1802.0	1810	0.4	7.1	10.9	16.0
c105	870	860	864.0	870	0.7	27.4	36.5	54.9	c205	*1810*	1810	1810.0	1810	0.0	4.5	4.9	5.5
c106	870	860	866.0	870	0.5	39.3	50.2	77.2	c206	*1810*	1810	1810.0	1810	0.0	6.3	6.5	6.7
c107	910	890	906.0	910	0.4	51.7	63.4	93.0	c207	*1810*	1810	1810.0	1810	0.0	6.8	7.6	8.3
c108	920	910	914.0	920	0.7	70.6	105.9	143.9	c208	*1810*	1810	1810.0	1810	0.0	7.4	8.3	9.1
c109	970	950	958.0	960	1.2	125.8	180.7	230.9									
r101	484	476	477.4	481	1.4	5.3	7.7	11.3	r201	1441	1413	1416.4	1423	1.7	4.7	6.4	10.0
r102	691	664	674.2	686	2.4	13.7	21.9	34.7	r202	*1458*	1443	1450.2	1458	0.5	11.4	21.0	53.0
r103	747	717	724.4	730	3.0	28.6	68.4	117.7	r203	*1458*	1458	1458.0	1458	0.0	7.7	9.0	10.6
r104	777	749	762.0	770	1.9	66.9	103.7	141.6	r204	*1458*	1458	1458.0	1458	0.0	8.4	11.9	19.9
r105	620	608	612.4	615	1.2	9.3	11.0	12.1	r205	*1458*	1458	1458.0	1458	0.0	7.1	9.0	14.8
r106	726	705	713.4	723	1.7	34.6	47.0	57.2	r206	*1458*	1458	1458.0	1458	0.0	6.8	9.0	11.2
r107	760	749	755.0	759	0.7	97.9	110.1	132.2	r207	*1458*	1458	1458.0	1458	0.0	0.2	0.2	0.2
r108	797	763	770.4	781	3.3	123.9	187.9	232.3	r208	*1458*	1458	1458.0	1458	0.0	0.1	0.1	0.1
r109	710	699	699.6	700	1.5	15.8	24.1	39.4	r209	*1458*	1458	1458.0	1458	0.0	0.2	0.2	0.2
r110	737	705	716.2	734	2.8	47.6	69.8	91.0	r210	*1458*	1458	1458.0	1458	0.0	8.7	9.7	11.0
r111	773	739	749.4	761	3.1	40.0	62.6	111.7	r211	*1458*	1458	1458.0	1458	0.0	0.2	0.2	0.2
r112	772	740	751.0	757	2.7	133.2	173.0	248.9									
rc101	621	599	602.4	614	3.0	3.6	5.7	10.9	rc201	1698	1672	1676.0	1682	1.3	7.9	11.6	16.8
rc102	714	677	692.0	705	3.1	10.6	14.0	19.0	rc202	*1724*	1684	1702.8	1709	1.2	6.8	10.5	13.7
rc103	764	732	741.2	764	3.0	35.6	47.0	64.1	rc203	*1724*	1724	1724.0	1724	0.0	6.5	7.9	9.1
rc104	833	813	826.0	834	0.8	47.8	85.6	109.0	rc204	*1724*	1724	1724.0	1724	0.0	0.1	0.1	0.1
rc105	682	655	670.2	680	1.7	13.0	20.8	26.8	rc205	1709	1695	1702.2	1706	0.4	6.1	11.6	17.7
rc106	705	689	692.6	695	1.8	9.7	17.4	29.0	rc206	*1724*	1724	1724.0	1724	0.0	7.2	8.7	9.7
rc107	773	749	756.4	762	2.1	22.9	30.7	38.8	rc207	1724	1724	1724.0	1724	0.0	7.2	8.7	11.9
rc108	795	766	773.0	782	2.8	27.4	48.2	87.0	rc208	*1724*	1724	1724.0	1724	0.0	0.2	0.2	0.2

Table 8
Detailed results of GVNS on Solomon's instances with $m = 4$.

Instance	BK	GVNS			Gap (%)	Time (seconds)			Instance	BK	GVNS			Gap (%)	Time (seconds)		
		Min	Avg	Max		Min	Avg	Max			Min	Avg	Max		Min	Avg	Max
c101	1020	1010	1014.0	1020	0.6	17.9	22.1	29.0	c201	*1810*	1810	1810.0	1810	0.0	0.1	0.1	0.1
c102	1150	1130	1132.0	1140	1.6	49.4	58.2	71.8	c202	*1810*	1810	1810.0	1810	0.0	0.6	1.9	4.1
c103	1200	1170	1178.0	1180	1.8	151.3	221.2	331.2	c203	*1810*	1810	1810.0	1810	0.0	0.8	1.5	2.3
c104	1260	1220	1226.0	1230	2.7	413.4	513.0	623.7	c204	*1810*	1810	1810.0	1810	0.0	0.5	0.5	0.6
c105	1060	1060	1060.0	1060	0.0	30.8	33.3	34.4	c205	*1810*	1810	1810.0	1810	0.0	0.1	0.1	0.1
c106	1080	1050	1052.0	1060	2.6	45.4	51.5	59.1	c206	*1810*	1810	1810.0	1810	0.0	0.1	0.1	0.1
c107	1120	1110	1112.0	1120	0.7	52.5	62.0	84.4	c207	*1810*	1810	1810.0	1810	0.0	0.1	0.1	0.1
c108	1130	1110	1116.0	1130	1.2	74.3	94.5	121.4	c208	*1810*	1810	1810.0	1810	0.0	0.1	0.1	0.1
c109	1190	1160	1170.0	1180	1.7	125.0	143.2	199.0									
r101	611	605	607.0	608	0.7	8.7	11.9	15.3	r201	*1458*	1458	1458.0	1458	0.0	1.2	1.4	1.7
r102	840	809	814.6	822	3.0	16.1	23.0	34.0	r202	*1458*	1458	1458.0	1458	0.0	0.1	0.1	0.1
r103	924	894	902.0	914	2.4	29.4	68.9	100.5	r203	*1458*	1458	1458.0	1458	0.0	0.2	0.2	0.2
r104	972	939	939.0	939	3.4	86.3	103.0	128.4	r204	*1458*	1458	1458.0	1458	0.0	0.1	0.1	0.1
r105	778	765	770.6	776	1.0	11.7	15.5	26.9	r205	*1458*	1458	1458.0	1458	0.0	0.1	0.1	0.1
r106	905	870	878.2	884	3.0	42.1	54.7	66.1	r206	*1458*	1458	1458.0	1458	0.0	0.2	0.2	0.2
r107	945	922	932.8	946	1.3	106.4	112.4	125.5	r207	*1458*	1458	1458.0	1458	0.0	0.2	0.2	0.2
r108	994	953	958.6	968	3.6	123.0	198.5	256.2	r208	*1458*	1458	1458.0	1458	0.0	0.1	0.1	0.1
r109	884	859	869.4	879	1.7	45.3	55.0	71.6	r209	*1458*	1458	1458.0	1458	0.0	0.2	0.2	0.2
r110	914	873	893.2	915	2.3	51.4	92.9	112.4	r210	*1458*	1458	1458.0	1458	0.0	0.2	0.2	0.2
r111	949	928	937.4	952	1.2	78.7	96.5	108.4	r211	*1458*	1458	1458.0	1458	0.0	0.2	0.2	0.2
r112	971	935	950.4	968	2.1	144.3	184.6	232.9									
rc101	811	791	797.6	808	1.7	7.1	10.1	14.6	rc201	*1724*	1719	1723.0	1724	0.1	1.1	1.9	2.6
rc102	908	878	887.0	897	2.3	13.9	21.3	34.1	rc202	*1724*	1724	1724.0	1724	0.0	0.5	0.8	1.2
rc103	967	942	948.4	953	1.9	29.6	46.7	60.1	rc203	*1724*	1724	1724.0	1724	0.0	0.1	0.1	0.2
rc104	1059	1019	1036.4	1048	2.1	42.9	76.6	105.9	rc204	*1724*	1724	1724.0	1724	0.0	0.1	0.1	0.1
rc105	875	848	866.0	875	1.0	15.7	20.3	24.2	rc205	*1724*	1724	1724.0	1724	0.0	1.0	3.3	9.4
rc106	909	884	896.2	902	1.4	16.5	20.7	24.2	rc206	*1724*	1724	1724.0	1724	0.0	0.1	0.1	0.1
rc107	980	959	963.0	966	1.7	26.2	46.0	62.5	rc207	*1724*	1724	1724.0	1724	0.0	0.3	0.5	0.6
rc108	1025	1009	1014.8	1023	1.0	41.5	53.6	89.9	rc208	*1724*	1724	1724.0	1724	0.0	0.2	0.2	0.2

Table 9
Detailed results of GVNS on Cordeau's instances with $m = 1$.

Instance	BK	GVNS			Gap (%)	Time (seconds)			Instance	BK	GVNS			Gap (%)	Time (seconds)		
		Min	Avg	Max		Min	Avg	Max			Min	Avg	Max		Min	Avg	Max
pr01	*308*	304	307.2	308	0.3	0.9	1.2	1.6	pr11	330	327	329.0	330	0.3	1.2	1.9	3.1
pr02	*404*	402	403.6	404	0.1	2.9	3.7	5.1	pr12	442	435	435.0	435	1.6	5.1	6.5	8.5
pr03	*394*	388	388.0	388	1.5	3.9	4.1	4.4	pr13	461	452	452.4	453	1.9	9.1	16.2	20.6
pr04	*489*	460	475.4	489	2.8	9.0	14.7	24.8	pr14	567	531	540.6	549	4.7	20.1	32.4	46.6
pr05	*595*	567	578.0	587	2.9	13.4	20.5	31.5	pr15	685	653	656.6	665	4.1	21.9	29.4	45.9
pr06	590	576	584.2	590	1.0	21.1	29.0	41.9	pr16	674	601	643.4	673	4.5	47.8	60.9	73.0
pr07	*298*	293	297.0	298	0.3	1.2	1.7	2.7	pr17	359	352	354.6	359	1.2	3.5	5.4	7.3
pr08	*463*	463	463.0	463	0.0	3.2	3.8	4.6	pr18	535	529	530.8	535	0.8	8.2	10.4	12.6
pr09	*493*	467	482.0	493	2.2	7.6	12.3	16.0	pr19	562	507	507.8	510	9.6	17.6	22.1	27.4
pr10	594	559	564.4	579	5.0	26.2	32.7	47.7	pr20	667	650	655.0	662	1.8	39.6	57.0	67.6

Table 10
Detailed results of GVNS on Cordeau's instances with $m = 2$.

Instance	BK	GVNS			Gap (%)	Time (seconds)			Instance	BK	GVNS			Gap (%)	Time (seconds)		
		Min	Avg	Max		Min	Avg	Max			Min	Avg	Max		Min	Avg	Max
pr01	502	480	495.0	502	1.4	2.7	3.6	4.2	pr11	547	531	539.4	547	1.4	2.3	3.7	5.3
pr02	714	698	706.4	713	1.1	9.6	12.3	15.4	pr12	774	742	750.8	767	3.0	12.5	28.0	57.8
pr03	742	715	718.4	721	3.2	14.0	19.2	27.3	pr13	831	825	827.0	829	0.5	32.0	41.3	48.9
pr04	924	902	903.8	908	2.2	25.7	33.3	46.3	pr14	1017	987	999.0	1009	1.8	88.3	117.3	132.5
pr05	1089	1042	1073.6	1082	1.4	75.9	94.3	108.5	pr15	1216	1198	1210.4	1219	0.5	87.9	126.6	153.3
pr06	1076	1057	1070.2	1076	0.5	47.1	71.0	87.8	pr16	1218	1212	1217.8	1224	0.0	147.2	209.6	265.6
pr07	566	551	557.8	566	1.4	4.8	6.0	7.8	pr17	652	620	626.0	630	4.0	7.6	9.6	13.5
pr08	832	819	824.4	829	0.9	14.8	20.0	27.3	pr18	938	904	914.4	927	2.5	25.7	39.5	52.4
pr09	900	865	883.4	901	1.8	37.9	49.7	70.5	pr19	1008	990	1004.4	1018	0.4	87.7	120.3	168.3
pr10	1124	1081	1109.0	1124	1.3	64.9	81.5	108.9	pr20	1223	1214	1224.4	1232	−0.1	104.3	128.5	153.5

Table 11
Detailed results of GVNS on Cordeau's instances with $m = 3$.

Instance	BK	GVNS			Gap (%)	Time (seconds)			Instance	BK	GVNS			Gap (%)	Time (seconds)		
		Min	Avg	Max		Min	Avg	Max			Min	Avg	Max		Min	Avg	Max
pr01	622	604	608.4	614	2.2	2.5	3.0	3.7	pr11	654	637	647.0	651	1.1	4.6	5.9	9.2
pr02	942	936	937.8	940	0.4	13.8	18.6	24.4	pr12	1002	971	975.2	979	2.7	39.2	61.0	75.6
pr03	1010	989	997.4	1003	1.2	25.6	29.9	38.7	pr13	1129	1095	1102.2	1106	2.4	58.6	77.0	102.8
pr04	1284	1257	1279.6	1294	0.3	51.8	106.5	177.7	pr14	1368	1345	1357.4	1369	0.8	80.8	130.9	178.5
pr05	1482	1453	1464.0	1480	1.2	120.9	154.9	213.1	pr15	1622	1571	1594.2	1609	1.7	187.3	246.6	327.7
pr06	1514	1477	1499.4	1512	1.0	146.0	188.9	226.8	pr16	1667	1619	1654.6	1668	0.7	299.1	413.5	613.3
pr07	744	728	736.6	744	1.0	8.9	11.8	15.1	pr17	832	816	822.4	832	1.2	12.3	15.2	19.6
pr08	1138	1105	1122.6	1132	1.4	25.7	44.8	56.3	pr18	1276	1266	1270.8	1281	0.4	52.8	63.5	76.4
pr09	1275	1255	1260.6	1268	1.1	89.1	102.6	109.7	pr19	1385	1361	1393.6	1417	−0.6	125.4	216.3	285.9
pr10	1549	1559	1563.4	1573	−0.9	139.3	198.0	225.8	pr20	1682	1673	1677.6	1684	0.3	186.7	277.4	362.5

Table 12
Detailed results of GVNS on Cordeau's instances with $m = 4$.

Instance	BK	GVNS			Gap (%)	Time (seconds)			Instance	BK	GVNS			Gap (%)	Time (seconds)		
		Min	Avg	Max		Min	Avg	Max			Min	Avg	Max		Min	Avg	Max
pr01	657	654	655.8	657	0.2	2.9	3.6	4.4	pr11	657	657	657.0	657	0.0	4.2	5.0	6.2
pr02	1079	1055	1064.2	1070	1.4	15.2	25.6	34.0	pr12	1132	1102	1110.2	1123	1.9	23.1	36.7	57.3
pr03	1222	1202	1209.6	1215	1.0	30.9	57.2	75.3	pr13	1346	1310	1324.8	1334	1.6	46.1	76.6	145.2
pr04	1557	1522	1525.8	1531	2.0	61.4	137.2	196.9	pr14	1660	1624	1636.6	1648	1.4	144.5	243.6	339.8
pr05	1833	1785	1811.4	1827	1.2	134.3	208.4	252.0	pr15	1958	1920	1933.4	1946	1.3	253.9	428.3	576.8
pr06	1854	1826	1840.4	1855	0.7	216.0	309.9	363.0	pr16	2021	1989	2000.0	2014	1.0	371.2	632.0	758.1
pr07	872	855	862.6	870	1.1	14.1	17.9	22.3	pr17	933	910	914.6	921	2.0	9.8	14.0	23.0
pr08	1382	1355	1358.2	1361	1.7	45.3	67.3	93.7	pr18	1521	1497	1505.4	1518	1.0	50.4	93.0	114.5
pr09	1592	1589	1595.4	1607	−0.2	149.3	171.1	186.3	pr19	1695	1671	1688.2	1723	0.4	230.6	334.1	396.4
pr10	1933	1894	1903.6	1916	1.5	246.6	275.1	310.1	pr20	2037	1999	2012.0	2019	1.2	359.0	463.1	577.5

Table 13
Comparison of the state-of-the-art methods on the first part of the new Solomon's instances.

Instance	m	Opt	GVNS			GRASP-ELS			VNS*			ILS		
			Avg	Gap (%)	Time (seconds)	Avg	Gap (%)	Time (seconds)	Avg	Gap (%)	Time (seconds)	z	Gap (%)	Time (seconds)
c101	10	*1810*	1754.0	3.1	27.0	1810.0	0.0	8.9	1809.0	0.1	21.7	1720	5.0	4.1
c102	10	*1810*	1794.0	0.9	23.2	1810.0	0.0	0.3	1810.0	0.0	19.8	1790	1.1	4.2
c103	10	*1810*	1810.0	0.0	5.7	1810.0	0.0	0.4	1810.0	0.0	18.8	1810	0.0	3.0
c104	10	*1810*	1810.0	0.0	1.6	1810.0	0.0	0.4	1810.0	0.0	16.8	1810	0.0	1.8
c105	10	*1810*	1810.0	0.0	1.6	1810.0	0.0	0.3	1810.0	0.0	22.7	1770	2.2	2.8
c106	10	*1810*	1806.0	0.2	9.1	1810.0	0.0	0.1	1808.0	0.1	23.5	1750	3.3	3.8
c107	10	*1810*	1810.0	0.0	1.1	1810.0	0.0	0.1	1810.0	0.0	21.9	1790	1.1	3.1
c108	10	*1810*	1810.0	0.0	0.7	1810.0	0.0	0.2	1810.0	0.0	19.6	1810	0.0	2.5
c109	10	*1810*	1810.0	0.0	0.1	1810.0	0.0	0.0	1810.0	0.0	16.1	1810	0.0	2.0
r101	19	*1458*	1432.2	1.8	23.4	1454.4	0.2	60.9	1455.8	0.2	2.5	1441	1.2	2.5
r102	17	1458	1441.2	1.2	20.7	1451.8	0.4	146.5	1451.1	0.5	3.1	1450	0.5	3.1
r103	13	*1458*	1446.6	0.8	23.5	1455.0	0.2	178.6	1453.3	0.3	2.0	1450	0.5	2.0
r104	9	*1458*	1418.2	2.7	55.0	1445.0	0.9	103.9	1443.5	1.0	2.3	1402	3.8	2.3
r105	14	*1458*	1441.6	1.1	32.3	1449.6	0.6	74.4	1450.1	0.5	4.1	1435	1.6	4.1
r106	12	*1458*	1437.6	1.4	25.3	1453.8	0.3	89.1	1451.9	0.4	3.1	1441	1.2	3.1
r107	10	*1458*	1435.0	1.6	46.0	1450.6	0.5	96.6	1449.7	0.6	3.3	1431	1.9	3.3
r108	9	*1458*	1441.8	1.1	56.4	1455.0	0.2	116.8	1453.2	0.3	2.7	1430	1.9	2.7
r109	11	*1458*	1433.4	1.7	42.8	1439.8	1.2	78.6	1448.3	0.7	2.5	1432	1.8	2.5
r110	10	*1458*	1433.4	1.7	50.1	1434.2	1.6	102.6	1450.5	0.5	4.4	1419	2.7	4.4
r111	10	*1458*	1430.2	1.9	40.0	1438.6	1.3	88.6	1449.2	0.6	3.0	1410	3.3	3.0
r112	9	*1458*	1434.4	1.6	58.7	1440.4	1.2	127.3	1451.9	0.4	2.4	1418	2.7	2.4
rc101	14	*1724*	1690.2	2.0	28.1	1705.2	1.1	65.4	1704.5	1.1	4.3	1686	2.2	4.3
rc102	12	*1724*	1685.0	2.3	30.1	1693.8	1.8	74.6	1696.0	1.6	2.9	1659	3.8	2.9
rc103	11	*1724*	1709.0	0.9	37.9	1716.4	0.4	129.3	1715.8	0.5	3.4	1689	2.0	3.4
rc104	10	*1724*	1718.0	0.3	43.6	1724.0	0.0	4.0	1722.7	0.1	3.2	1719	0.3	3.2
rc105	13	*1724*	1689.8	2.0	32.9	1694.4	1.7	90.4	1698.2	1.5	3.9	1691	1.9	3.9
rc106	11	*1724*	1690.6	1.9	40.5	1691.8	1.9	77.8	1694.8	1.7	4.8	1665	3.4	4.8
rc107	11	*1724*	1718.4	0.3	51.5	1724.0	0.0	66.4	1721.0	0.2	2.4	1701	1.3	2.4
rc108	10	*1724*	1713.0	0.6	51.1	1718.8	0.3	121.7	1721.8	0.1	5.6	1698	1.5	5.6
Average				1.1	29.7		0.5	65.7		0.4	8.5		1.8	3.2

Table 14
Comparison of the state-of-the-art methods on the second part of the new Solomon's instances.

Instance	*m*	Opt	GVNS			GRASP-ELS			ILS		
			Avg	Gap (%)	Time (seconds)	Avg	Gap (%)	Time (seconds)	*z*	Gap (%)	Time (seconds)
c201	4	*1810*	1810.0	0.0	0.1	1810.0	0.0	0.0	1810	0.0	1.5
c202	4	*1810*	1810.0	0.0	1.8	1810.0	0.0	0.0	1810	0.0	1.1
c203	4	*1810*	1810.0	0.0	1.5	1810.0	0.0	0.0	1810	0.0	1.0
c204	4	*1810*	1810.0	0.0	0.5	1810.0	0.0	0.0	1810	0.0	1.0
c205	4	*1810*	1810.0	0.0	0.1	1810.0	0.0	0.0	1810	0.0	1.0
c206	4	*1810*	1810.0	0.0	0.1	1810.0	0.0	0.0	1810	0.0	1.0
c207	4	*1810*	1810.0	0.0	0.1	1810.0	0.0	0.0	1810	0.0	1.0
c208	4	*1810*	1810.0	0.0	0.1	1810.0	0.0	0.0	1810	0.0	0.9
r201	4	*1458*	1458.0	0.0	1.4	1458.0	0.0	0.2	1458	0.0	1.3
r202	3	1458	1450.2	0.5	19.8	1456.2	0.1	19.2	1443	1.0	2.7
r203	3	*1458*	1458.0	0.0	2.8	1458.0	0.0	0.0	1458	0.0	1.5
r204	2	*1458*	1452.6	0.4	6.6	1458.0	0.0	6.1	1440	1.2	3.3
r205	3	*1458*	1458.0	0.0	0.4	1458.0	0.0	0.0	1458	0.0	1.1
r206	3	*1458*	1458.0	0.0	0.7	1458.0	0.0	0.0	1458	0.0	1.0
r207	2	*1458*	1449.8	0.6	12.4	1456.0	0.1	19.9	1428	2.1	1.6
r208	2	*1458*	1458.0	0.0	0.9	1458.0	0.0	0.1	1458	0.0	1.6
r209	3	*1458*	1458.0	0.0	0.2	1458.0	0.0	0.0	1458	0.0	1.1
r210	3	*1458*	1458.0	0.0	0.9	1458.0	0.0	0.1	1458	0.0	1.2
r211	2	*1458*	1452.6	0.4	14.6	1456.0	0.1	26.5	1422	2.5	1.9
rc201	4	*1724*	1724.0	0.0	1.9	1724.0	0.0	0.0	1724	0.0	2.2
rc202	3	*1724*	1702.8	1.2	10.3	1719.8	0.2	20.3	1686	2.2	1.7
rc203	3	*1724*	1724.0	0.0	2.2	1724.0	0.0	0.7	1724	0.0	2.8
rc204	3	*1724*	1724.0	0.0	0.1	1724.0	0.0	0.0	1724	0.0	1.0
rc205	4	*1724*	1723.0	0.1	3.2	1724.0	0.0	0.1	1724	0.0	2.1
rc206	3	*1724*	1724.0	0.0	2.5	1724.0	0.0	0.7	1708	0.9	1.3
rc207	3	*1724*	1724.0	0.0	2.2	1724.0	0.0	1.1	1713	0.6	1.4
rc208	3	*1724*	1724.0	0.0	0.2	1724.0	0.0	0.0	1724	0.0	1.0
Average				0.1	3.2		0.0	3.5		0.4	1.5

Table 15
Comparison of the state-of-the-art methods on the first part of the new Cordeau's instances.

Instance	*m*	Opt	GVNS			GRASP-ELS			VNS*			ILS		
			Avg	Gap (%)	Time (seconds)	Avg	Gap (%)	Time (seconds)	Avg	Gap (%)	Time (seconds)	*z*	Gap (%)	Time (seconds)
pr01	3	*657*	608.4	7.4	3.1	619.0	5.8	10.1	612.9	6.7	69.6	608	7.5	0.7
pr02	6	*1220*	1198.8	1.7	27.9	1202.2	1.5	81.9	1195.3	2.0	85.0	1180	3.3	4.5
pr03	9	*1788*	1760.8	1.5	50.2	1771.0	1.0	171.0	1759.0	1.6	86.0	1738	2.8	11.3
pr04	12	*2477*	2467.4	0.4	114.0	2475.6	0.1	341.2	2466.4	0.4	81.2	2428	2.0	45.4
pr05	15	*3351*	3351.0	0.0	43.2	3351.0	0.0	3.5	3349.5	0.0	71.7	3297	1.6	37.3
pr06	18	*3671*	3670.6	0.0	196.4	3671.0	0.0	31.5	3670.1	0.0	83.5	3650	0.6	106.1
pr07	5	*948*	935.0	1.4	13.8	938.8	1.0	35.4	928.3	2.1	69.5	909	4.1	1.5
pr08	10	*2006*	2004.6	0.1	46.9	2006.0	0.0	3.8	2005.0	0.0	63.2	1984	1.1	12.0
pr09	15	*2736*	2736.0	0.0	8.5	2736.0	0.0	33.9	2735.6	0.0	54.6	2729	0.3	33.0
pr10	20	*3850*	3850.0	0.0	9.4	3850.0	0.0	2.5	3850.0	0.0	52.4	3850	0.0	52.0
Average				1.2	51.3		0.9	71.5		1.3	71.7		2.3	30.4

particular data set. The results underline the fact that GVNS performs better on Cordeau's instances.

5. Conclusions and future developments

In this paper we analyze a variant of the Team Orienteering Problem taking hard time windows into account. We propose a Granular Variable Neighborhood Search approach exploring, most of the time, limited instead of complete neighborhoods. Granularity is based on dual information (reduced costs) provided by the optimal solution of an ad hoc LP problem (LP-granularity). Computational results have shown that proposed VNS approach is already an effective algorithm, whereas the introduction of the granularity can improve algorithm's efficiency while maintaining effectiveness. The algorithm performs, on average, quite well compared to state-of-the-art algorithms and has been able to improve 25 best known solution values among the most difficult instances.

The use of primal criteria may be an interesting alternative to reduced costs to control granularity. We did some preliminary experiments where the use of primal information does not seem to provide better results than the use of reduced costs. Nevertheless, we believe that this line of research deserves to be further explored to draw clear conclusions.

Finally, as future developments it is our intention to apply this procedure to the solution of a dynamic and more realistic version of the OPTW. Such dynamic problem has indeed motivated the introduction of the granular version of the proposed VNS.

Acknowledgments

We wish to thank anonymous reviewers for fruitful suggestions which help improve a previous version of this paper. Moreover, Renata Mansini and Roberto Wolfler Calvo would like to thank Marianna Pelizzari for the work she did for the preliminary version of this paper.

References

[1] H. Bouly, D.C. Dang, A. Moukrim, A memetic algorithm for the team orienteering problem, 4OR: A Quarterly Journal of Operations Research (2009) (published online).
[2] S. Boussier, D. Feillet, M. Gendreau, An exact algorithm for team orienteering problems, 4OR: A Quarterly Journal of Operations Research 5 (3) (2007) 211–230.
[3] R. Carraghan, P.M. Pardalos, An exact algorithm for the maximum clique problem, Operations Research Letters 9 (1990) 375–382.
[4] I.M. Chao, B.L. Golden, E.A. Wasil, A fast and effective heuristic for the Orienteering Problem, European Journal of Operational Research 88 (1996) 475–489.
[5] I.M. Chao, B.L. Golden, E.A. Wasil, The team orienteering problem, European Journal of Operational Research 88 (3) (1996) 464–474.
[6] J.F. Cordeau, M. Grendreau, G. Laporte, A tabu search heuristic for periodic and multidepot problems, Networks 30 (1997) 105–119.
[7] J.J. Dongarra, Performance of various computers using standard linear equations software in a FORTRAN environment, Technical Report of the Electrical Engineering and Computer Science Department, University of Tennessee, 2009. <http://www.netlib.org/benchmark/performance.ps>.
[8] D. Feillet, P. Dejax, M. Gendreau, Traveling Salesman Problem with profits: an overview, Transportation Science 39 (2005) 188–205.
[9] M. Gendreau, G. Laporte, F. Semet, The selective traveling salesman problem, Networks 32 (1998) 263–273.
[10] B.L. Golden, L. Levy, R. Vohra, The orienteering problem, Naval Research Logistics 34 (1987) 307–318.
[11] P. Hansen, N. Mladenović, Variable Neighborhood Search: principles and applications, European Journal of Operational Research 130 (2001) 449–467.
[12] M. Kantor, M. Rosenwein, The Orienteering Problem with Time Windows, Journal of the Operational Research Society (43/6) (1992) 629–635.
[13] L. Ke, C. Archetti, Z. Feng, Ants can solve the team orienteering problem, Computers Industrial Engineering 54 (3) (2008) 648–665.
[14] P.C. Keller, Algorithms to solve the orienteering problem: a comparison, European Journal of Operational Research 41 (1989) 224–231.
[15] G.A.P. Kindervater, M.W.P. Savelsbergh, Vehicle routing: handling edge exchanges, in: E. Aarts, J. Lenstra (Eds.), Local Search in Combinatorial Optimization, Wiley, Chichester, 1997, pp. 311–336.
[16] N. Labadie, J. Melechovský, R. Wolfler Calvo, An effective hybrid evolutionary local search for orienteering and team orienteering problem with time windows, in: R. Schaefer et al. (Eds.), Lecture Notes in Computer Science, vol. 6239, Springer, Berlin/Heidelberg, 2010, pp. 219–228.
[17] N. Labadie, J. Melechovský, R. Wolfler Calvo, Hybridized evolutionary local search algorithm for the team orienteering problem with time windows, Journal of Heuristics 17 (6) (2011) 729–753.
[18] G. Laporte, S. Martello, The selective traveling salesman problem, Discrete Applied Mathematics 26 (1990) 193–207.
[19] R. Mansini, M. Pelizzari, R. Wolfler Calvo, A Granular Variable Neighborhood Search for the Tour Orieentering Problem with Time Windows, Technical Report of the Department of Electronics for Automation, University of Brescia, RT 2006-02-52, 2006.
[20] N. Mladenović, P. Hansen, Variable Neighborhood Search, Computers and Operations Research 24 (1997) 1097–1100.
[21] C. Miller, A. Tucker, R. Zemlin, Integer programming formulations and traveling salesman problems, Journal of the Association for Computing Machinery 7 (1960) 326–329.
[22] R. Montemanni, L.M. Gambardella, An ant colony system for team orienteering problem with time windows, Foundations of Computing and Decision Sciences 34 (4) (2009) 287–306.
[23] R. Montemanni, D. Weyland, L.M. Gambardella, An enhanced ant colony system for the team orienteering problem with time windows, in: Proceedings of IEEE ISCCS 2011, International Symposium on Computer Science and Society, 2011, pp. 381–384, ISBN: 978-1-4577-0644-8.
[24] G. Righini, 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 36 (4) (2009) 1191–1203.
[25] M. Solomon, J. Desrosiers, Time windows constrained routing and scheduling problems, Transportation Science 22 (1998) 1–22.
[26] W. Souffriau, P. Vansteenwegen, G.V. Berghe, D.V. Oudheusden, A path relinking approach for the team orienteering problem, Computers and Operations Research 37 (11) (2010) 1853–1859.
[27] H. Tang, E. Miller-Hooks, A tabu search heuristic for the team orienteering problem, Computers and Operations Research 32 (6) (2005) 1379–1407.
[28] P. Toth, D. Vigo, The Granular Tabu Search and its application to the Vehicle Routing Problem, INFORMS Journal on Computing 15 (4) (2003) 333–346.
[29] F. Tricoire, M. Romauch, K.F. Doerner, R.F. Hartl, Heuristics for the multi-period orienteering problem with multiple time windows, Computers and Operations Research 37 (2) (2010) 351–367.
[30] F. Tricoire, M. Romauch, K.F. Doerner, R.F. Hartl, Addendum to the paper "Heuristics for the multi-period orienteering problem with multiple time windows", 2010. <http://prolog.univie.ac.at/research/op/>.
[31] T. Tsiligrides, Heuristic methods applied to orienteering, Journal of Operational Research Society 35 (1984) 797–809.
[32] P. Vansteenwegen, W. Souffriau, G.V. Berghe, D.V. Oudheusden, A guided local search metaheuristic for the team orienteering problem, European Journal of Operational Research 196 (1) (2009) 118–127.
[33] P. Vansteenwegen, W. Souffriau, G.V. Berghe, D.V. Oudheusden, Iterated local search for the team orienteering problem with time windows, Computers and Operations Research 36 (12) (2009) 3281–3290.
[34] P. Vansteenwegen, W. Souffriau, G.V. Berghe, D.V. Oudheusden, Metaheuristics for tourist trip planning, in: K. Srensen, M. Sevaux, W. Habenicht, M.J. Geiger (Eds.), Metaheuristics in the Service Industry, Lecture Notes in Economics and Mathematical Systems, vol. 624, Springer, Berlin/Heidelberg, 2009, pp. 15–31.
[35] P. Vansteenwegen, W. Souffriau, D.V. Oudheusden, The orienteering problem: a survey, European Journal of Operational Research 209 (1) (2011) 1–10.