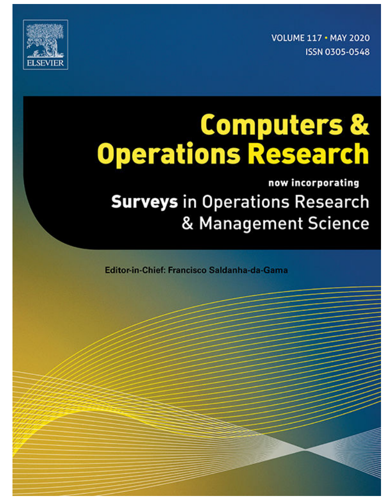
Journal Pre-proofs



A hybrid adaptive large neighborhood search heuristic for the team orienteering problem

Farouk Hammami, Monia Rekik, Leandro C. Coelho

PII: S0305-0548(20)30151-9
DOI: https://doi.org/10.1016/j.cor.2020.105034
Reference: CAOR 105034

To appear in: *Computers and Operations Research*

Received Date: 4 November 2019
Accepted Date: 11 June 2020

Please cite this article as: F. Hammami, M. Rekik, L.C. Coelho, A hybrid adaptive large neighborhood search heuristic for the team orienteering problem, *Computers and Operations Research* (2020), doi: https://doi.org/10.1016/j.cor.2020.105034

A hybrid adaptive large neighborhood search heuristic for the team orienteering problem

Farouk Hammami[a,b,*], Monia Rekik[a,b], Leandro C. Coelho[a,b]

[a] *Laval University, Department of Operations and Decision Systems, Quebec, Canada*
[b] *Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT), Canada*

Abstract

The Team Orienteering Problem (TOP) is a well-known NP-Hard vehicle routing problem in which one maximizes the collected profits for visiting some nodes. In this paper, we propose a Hybrid Adaptive Large Neighborhood Search (HALNS) to solve this problem. Our algorithm combines the exploration power of ALNS with local search procedures and an optimization stage using a Set Packing Problem to further improve the solutions. Extensive computational experiments demonstrate the high performance of our HALNS outperforming all the competing algorithms in the literature on a large set of benchmark instances in terms of solution quality and/or computational time. Our HALNS identifies all the 387 Best Known Solutions (BKS) from the literature on a first dataset including small-scale benchmark instances and all the 333 BKS for large-scale benchmark instances within very short computational times. Moreover, we improve one large-scale instance solution.

Keywords: Team Orienteering Problem, Adaptive Large Neighborhood Search, Hybrid Heuristic, Homogeneous fleet, Vehicle Routing Problem

*Corresponding author
Email address: `Farouk.Hammami@cirrelt.ca` (Farouk Hammami)

1. Introduction

The Team Orienteering Problem (TOP) is a routing problem with profits involving multiple vehicles and is a variant of the Vehicle Routing Problem (VRP) (Archetti et al., 2014). The aim of the TOP is to maximize the profit accumulated by a set of vehicles while visiting some locations. Each vehicle starts its route from a depot node and finishes at a different depot node within a predefined time limit. The vehicle collects a profit associated with each node visited, which is visited at most once. The problem was introduced by Butt and Cavalier (1994) as the *Multiple Tour Maximum Collection Problem*, also known as the vehicle routing problem with profits, and Chao et al. (1996) coined the term TOP.

The OP is a special case of the TOP which consists of a single vehicle problem. The OP was introduced by Golden et al. (1987) and is also known as the Selective Travelling Salesman Problem (STSP) (Laporte and Martello, 1990), the maximum collection problem (Butt and Cavalier, 1994) or the bank robber problem Arkin et al. (1998). Surveys about the OP can be found in Feillet et al. (2005) and Laporte and Martín (2007). Vansteenwegen et al. (2011) elaborate a survey on the OP and cover its variants such as the TOP and the TOP with Time Windows (TOPTW), describing formulations and solution algorithms. Later, Gunawan et al. (2016) extended this survey covering more recent papers including new variants of the OP such as the Arc OP (Archetti and Speranza, 2015; Archetti et al., 2016), the Team Orienteering Arc Routing Problem (TOARP) (Archetti et al., 2013, 2015), the OP with stochastic profits (OPSP) (Ilhan et al., 2008; Evers et al., 2014), and the clustered OP (COP) (Angelelli et al., 2014). Other OP variants from the literature are discussed in Vansteenwegen and Gunawan (2019a). Recently, Vansteenwegen and Gunawan (2019b) surveyed the benchmark instances and some of state-of-the-art exact and heuristic algorithms for both OP and TOP.

In this paper we propose a Hybrid Adaptive Large Neighborhood Search (HALNS) algorithm to solve the TOP. Our proposed algorithm combines the exploration power of ALNS and different local search procedures to speed up the solution process. We also design a new sub-route optimization procedure to improve solution obtained by ALNS. Promising ones discovered during the search process are then passed to a Set Packing Problem (SPP) in an attempt to further improve the HALNS solution. Our heuristic is hybridized by addressing the Sub-Route Optimization Problem (SROP) and the SPP which are solved via the branch-and-cut procedure of a commercial solver. HALNS is evaluated on two sets of instances: a set of small-scale instances proposed by Chao et al. (1996) and a set of large-scale instances proposed by Dang et al. (2013b). The obtained results are compared to different algorithms from the literature. Our results show that HALNS outperforms all 26 existing state-of-the-art heuristics in terms of solution quality and/or computational time for the small and large-scale instances of the TOP. To the best of our knowledge, only two methods from the literature have been tested on the large instances. Here, our algorithm finds all the Best Known Solutions (BKS) in shorter computing time for the majority of the instances, and we report a new improved solution value for one large-scale instance.

The remainder of this paper is organized as follows. Section 2 presents a literature review of the TOP covering different exact and heuristic solution approaches. In Section 3 we propose a MIP mathematical formulation for the TOP. In Section 4, we describe the HALNS and its different features. Section 5 presents the results obtained by our HALNS and compare them to those reported by other methods in the literature. Section 6 concludes the paper and offers insights for future research.

2. Literature review

The TOP is one of the most studied problems in the context of routing with profits (Archetti et al., 2014). Several exact and heuristic solution approaches have been proposed to solve the single vehicle version of the problem, the OP, since it has been proved to be NP-hard (Laporte and Martello, 1990).

Boussier et al. (2007) were the first to propose an exact method to solve the TOP. The authors proposed a branch-and-price approach based on an SPP formulation with special branching rules tailored to the OP. Computation results show that the proposed approach is able to solve to optimality 270 out of the 387 small-scale benchmark instances already proposed by Chao et al. (1996). The method was also adapted to solve the TOPTW. Poggi de Aragão et al. (2010) proposed three different formulations for the TOP. The authors developed a robust branch-cut-and-price algorithm to solve the problem and used two different cuts in their algorithm (Min Cut inequalities and Triangle Clique cuts) inspired from the work of Pessoa et al. (2009).

Dang et al. (2013a) introduced a branch-and-cut algorithm to solve a three-index mathematical formulation with a polynomial number of binary variables for the TOP. The method is based on a set of valid inequalities and dominance criteria. The authors were able to prove optimality for 29 previously open small-scale instances. Later, Keshtkaran et al. (2016) developed a branch-and-price approach, based on that proposed by Boussier et al. (2007). They also developed the first branch-cut-and-price explicitly designed for the TOP. The proposed algorithm was able to identify 17 new optimal solutions for the small-scale benchmark instances in addition to instances already solved by the previous exact methods.

El-Hajj et al. (2016) investigate the use of a linear formulation with a polynomial number of variables to solve the TOP. The authors proposed an exact algorithm based on a cutting-plane approach and added several types of cuts to strengthen the classical linear formulation. Adding cuts dynamically during the solution process was confirmed to be effective when tested on small-scale benchmark instances and yielded 12 new optimal solutions.

Bianchessi et al. (2018) presented a new two-index formulation with a polynomial number of variables and constraints for the TOP. The authors reinforced the proposed formulation with a set of connectivity constraints and solved the problem by branch-and-cut. The developed solution approach was compared to all the previous exact algorithms. Their branch-and-cut solved to optimality 24 previously open small-scale instances.

Recently, Pessoa et al. (2019) proposed a branch-cut-and-price solver for a generic model which encompasses some VRPs variants, including the TOP. The authors reported that their algorithm incorporates the key elements present in the recent VRP solution approaches. The conducted computational experiments show that the developed solver outperforms the approach of Bianchessi et al. (2018) when tested on a set of 60 small-scale instances of the TOP in terms of computational time. Pessoa et al. (2019) reported 56 optimal solutions in less than two hours of computational time.

More recently, Orlis et al. (2020) introduced a new variant of the TOP: the TOP with Overlaps (TOPO), where each node can be serviced via a set of service points. The authors developed an exact branch-and-cut-and-price and a Large Neighborhood Search (LNS) metaheuristic to solve the problem. When applied to the small-scale benchmark instances of the TOP, the exact procedure was able to prove optimality for 371 out of 387 instances within one hour of time limit.

Although exact solution approaches allowed to solve 371 out of the 387 small-scale benchmark instances, they remain very time- and resource-consuming given the complexity of the problem even with small-scale instances. Moreover, none of the exact methods was tested on the large-scale benchmark instances. In order to solve the TOP faster with less computational resources, several heuristics have been proposed, as reviewed next.

Chao et al. (1996) were the first to propose a heuristic approach to solve the TOP. The authors developed a fast and effective heuristic based on the notion of record-to-record improvement and compared its performance against a modified heuristic developed by Tsiligirides (1984), which was initially designed to solve the OP.

Later, Tang and Miller-Hooks (2005) proposed a tabu search heuristic embedded in an adaptive memory procedure (Rochat and Taillard, 1995) that alternates between small and large neighborhood stages during a solution improvement phase. Computational results show that this method outperformed the results at that time.

Archetti et al. (2007) proposed two variants of a generalized tabu search algorithm as well as a slow and a fast Variable Neighborhood Search (VNS) algorithm. The first tabu search procedure only considers feasible solutions, while the second accepts infeasible ones. Computation results showed that these heuristics outperform the algorithm proposed by Tang and Miller-Hooks (2005) in terms of solution quality, with the VNS being the most efficient one.

Ke et al. (2008) presented an Ant Colony Optimization (ACO) approach developed for the TOP. Four algorithms were proposed to construct candidate solutions in their framework. These are the sequential, deterministic-concurrent, random-concurrent, and simultaneous methods. By comparing the four variants of ACO the authors showed that the sequential one obtained the best solution quality within less than one minute for each instance.

Vansteenwegen et al. (2009b) proposed a Skewed VNS (SkVNS) using a combination of heuristics to efficiently solve the TOP. The obtained results are comparable to the results of the best known heuristics. Later, Vansteenwegen et al. (2009a) described an algorithm combining different local search procedures to solve the problem. Guided Local Search (GLS) is used to improve two local search heuristics. Although the GLS results are almost the same as the best known ones, it significantly reduces the computational time.

Souffriau et al. (2010) designed two variants of a Greedy Randomized Adaptive Search Procedure (GRASP) with Path Relinking for the TOP. The authors tested a fast variant of the method (FPR) and a slow variant (SPR) which yields much better results. According to numerical results, the solution quality of SPR is comparable to that of state-of-the-art heuristics.

Bouly et al. (2010) proposed a simple hybrid genetic algorithm using new algorithms dedicated to the specific scope of the TOP. Their Memetic Algorithm (MA) exploits an optimal split procedure for chromosome evaluation and local search techniques for mutation. The reported results showed that this evolutionary algorithm is competitive when compared to state-of-the-art heuristics; however, it may require up to 357.05 seconds to solve some instances, which is much more than some of the competing algorithms.

Dang et al. (2011) presented a Particle Swarm Optimization-based MA (PSOMA) for the TOP. Computational results showed that it outperformed the previous MA in terms of computational time and solution quality with a reported average gap of 0.016% to the BKS. Lin (2013) designed a Multi-start Simulated Annealing (MSA) algorithm which combines a Simulated Annealing (SA) based metaheuristic with a multi-start hill-climbing strategy to solve the TOP. Numerical results showed that MSA obtained five new best solutions. Starting with this paper, many papers only test 57 instances out of the 387 available ones as all methods find the same (proven optimal) solution.

Kim et al. (2013) proposed an augmented large neighborhood search (AuLNS) method with three improvement algorithms: local search improvement, shift and insertion, and replacement. The proposed solution approach was able to identify 386 of the best known solutions for the 387 benchmark instances proposed by Chao et al. (1996), outperforming all previous algorithms.

Dang et al. (2013b) proposed a PSO-inspired Algorithm (PSOiA) based on their previous study (Dang et al., 2011). The authors stated that the main contribution lies on a faster evaluation process than the one proposed in Bouly et al. (2010). Reported computation results showed that this heuristic outperforms other methods. Furthermore, Dang et al. (2013b) proposed a new package of large-scale instances with up to 401 nodes to test the performance of their PSOiA. Numerical results for this set of large instances showed that the PSOiA requires up to 96,187.70 seconds to solve these instances with an average computational time of 11,031.04 seconds.

Ferreira et al. (2014) proposed a genetic algorithm for which computational results obtained BKS in more than half of tested instances; however, one should note that tests were only conducted on 20 of the 387 available benchmark instances, and they did not include the large benchmark set.

Vidal et al. (2016) proposed some new neighborhood search for the VRP with profits. These neighborhoods were integrated within three heuristic frameworks, the Unified Hybrid Genetic Search (UHGS) procedure of Vidal et al. (2014), a local improvement heuristic, and the Iterated Local Search (ILS) of Prins (2009), on the TOP. The computational experiments were conducted on the small-scale instances and showed that the simple local search improvement method with the proposed neighborhoods obtains solutions with similar quality to most current state-of-the-art heuristics.

Ke et al. (2016) proposed a new algorithm called Pareto Mimic Algorithm (PMA) for the TOP. One of the main features of this algorithm is the swallow operator which inserts unvisited nodes based on the largest dynamic preference value. The authors report that this new oprator improved solution quality for large-scale instances. Numerical results show that PMA outperforms all the previous state-of-the-art methods when tested on the packages of small- and large-scale benchmark instances. Moreover, the authors extended the PMA to solve the Capacitated Vehicle Routing Problem (CVRP).

Recently, Tsakirakis et al. (2019) proposed a Similarity Hybrid Harmony Search (SHHS) algorithm as a solution approach for the TOP. Two versions of the method have been developed and tested. The first variant is static with predefined values of the parameters, and the second one contains a dynamic adjustment of the parameters. Computational results showed the performance of the second variant outperforms the first one. However, the second version of the proposed solution approach reached the BKS for only 84% of the instances.

Overall, among the 387 small-scale benchmark instances available in seven different sets, 26 solution approaches have provided results for some of them. The best results come mainly from the works of Dang et al. (2013b), Kim et al. (2013) and Ke et al. (2016). Optimality is known for 371 of these instances. On the large-scale benchmark instances set, BKS are provided by Ke et al. (2016). To the best of our knowledge and to date, no optimal solution is reported for any of the large-scale benchmark instances.

3. Problem definition and mathematical formulation

We consider a directed graph $G = (V, A)$ where $V = \{1, \ldots, N\} = V^* \cup \{1, N\}$ represents the set of nodes and nodes 1 and N represent respectively the start and end depot. The set of arcs is defined as $A = \{(i, j) : i \neq j, i \in V^*, j \in V^*\} \cup \{(1, j) : j \in V^*\} \cup \{(i, N) : i \in V^*\}$. To each arc $(i, j) \in A$ is associated a travel time t_{ij}. Each node $i \in V^*$ has an associated profit p_i. L denotes the set of vehicles. All vehicles are identical and must respect a maximum route duration D_{max}. All vehicles must be used and each one starts at the start depot and ends its route at the end depot.

We propose to model the TOP with three sets of variables: (1) binary variables x^l_{ij} determining if arc $(i, j) \in A$ is traversed by vehicle $l \in L$, (2) continuous variables B_i for each node $i \in V$ and each vehicle $l \in L$ indicating the order of visit of node i, and (3) binary variables y^l_i for each node $i \in V^*$ and each vehicle $l \in L$ indicating whether the node i is visited by vehicle l. The mathematical model for the TOP, denoted M_{TOP}, is inspired from Vansteenwegen et al. (2011) which is basically the formulation of Tang and Miller-Hooks (2005). M_{TOP} can be formulated as follows:

$$M_{TOP} : \max \sum_{l \in L} \sum_{i \in V^*} p_i y^l_i \tag{1}$$

$$\text{s.t.} \quad \sum_{l \in L} \sum_{\substack{j \in V \\ (i,j) \in A}} x^l_{ij} \leq 1 \qquad \forall i \in V^* \tag{2}$$

$$\sum_{l \in L} \sum_{\substack{j \in V \\ j \neq 1}} x^l_{1j} = \sum_{l \in L} \sum_{\substack{i \in V \\ i \neq N}} x^l_{iN} = |L| \tag{3}$$

$$\sum_{\substack{j \in V \\ (i,j) \in A}} x^l_{ji} = \sum_{\substack{j \in V \\ (i,j) \in A}} x^l_{ij} = y^l_i \qquad \forall l \in L, i \in V^* \tag{4}$$

$$\sum_{(i,j)\in A} t_{ij}x_{ij}^l \leq D_{max} \qquad \forall l \in L \tag{5}$$

$$\sum_{l\in L}\sum_{(i,j)\in A} t_{ij}x_{ij}^l \leq |L|D_{max} \tag{6}$$

$$\sum_{i\in S}\sum_{j\in S: j\neq i} x_{ij}^l \leq |S|-1 \qquad \forall l \in L, S\subseteq V^*, |S|\geq 2 \tag{7}$$

$$x_{ij}^l \in \{0,1\} \qquad \forall (i,j)\in A, l\in L \tag{8}$$

$$y_i^l \in \{0,1\} \qquad \forall i \in V^*, l \in L. \tag{9}$$

The objective function (1) maximizes the total collected profit. Constraints (2) allow nodes to be served at most once. Constraint (3) implies that each route starts at node 1 and ends at node N and that each vehicle must be used. Flow conservation and links between variables x_{ij}^l and y_i^l are ensured via constraints (4). Constraints (5) impose maximum tour length. Constraint (6) is proposed by Bianchessi et al. (2018) and is imposed on the global duration of routes to strengthen the formulation. Observe that constraint (6) is the sum of those in (5). Constraints (7) are the standard subtours elimination constraints. Finally, constraints (8)–(9) define the domain of the decision variables.

4. Solution approach

We propose a HALNS to solve the TOP described in Section 3. The ALNS is an extension of the Large Neighborhood Search (LNS) developed by Shaw (1998) for VRPs. It is proved to be efficient when used to solve several variants of the VRP. Ropke and Pisinger (2006) and Pisinger and Ropke (2007) used this heuristic to tackle several variants of the VRP namely the Pickup and Delivery with Time Windows (PDPTW), the VRP with time windows, the CVRP, the multi-depot VRP, the open VRP, and the site-dependent VRP. Furthermore, several other routing problems exploited the ALNS such as the multi-PDPTW (Naccache et al., 2018), the two-echelon VRP (Hemmelmayr et al., 2012), the pollution routing problem (Demir et al., 2012), the VRP with drones (Sacramento et al., 2019), the multi-depot open VRP (Lahyani et al., 2019). Within the ALNS framework, the heuristic starts with an initial solution and tries to improve its value by applying removal and insertion operators. Applying these operators can be seen as a move that defines a very large neighborhood search (Li et al., 2016).

Our hybrid ALNS is inspired by many of these works, but we define some modifications and new features to deal with the TOP. The first feature we propose is the use of what we call a *node selection strategy* to select at each iteration which nodes to try to insert given that all nodes yield a profit when inserted in a route. The principle of the *node selection strategy* is to select nodes to be inserted independently of the selected insertion operator. In fact, the TOP has the particularities to be a VRP variant in which no node is mandatory to be visited (except the depots) and no traveling cost is associated for serving a node. The second feature we propose is the use of an efficient local search procedure in order to optimize each improved solution. Third, we propose to hybridize the ALNS by addressing a SROP where the objective is to find a more profitable sequence of nodes to replace a less profitable one. Observe that this sub-problem corresponds to an OP. Finally, we hybridize again the ALNS by solving a SPP where the objective is to find the combination of the best routes obtained during the search process. The SROP and the SPP are solved exactly by a commercial solver. It is important to mention that our HALNS uses a SA acceptation criterion on the basis of a temperature varying over the algorithm iterations (see van Laarhoven and Aarts (1987) for details about the SA).

The general structure of our HALNS is sketched in Algorithm 1 and its main components are detailed next. Our algorithm starts by eliminating nodes that cannot be visited in order to reduce the size of the problem and thus the computational time. This simple and efficient elimination procedure is proposed by El-Hajj et al. (2016): on the basis of the travel time matrix (t_{ij}), it eliminates nodes that cannot be serviced. A node is considered inaccessible if by serving it the route time limit D_{max} is exceeded: node $i \in V^*$ is eliminated from the problem if and only if $t_{1i} + t_{iN} > D_{max}$.

Algorithm 1 Structure of the HALNS heuristic for the TOP

Apply a node elimination procedure
Construct initial solution s using the nearest neighbor algorithm
$s_{best} \leftarrow s$, $s_{adm} \leftarrow s$, $T \leftarrow T_0$, $seg \leftarrow 0$, $iteration_{best} \leftarrow 0$
Initialize the scores of the node selection strategy and the removal and insertion operators
while $(seg < N_{seg}$ and $iteration_{best} < iteration_{best_{max}})$ **do**
  $iteration \leftarrow 0$
  while $(iteration < iteration_{max})$ **do**
    $s \leftarrow s_{adm}$
    Generate β
    Select a node selection strategy: γ
    Select a removal and an insertion operators: R and I
    Remove β nodes from s using R
    Insert nodes in s using I following γ
    Generate a random number $\delta \in]0, 1[$
    if $\left(f(s) \geq f(s_{adm}) \text{ or } \delta \leq e^{\frac{f(s)-f(s_{adm})}{T}}\right)$ **then**
      if $(f(s) > f(s_{adm}))$ **then**
        Apply local search procedures on s
      end if
      if $(f(s) > f(s_{best}))$ **then**
        $s \leftarrow$ Generate and solve a SROP
        $s_{best} \leftarrow s$, $iteration_{best} \leftarrow 0$
      else
        $iteration_{best} \leftarrow iteration_{best} + 1$
      end if
      $s_{adm} \leftarrow s$
    else
      $iteration_{best} \leftarrow iteration_{best} + 1$
    end if
    if $(T \leq T_{min})$ **then**
      $T \leftarrow T_0$
      Generate and solve a SPP
      if $(f(s) > f(s_{best}))$ **then**
        $iteration_{best} \leftarrow 0$
      end if
      $s_{best} \leftarrow s$, $s_{adm} \leftarrow s$
    end if
    $T \leftarrow T \times c$, $iteration \leftarrow iteration + 1$
  end while
  Update scores of the ALNS operators and insertion strategies
  $seg \leftarrow seg + 1$
end while
Generate and solve a SPP
Update and return s_{best}

After reducing the size of the problem, an initial solution is constructed using the nearest neighbor algorithm (Keller et al., 1985). Line 3 of Algorithm 1 initialize the best solution s_{best}, the admissible solution s_{adm}, the SA current temperature T with an initial value T_0 and the run segments counter (seg). While the stopping criteria is not met, the algorithm iterates the following procedure.

First, the number of nodes β to be removed is determined randomly taking into account the number of inserted nodes within the current solution s. Node selection strategy γ, removal (R) and insertion (I) operators are then selected based on their past performances, modeled by scores. At the end of each run segment of size $iteration_{max}$, these scores are updated. The adaptive selection of the insertion strategy and the removal/insertion operators is described in Section 4.1.

At each iteration, a current admissible solution s_{adm} is modified into a current solution s as follows. The selected removal operator R removes β nodes from s. Then, the selected insertion operator I tries to insert, following the node selection strategy γ the non-inserted nodes into s in order to improve its profit. Node selection strategies and removal/insertion operators are described in Section 4.2. The modified solution is accepted if $f(s) \geq f(s_{adm})$ or it satisfies a SA criterion (line 15): it is accepted following a probability $e^{\frac{f(s)-f(s_{adm})}{T}}$ where $T > 0$ denotes the current temperature. T is decreased at the end of each iteration by a predefined cooling factor $c \in]0, \ldots, 1[$. If a solution s improves the last admissible solution s_{adm}, a local search procedure is performed to further improve it. Our local search algorithms are described in Section 4.3. If the current solution s improves the best solution s_{best}, a SROP is generated and solved using the branch-and-cut procedure of CPLEX. The sub-route optimization procedure is described in Section 4.4.

The search process is stopped when the maximum number of run segments (N_{seg}) is reached or the solution has not been improved for a given number of iterations ($iteration_{best_{max}}$). Finally, a SPP including all the routes generated during the search is solved using a branch-and-cut procedure at the end of the algorithm and each time $T \leq T_{min}$, where

T_{min} denotes the SA minimum temperature. s_{adm} and s_{best} are updated each time the SPP is solved. The SPP model is described in Section 4.5.

4.1. Adaptive selection of node selection strategies and removal/insertion operators

At the start of each iteration of the ALNS, a node selection strategy and removal and insertion operators are selected on the basis of their past performance. The principle of selecting the best operator to apply during a solution search procedure is known as the Adaptive Operator Selection (AOS) (Fialho et al., 2008; Maturana et al., 2009; Li et al., 2013) and is used in several solution methods such as hyper-heuristics (Özcan et al., 2008; Burke et al., 2013) and the ALNS (Ropke and Pisinger, 2006).

In our solution approach, each node selection strategy and removal/insertion operator k has a score $\pi_{k,q}$, indicating how well k performed recently, in each run segment q (a finite number of iterations) and set to one at the first iteration of the algorithm. Following that, $\pi_{k,q}$ is updated at the the end of a run segment q as $\pi_{k,q+1} = \lambda \frac{\overline{\pi_{k,q}}}{n_k} + (1-\lambda)\, \pi_{k,q}$ where n_k is the number of times the node selection strategy or operator k has been selected during the run segment q. Observe that $\overline{\pi_{k,q}}$ indicates the observed score of k for the run segment q and $\lambda \in \,]0,1[$ denotes a predefined reaction factor to adjust node selection strategies and operators weights. The observed score $\overline{\pi_{k,q}}$ is initialized to zero at the start of each run segment and is incremented at each iteration by a predefined parameter ρ which may take three different values:

$$\rho = \begin{cases} \rho_1 & \text{if the new solution value is a new best,} \\ \rho_2 & \text{if the new solution value is better than the last admissible one,} \\ \rho_3 & \text{if the new solution value does not improve the last admissible solution,} \end{cases}$$

The node selection strategies and removal/insertion operators are selected on the basis of a roulette wheel selection mechanism which is based on the strategy/operator score. The probability of selecting a strategy/operator k in run segment q is $\frac{\pi_{k,q}}{\sum_{k'=0}^{m} \pi_{k',q}}$ where m denotes the number of considered strategies/operators within the algorithm.

4.2. Node selection strategies and removal/insertion operators

The ALNS is an efficient heuristic often applied to several VRPs in which all nodes must be visited. Contrarily to classical VRPs, the TOP has the particularity that nodes are served only if they are profitable. Hence, it is necessary to make modifications to the removal and insertion operators proposed in the literature and to design new ones.

4.2.1. Node selection strategies

Since the TOP has the particularity of not taking into account traveling costs, a node inserted in any route yields the same profit. Four strategies are proposed to determine which nodes to insert first.

1. *Dynamic profit per travel time incremental selection*: this strategy is newly designed. Given a solution s, we compute for each non-inserted node i the ratio between its profit p_i and the total network travel time denoted $\mathfrak{D}(s^{+i})$ if i is inserted in the best position in s that minimizes the total travel time. Formally, node i^* to be first inserted is such that:
$$i^* := \arg\max_{i \in V^*} \frac{p_i}{\mathfrak{D}(s^{+i})}.$$
2. *Highest profit selection*: this strategy prioritizes the selection of nodes with highest profits to be inserted first. In order to prevent cycling, a roulette wheel mechanism is used so that the more profitable a node is the probability of selecting it by the roulette wheel algorithm is also larger.
3. *Random selection*: this strategy randomly selects the node to insert and is used to diversify the search.
4. *Last removed first inserted selection*: this strategy starts by selecting nodes to insert following the “Last removed, first inserted” (LRFI) rule. The aim of this strategy is to prioritize the insertion of nodes that have been removed the last and give them a chance to get inserted into better positions than their positions.

4.2.2. Removal operators

1. *Random removal*: this operator randomly selects nodes to be removed from the current solution. This operator is used in order to diversify the search.
2. *Lowest profit removal*: this operator is used to remove β nodes with the smallest profits. In order to prevent cycling, a roulette wheel is used to select nodes to be removed. This procedure gives high probabilities to remove nodes having the lowest profits.
3. *Largest saving in traveling time*: this newly designed operator is inspired *by the Largest saving in traveling cost* operator proposed by Hammami et al. (2019). For each node i visited by the current solution s, the algorithm computes the total traveling time if i is removed from s, denoted by $\mathfrak{D}(s^{-i})$. Then, the operator removes the node which maximizes $\mathfrak{D}(s) - \mathfrak{D}(s^{-i})$. A roulette wheel then gives larger probability to remove nodes inducing larger savings in total travel time.
4. *Route removal*: the aim of using this operator is to diversify the search. It randomly selects a vehicle and removes all nodes served by it.
5. *Sequence removal*: the idea of this operator is inspired from the related removal operator described by Pisinger and Ropke (2007). It removes a sequence of connected nodes from a randomly selected route. The motivation for removing a sequence of nodes served by the same route is to create a large slot to serve a non-inserted sequence of nodes which may be more profitable.

4.2.3. Insertion operators

1. *First available position insertion*: this newly designed operator inserts nodes in the first feasible position in a route, one node at a time. A position is feasible if the route resulting from inserting the node in this position respects the maximum duration constraint. In order to prevent cycling, our algorithm shuffles the order of routes.
2. *Last available position insertion*: this operator is similar to the previous one and inserts nodes in the last feasible position in a route, one node at a time.

3. *Random available position insertion*: this operator starts by checking for each node all the feasible positions in all routes then inserts the node in a feasible position randomly chosen.
4. *Best overall position*: this operator inserts each non-inserted node within a feasible position that minimizes the total travel time.
5. *Best position insertion*: this operator inserts each node within the feasible route position minimizing the sum of travel time between the node and its predecessor and between the node and its successor within the route.

4.3. Local search procedures

In our solution approach, we sequentially apply four local search heuristics for each new improved admissible solution. First, we start by applying a complete 2-opt procedure to each route from the current solution s to reduce the corresponding total traveling time (Croes, 1958). The 2-opt procedure enables to create more slots within routes so more nodes could be inserted in the following steps. Second, we randomly select non-inserted nodes and try to insert them within the current solution using our "Best position insertion" operator. A third local search heuristic randomly selects two inserted nodes i and j and one non-inserted node k, removes the first two and tries to insert k then i and j in order to improve the solution value. Finally, we randomly select two inserted nodes served by different routes and swap them in order to reduce the total travel time. If the solution is not improved after these moves, it gets rejected and the algorithm returns the best solution identified so far.

4.4. Sub-route optimization procedure

We designed a new sub-route optimization procedure to improve the value of each newly obtained best solution. The principle of this procedure consists in finding the best node sequence which can be inserted between two inserted nodes and replace the sequence already inserted. If such sequence exists, can be inserted and improves the profit, it replaces the old inserted one.

Given a solution s, we define V_+ as the set of inserted nodes and V_- as set of non-inserted nodes ($V_+ \cup V_- = V^*$). Let seq denote a sequence of α^l nodes served by vehicle l within s. We define $\mathfrak{D}_{seq}$ as the required time to serve the nodes forming seq. Let o and d denote respectively the first and last node of the sequence. Here, the algorithm starts by removing the nodes between o and d forming seq. The SROP, corresponding to an OP, is then formulated using model (1)-(9) with $|L| = 1$, start and end depot are o and d, $D_{max} = \mathfrak{D}_{seq}$, and V^* includes the nodes in V_- and those between o and d forming seq.

Observe that enumerating all subtour elimination constraints (7) results in a large number of constraints which slows down the solution procedure and can be prohibitive for large instances. As known in solving VRPs, these constraints may be relaxed then added when required, i.e., when a subtour is detected during the solution procedure (El-Hajj et al., 2016). To do so, we solve the SROP with an exact branch-and-cut procedure implementing a callback feature (available in the modern solvers). At first, we relax constraints (7); if the current integer solution does not contain a subtour then it is optimal, otherwise, each detected subtour is eliminated by adding the corresponding constraints. This process is repeated until no subtour is detected.

4.5. Set packing problem

The idea of combining routes generated during the search procedure to obtain a high quality solution by addressing a SPP or Set Covering Problem as a post-processing step is exploited by several heuristic methods to solve VRP variants (Desrochers et al., 1992; Renaud et al., 1996; Alvarenga et al., 2007; Hammami et al., 2019). In our HALNS, the SPP considers a set of routes generated during the search procedure and optimally selects $|L|$ routes that maximize the total profit.

Let $\mathfrak{R}$ denote the set of non-duplicated routes generated during the ALNS iterations. Each route $r \in \mathfrak{R}$ has an associated profit φ_r which is equal to the sum of nodes' profits served by this route. Observe that two routes r_1 and r_2 are considered duplicated if they both serve the same set of nodes. Hence, before adding a route to the set $\mathfrak{R}$, our algorithm verifies if it is duplicated otherwise it adds it to $\mathfrak{R}$. Technically, this is done by using a hashset. Variables z_r are defined for each route $r \in \mathfrak{R}$ such that z_r equal to 1 if route r is chosen, and 0 otherwise. We also define a parameter for each route $r \in \mathfrak{R}$ and each node $i \in V^*$ such that $a_{ri} = 1$ if route r serves node i, and 0 otherwise. The SPP is formulated as follows:

$$\max \quad \sum_{r\in\mathfrak{R}} \varphi_r z_r \tag{10}$$

$$\text{s.t.} \quad \sum_{r\in\mathfrak{R}} a_{ri} z_r \leq 1 \qquad \forall i \in V^* \tag{11}$$

$$\sum_{r\in\mathfrak{R}} z_r = |L| \tag{12}$$

$$z_r \in \{0,1\} \qquad \forall r \in \mathfrak{R}. \tag{13}$$

The objective function (10) maximizes the global profit. Constraints (11) imply that each node $i \in V^*$ can be served at most once. Constraint (12) imposes that the number of selected routes is equal to $|L|$. Finally, constraints (13) define the variables.

5. Computational results

The HALNS is implemented in Java. All experiments were conducted on a 64-bit version of Windows 10, with an Intel Core i7 processor 7700-HQ, 2.80 GHz of base frequency, 3.80 GHz as max turbo frequency with 8 threads and 16 GB of RAM. CPLEX 12.10 was used as MIP solver to solve the SPP and the SROP. Twenty independent runs are performed for each instance and the best obtained solutions are reported. In order to evaluate our solution approach, we compare it against the existing heuristics in the literature. We use two sets of instances. The first one includes 387 small-scale instances (Chao et al., 1996). The second set includes 333 large-scale instances proposed by Dang et al. (2013b). All instances and detailed computational results are available from `https://www.leandro-coelho.com/team-orienteering-problem/`.

5.1. Small/large-scale benchmark instances description

The small-scale instances were reported in Chao et al. (1996). They are divided into seven sets depending on the number of nodes $|V|$ (from 21 to 102). For each set of instances, the parameters are the number of vehicles $|L|$ and the time limit D_{max}. A total of 387 instances are reported in Chao et al. (1996). As in other papers dealing with the TOP, instances for which all the state-of-the-art heuristics obtain the same results are excluded from the comparison. Hence, we compare the results obtained for 157 relevant benchmark instances over the 387 instances of Chao et al. (1996) corresponding to instances from sets 4, 5, 6 and 7. For these instances, we compare our HALNS to 26 state-of-the-art heuristics described in Table 1. It is important to mention that the proposed state-of-the-art heuristics were executed on different processors versions with different base frequencies. These are also reported in Table 1.

Table 1: State-of-the-art heuristics for the TOP

Name	Description	Processor	Reference
TMH	Tabu search	DEC AlphaServer 1200/533 and DEC Alpha XP 1000	Tang and Miller-Hooks (2005)
GTP	Tabu search with penalty strategy	Intel Pentium 4, 2.80 GHz	Archetti et al. (2007)
GTF	Tabu search with feasible strategy	Intel Pentium 4, 2.80 GHz	Archetti et al. (2007)
FVNS	Fast Variable Neighborhood Search	Intel Pentium 4, 2.80 GHz	Archetti et al. (2007)
SVNS	Slow Variable Neighborhood Search	Intel Pentium 4, 2.80 GHz	Archetti et al. (2007)
SACO	Sequential Ant Colony Optimization	Intel PC, 3.00 GHz	Ke et al. (2008)
DACO	Deterministic variant of Ant Colony Optimization	Intel PC, 3.00 GHz	Ke et al. (2008)
RACO	Random variant of Ant Colony Optimization	Intel PC, 3.00 GHz	Ke et al. (2008)
SiACO	Simultaneous variant of Ant Colony Optimization	Intel PC, 3.00 GHz	Ke et al. (2008)
SkVNS	Skewed Variable Neighborhood Search	Intel Pentium 4, 2.80 GHz	Vansteenwegen et al. (2009b)
GLS	Guided Local Search	Intel Pentium 4, 2.80 GHz	Vansteenwegen et al. (2009a)
FPR	Fast variant of Path Relinking	Intel XEON, 2.50 GHz	Souffriau et al. (2010)
SPR	Slow variant of Path Relinking	Intel XEON, 2.50 GHz	Souffriau et al. (2010)
MA	Memetic Algorithm	Intel Core 2 Duo-E6750, 2.67 GHz	Bouly et al. (2010)
PSOMA	Particle Swarm Optimization-based MA	AMD Opteron, 2.60 GHz	Dang et al. (2011)
AuLNS	Augmented Large Neighborhood Search	Intel Core i7-2600, 3.40 GHz	Kim et al. (2013)
PSOiA	PSO-inspired Algorithm	AMD Opteron, 2.60 GHz	Dang et al. (2013b)
MSA	Multi-start Simulated Annealing	Intel Core 2, 2.50 GHz	Lin (2013)
UHGS	Unified Hybrid Genetic Search	Intel Xeon, 3.07 GHz	Vidal et al. (2016)
UHGS-f	Fast version of UHGS	Intel Xeon, 3.07 GHz	Vidal et al. (2016)
MS-ILS	Multistart Integrated Local Search	Intel Xeon, 3.07 GHz	Vidal et al. (2016)
MS-LS	Multistart local-improvement	Intel Xeon, 3.07 GHz	Vidal et al. (2016)
PMA	Pareto Mimic Algorithm	Intel Core i5, 3.20 GHz	Ke et al. (2016)
SHHS	Similarity Hybrid Harmony Search	Intel Core i7-2670QM, 2.20 GHz	Tsakirakis et al. (2019)
SHHS2	Second version of SHHS	Intel Core i7-2670QM, 2.20 GHz	Tsakirakis et al. (2019)
LNS	Large Neighborhood Search	Intel Core i7-6700U, 4.00 GHz	Orlis et al. (2020)
HALNS	Hybrid Adaptive Large Neighborhood Search	Intel Core i7-7700HQ, 2.80 GHz	This paper

The results obtained by Chao et al. (1996) are not considered in this comparison as they use a different rounding precision to obtain the travel times and are outperformed by the other solution approaches (Bianchessi et al., 2018).

In their work, Dang et al. (2013b) estimated that it will be more difficult to improve the BKS for the small-scale instances of Chao et al. (1996). Hence, they introduced a new set of instances with a larger number of nodes. Dang et al. (2013b) generated 333 instances on the basis of the ones of the OP previously generated by Fischetti et al. (1998) with the transformation of Chao et al. (1996). Two classes of large-scale instances were generated by Dang et al. (2013b). The first class is derived from instances of the CVRP (Christofides et al., 1979; Reinelt, 1991) in which customers demands were transformed into profits and different values of D_{max} were considered. The second class is derived from instances of the Traveling Salesman Problem (Reinelt, 1991) in which customers' profits were generated on the basis of three different methods. For these large instances, the number of nodes $|V|$ varies from 102 to 401 and the number of vehicles $|L|$ varies between 2 and 4. We refer the reader to Dang et al. (2013b) for more details. The methods of Dang et al. (2013b) and Ke et al. (2016) are used to compare to our solutions.

5.2. Parameter settings for HALNS

In order to set the HALNS parameters, we have run several preliminary tests. We noticed that a high value for the number of nodes to be removed from the solution (β) and a high number of iterations exceeding 100,000 have a negative impact on the computational time. Furthermore, the algorithm tends to quickly obtain very good solutions when the SA initial temperature T_0 is set to a value lower than 100. As for the stopping criterion for ALNS, a maximum of 3,000 and 5,000 iterations without any improvement is considered respectively for small/large-scale instances. After an experimental phase, the retained parameters are presented in Table 2. It is important to mention that $|V_+^l|$ denotes the number of nodes served by vehicle l.

Observe that the maximum value that could be taken by α^l is set to $15\%|V_+^l|$ for the SROP is based on our knowledge of the OP which depends, among others, on the number of nodes. To solve the SPP, we set the time limit to 60 seconds and provided the solver with the best solution (s_{best}) returned by the ALNS as an initial solution. As for the

Table 2: Parameter tuning of our HALNS heuristic

Parameter	Description	Value
N_{seg}	Maximum number of run segments	100/500 for small/large-scale instances
$iteration_{max}$	Maximum number of iterations per run segment	1,000
$iteration_{best_{max}}$	Maximum number of iterations without improvement	3,000/5,000 for small/large-scale instances
β	Random number of nodes to remove from the current solution	$\beta \in [1, 0.25\|V_+\|]$
T_0	SA initial temperature	95
T_{min}	SA minimum temperature	0.0001
c	Cooling rate for the SA	0.9999
ρ_1	Operator score increment case 1	20
ρ_2	Operator score increment case 2	5
ρ_3	Operator score increment case 3	1
λ	Reaction factor to adjust node selection strategies and operators weights	0.85
α^l	Random size of the sequence to remove	$[2, \ldots, 15\%\|V_+^l\|]$

SROP, we set the same time limit. The latter is solved by our branch-and-cut procedure described in Section 4.4. It is important to mention that HALNS uses one thread except when solving the SROP and the SPP which are solved by CPLEX in a multi-threads mode.

5.3. Results for the small-scale benchmark instances

In Tables 3–6, we report the values of the solutions obtained for the small-scale instances by the different state-of-the-art algorithms as well as those of our proposed algorithm under the column HALNS for sets 4, 5, 6 and 7, respectively. The best obtained solution for each instance is reported under the column "Best".

As depicted in Table 3, HALNS is able to obtain all the BKS for the 54 instances of set 4. Four heuristics were able to obtain the same results: AuLNS, PSOiA, PMA and LNS proposed respectively by Kim et al. (2013), Dang et al. (2013b), Ke et al. (2016) and Orlis et al. (2020). Although 5 methods reported the same results, our HALNS beats them with an average computational time of 32.24 seconds versus 77.30, 218.58, 109.30 and 218.02 seconds respectively required by AuLNS, PSOiA, PMA and LNS. Observe from Table 3 that the BKS for instance "p4.4.n" reported by TMH proposed by Tang and Miller-Hooks (2005) is proven infeasible according to the optimal solutions reported by the branch-cut-and-price algorithms of Pessoa et al. (2019) and Orlis et al. (2020). Tang and Miller-Hooks (2005) report a solution of 977 for instance "p4.4.n" whereas our HALNS and eight other heuristics report a solution value of 976, besides the exact methods of Pessoa et al. (2019) and Orlis et al. (2020).

For set 5, our HALNS identified the BKS for all 45 instances. This was also the case for the solution approaches proposed in Dang et al. (2013b), Kim et al. (2013), Vidal et al. (2016) and Ke et al. (2016) who obtained these solutions respectively within an average computational time of 49.50, 22.10, 138.02, 52.86, 89.34 and 22.90 seconds vs only 11.63 seconds for our HALNS. Observe however that the MSA proposed by Lin (2013) obtained 44 BKS in a very short average computational time of 6.60 seconds.

For set 6, 16 solutions approaches in addition to our HALNS were able to obtain all the BKS for all the instances (15 on total). In terms of computational time, the fastest approach was MSA developed by Lin (2013) with an average computational time of 1.40 seconds which is much smaller than the average time required by all the other solution approaches.

For set 7, in addition to AuLNS, PSOiA, UHGS, UHGS-f, MS-ILS, PMA and LNS proposed respectively by Kim et al. (2013), Dang et al. (2013b), Vidal et al. (2016), Ke et al. (2016) and Orlis et al. (2020), our HALNS identified all 43 BKS. In terms of computational time, only 30.89 seconds in average were required by our algorithm to obtain the BKS versus 66.80, 97.47, 228.01, 81.03, 201.87, 54.60 and 152.95 seconds required respectively by AuLNS, PSOiA, UHGS, UHGS-f, MS-ILS, PMA and LNS which proves that our heuristic is very efficient and fast.

Tables 7–9 summarize the results and the computational time for each studied solution approach for the small-scale benchmark instances. In Table 7, we provide the number of times each solution approach reached the BKS for each set of the benchmark instances. Table 8 provides the average gap over these instances to the BKS computed as follows:

$$Gap(\%) = \sum_{ins=1}^{157} \left(\frac{\frac{BKS_{ins} - S_{ins}}{S_{ins}}}{157} \right)$$

where BKS_{ins} is the BKS value for instance ins and S_{ins} denotes the solution value obtained by the corresponding solution approach. Table 9 reports for each heuristic the average computational time in seconds over all the instances.

5.4. Results for the large-scale benchmark instances

Considering the fact that only Dang et al. (2013b) and Ke et al. (2016) tested their solution approaches, respectively the PSOiA and the PMA, on the large-scale instances, we compare in the following their best results and the average computational time with our HALNS. According to Dang et al. (2013b), the average relative percentage error (ARPE) between the best value ($Best$) and the mean solution value obtained by the PSOiA, defined by $\frac{Best-mean}{mean}(\%)$, is zero in 251 instances. Hence, only the results for 82 instances are reported. Table 10, respectively Table 11, shows the best solution values and the CPU times reported by Dang et al. (2013b), Ke et al. (2016) and those of our HALNS for the large scale instances with up to 230, respectively, 401 nodes. Given that our HALNS uses 20 run replications for each instance whereas PMA and PSOiA only use 10, we report the mean solution value obtained by each heuristic over these runs. Tables 10 and 11 also report for each instance the BKS and the best CPU in seconds required to identify the BKS.

The results of Table 10 show that for the 40 instances with up to 230 nodes, an average computational time of 300.38 seconds was required for HALNS to obtain the 40 BKS versus an average computational time of 3,234.16 and 627.24 seconds respectively for PSOiA and PMA which obtained 36 and 40 of the BKS. Furthermore, HALNS was faster than PMA for 36 instances. As for PSOiA, it is dominated, in terms of solution quality and computational time, by both HALNS and PMA for all the instances with up to 230 nodes.

For the remaining 42 instances with up to 401 nodes, the results reported in Table 11 show that HALNS identifies all the BKS with an average computational time of 1,234.35 seconds and reports a new solution for the instance "rd400_gen2_m3" (401 nodes and 3 vehicles) with a value of 12,646. PSOiA and PMA reported respectively 35 and 41 of the

Table 3: Results for set 4 of the small-scale benchmark

Ins	BKS	TMH	GTP	GTF	FVNS	SVNS	SACO	DACO	RACO	SiACO	SkVNS	GLS	FPR	SPR	MA	PSOMA	AuLNS	PSOiA	MSA	UHGS	UHGS-f	MS-ILS	MS-LS	PMA	SHHS	SHHS2	LNS	HALNS
p4.2.a	206	202	206	206	206	206	206	206	206	206	202	206	206	206	206	206	206	206	206	206	206	206	206	206	206	206	206	206
p4.2.b	341	341	341	341	341	341	341	341	341	341	341	303	341	341	341	341	341	341	341	341	341	341	341	341	341	341	341	341
p4.2.c	452	438	452	452	452	452	452	452	452	452	452	447	452	452	452	452	452	452	452	452	452	452	452	452	452	452	452	452
p4.2.d	531	517	530	531	531	531	531	531	530	531	528	526	531	531	531	531	531	531	531	531	531	531	531	531	528	531	531	531
p4.2.e	618	593	618	613	618	618	618	600	600	613	593	602	612	618	618	618	618	618	618	618	618	618	618	618	611	613	618	618
p4.2.f	687	666	687	676	684	687	687	672	672	672	675	651	687	687	687	687	687	687	687	687	687	687	687	687	673	684	687	687
p4.2.g	757	749	751	756	750	753	757	756	756	756	750	734	757	757	757	757	757	757	757	757	757	757	752	757	753	757	757	757
p4.2.h	835	827	795	820	827	835	827	819	819	820	819	797	835	835	835	835	835	835	835	835	835	835	825	835	816	827	835	835
p4.2.i	918	915	882	899	916	918	918	900	918	918	916	826	918	918	918	918	918	918	918	918	918	918	918	918	896	912	918	918
p4.2.j	965	914	946	962	962	962	965	962	962	962	962	939	962	965	965	965	965	965	962	965	964	965	964	965	936	957	965	965
p4.2.k	1022	963	1013	1013	1019	1022	1022	1016	1016	1016	1007	994	1013	1022	1022	1022	1022	1022	1022	1022	1022	1022	1022	1022	983	1010	1022	1022
p4.2.l	1074	1022	1061	1058	1073	1074	1071	1070	1071	1069	1051	1051	1064	1074	1071	1071	1074	1074	1073	1074	1074	1074	1073	1074	1048	1074	1074	1074
p4.2.m	1132	1089	1106	1098	1132	1132	1130	1115	1119	1113	1051	1051	1130	1132	1132	1132	1132	1132	1132	1132	1132	1132	1132	1132	1078	1125	1132	1132
p4.2.n	1174	1150	1169	1171	1159	1171	1168	1149	1158	1169	1124	1117	1161	1173	1174	1174	1174	1174	1174	1174	1174	1174	1172	1174	1152	1168	1174	1174
p4.2.o	1218	1175	1180	1192	1216	1218	1215	1209	1198	1210	1195	1191	1206	1218	1218	1218	1218	1218	1217	1218	1218	1218	1213	1218	1171	1216	1218	1218
p4.2.p	1242	1208	1226	1239	1239	1241	1242	1229	1233	1239	1237	1214	1240	1242	1242	1241	1242	1242	1241	1242	1241	1242	1239	1242	1211	1242	1242	1242
p4.2.q	1268	1255	1252	1255	1265	1263	1263	1253	1252	1260	1239	1248	1257	1263	1267	1267	1268	1268	1259	1267	1267	1267	1262	1268	1245	1259	1268	1268
p4.2.r	1292	1277	1281	1283	1283	1286	1288	1278	1278	1279	1279	1267	1278	1286	1292	1292	1292	1292	1290	1292	1285	1287	1285	1292	1269	1282	1292	1292
p4.2.s	1304	1294	1296	1299	1300	1301	1304	1304	1303	1304	1295	1286	1293	1296	1304	1304	1304	1304	1300	1302	1302	1304	1301	1304	1284	1294	1304	1304
p4.2.t	1306	1306	1306	1306	1306	1306	1306	1306	1306	1306	1305	1294	1299	1306	1306	1306	1306	1306	1306	1306	1306	1306	1306	1306	1302	1306	1306	1306
p4.3.c	193	192	193	193	193	193	193	193	193	193	193	193	193	193	193	193	193	193	193	193	193	193	193	193	193	193	193	193
p4.3.d	335	333	335	335	335	335	335	333	333	335	331	335	333	335	335	335	335	335	335	335	335	335	335	335	335	335	335	335
p4.3.e	468	465	468	468	468	468	468	468	468	468	460	444	468	468	468	468	468	468	468	468	468	468	468	468	468	468	468	468
p4.3.f	579	579	579	579	579	579	579	579	579	579	556	564	579	579	579	579	579	579	579	579	579	579	579	579	579	579	579	579
p4.3.g	653	646	651	652	653	653	653	652	653	652	651	644	653	653	653	653	653	653	653	653	653	653	653	653	651	653	653	653
p4.3.h	729	709	722	727	724	729	720	713	713	713	718	706	725	729	728	729	729	729	729	729	729	729	728	729	715	729	729	729
p4.3.i	809	785	806	806	806	807	796	793	793	786	807	806	797	809	809	809	809	809	809	809	809	809	809	809	796	809	809	809
p4.3.j	861	860	858	858	861	861	861	857	855	858	854	826	858	861	861	861	861	861	860	861	861	861	861	861	854	858	861	861
p4.3.k	919	906	919	918	919	919	918	913	910	910	902	864	918	918	919	919	919	919	919	919	919	919	919	919	908	919	919	919
p4.3.l	979	951	976	973	975	978	979	958	976	966	969	960	968	979	979	979	979	979	978	979	979	979	979	979	954	973	979	979
p4.3.m	1063	1005	1034	1049	1056	1063	1053	1039	1028	1046	1047	1030	1043	1063	1063	1063	1063	1063	1063	1063	1063	1063	1051	1063	1028	1063	1063	1063
p4.3.n	1121	1119	1108	1115	1111	1121	1121	1109	1112	1103	1106	1113	1108	1120	1121	1121	1121	1121	1121	1121	1121	1121	1121	1121	1093	1116	1121	1121
p4.3.o	1172	1151	1156	1157	1172	1170	1170	1163	1167	1165	1136	1121	1165	1170	1172	1172	1172	1172	1170	1172	1170	1172	1170	1172	1149	1168	1172	1172
p4.3.p	1222	1218	1207	1221	1208	1222	1221	1202	1207	1207	1200	1190	1209	1220	1222	1222	1222	1222	1222	1222	1222	1222	1208	1222	1213	1222	1222	1222
p4.3.q	1253	1249	1237	1241	1250	1251	1252	1239	1239	1238	1236	1210	1246	1253	1253	1253	1253	1253	1251	1253	1253	1253	1253	1253	1226	1251	1253	1253
p4.3.r	1273	1265	1224	1269	1272	1272	1267	1263	1263	1263	1250	1239	1257	1272	1273	1273	1273	1273	1265	1273	1272	1272	1271	1273	1247	1269	1273	1273
p4.3.s	1295	1282	1250	1294	1289	1293	1293	1291	1289	1291	1280	1279	1276	1287	1295	1295	1295	1295	1293	1295	1295	1295	1290	1295	1265	1285	1295	1295
p4.3.t	1305	1288	1303	1304	1298	1304	1305	1304	1303	1304	1299	1290	1294	1299	1305	1304	1305	1305	1299	1305	1305	1304	1301	1305	1278	1302	1305	1305
p4.4.e	183	182	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183	183
p4.4.f	324	315	324	324	324	324	324	324	324	324	319	312	324	324	324	324	324	324	324	324	324	324	324	324	324	324	324	324
p4.4.g	461	453	461	461	461	461	461	461	461	460	461	461	461	461	461	461	461	461	461	461	461	461	461	461	461	461	461	461
p4.4.h	571	554	571	571	571	571	571	556	556	556	553	565	571	571	571	571	571	571	571	571	571	571	571	571	556	571	571	571
p4.4.i	657	627	655	657	657	657	657	653	652	653	657	657	653	657	657	657	657	657	657	657	657	657	657	657	653	657	657	657
p4.4.j	732	732	731	731	732	732	732	731	711	731	723	691	732	732	732	732	732	732	732	732	732	732	732	732	731	732	732	732
p4.4.k	821	819	821	816	821	821	821	820	818	818	821	815	820	821	821	821	821	821	821	821	821	821	821	821	819	820	821	821
p4.4.l	880	875	878	878	879	880	880	877	875	875	876	852	875	879	880	880	880	880	880	880	880	880	880	880	859	878	880	880
p4.4.m	919	910	916	918	916	919	918	911	906	911	903	910	914	919	916	919	919	919	919	919	919	919	919	919	896	916	919	919
p4.4.n	976	~~977~~	972	976	968	968	961	956	956	956	948	942	953	969	969	969	976	976	975	976	976	976	967	976	948	971	976	976
p4.4.o	1061	1014	1057	1057	1051	1061	1036	1030	1021	1029	1030	937	1033	1057	1061	1061	1061	1061	1061	1061	1061	1061	1057	1061	1035	1061	1061	1061
p4.4.p	1124	1056	1120	1120	1120	1120	1111	1108	1088	1110	1120	1091	1098	1122	1124	1124	1124	1124	1124	1124	1124	1124	1124	1124	1112	1120	1124	1124
p4.4.q	1161	1124	1148	1157	1160	1161	1145	1150	1137	1148	1149	1106	1139	1160	1161	1161	1161	1161	1161	1161	1161	1161	1159	1161	1129	1158	1161	1161
p4.4.r	1216	1165	1203	1211	1207	1203	1200	1195	1195	1194	1193	1148	1196	1213	1216	1216	1216	1216	1216	1216	1216	1216	1207	1216	1170	1216	1216	1216
p4.4.s	1260	1243	1245	1256	1259	1255	1249	1256	1249	1252	1213	1242	1231	1250	1260	1259	1260	1260	1256	1260	1260	1260	1259	1260	1223	1252	1260	1260
p4.4.t	1285	1255	1279	1285	1282	1279	1281	1281	1283	1281	1281	1250	1256	1280	1285	1285	1285	1285	1285	1285	1285	1285	1280	1285	1264	1281	1285	1285

Table 4: Results for set 5 of the small-scale benchmark

Ins	BKS	TMH	GTP	GTF	FVNS	SVNS	SACO	DACO	RACO	SiACO	SkVNS	GLS	FPR	SPR	MA	PSOMA	AuLNS	PSOiA	MSA	UHGS	UHGS-f	MS-ILS	MS-LS	PMA	SHHS	SHHS2	LNS	HALNS
p5.2.h	410	410	410	410	410	410	410	410	410	410	395	385	410	410	410	410	410	410	410	410	410	410	410	410	410	410	410	410
p5.2.j	580	560	580	580	580	580	580	580	580	580	580	580	580	580	580	580	580	580	580	580	580	580	580	580	580	580	580	580
p5.2.k	670	670	670	670	670	670	670	670	670	670	670	665	670	670	670	670	670	670	670	670	670	670	670	670	670	670	670	670
p5.2.l	800	770	800	800	800	800	800	800	800	800	770	760	800	800	800	800	800	800	800	800	800	800	800	800	800	800	800	800
p5.2.m	860	860	860	860	860	860	860	860	860	860	860	830	860	860	860	860	860	860	860	860	860	860	860	860	860	860	860	860
p5.2.n	925	920	925	925	925	925	925	920	920	925	920	920	925	925	925	925	925	925	925	925	925	925	925	925	925	925	925	925
p5.2.o	1020	975	1020	1020	1020	1020	1020	1020	1010	1010	1020	1010	1020	1020	1020	1020	1020	1020	1020	1020	1020	1020	1020	1020	1020	1020	1020	1020
p5.2.p	1150	1090	1130	1150	1150	1150	1150	1150	1150	1150	1150	1030	1150	1150	1150	1150	1150	1150	1150	1150	1150	1150	1150	1150	1150	1150	1150	1150
p5.2.q	1195	1185	1195	1195	1195	1195	1195	1195	1195	1195	1195	1145	1195	1195	1195	1195	1195	1195	1195	1195	1195	1195	1195	1195	1195	1195	1195	1195
p5.2.r	1260	1260	1260	1260	1260	1260	1260	1260	1260	1260	1260	1225	1260	1260	1260	1260	1260	1260	1260	1260	1260	1260	1260	1260	1260	1260	1260	1260
p5.2.s	1340	1310	1330	1340	1340	1340	1340	1330	1330	1330	1325	1325	1340	1340	1330	1340	1340	1340	1340	1340	1340	1340	1340	1340	1325	1340	1340	1340
p5.2.t	1400	1380	1380	1400	1400	1400	1400	1400	1400	1400	1380	1360	1400	1400	1400	1400	1400	1400	1400	1400	1400	1400	1400	1400	1380	1400	1400	1400
p5.2.u	1460	1445	1440	1460	1460	1460	1460	1460	1460	1460	1450	1460	1460	1460	1460	1460	1460	1460	1460	1460	1460	1460	1460	1460	1460	1460	1460	1460
p5.2.v	1505	1500	1490	1505	1500	1505	1505	1495	1500	1495	1500	1500	1505	1505	1505	1505	1505	1505	1505	1505	1505	1505	1505	1505	1505	1505	1505	1505
p5.2.w	1565	1560	1555	1565	1560	1560	1560	1555	1555	1555	1560	1560	1560	1560	1560	1560	1565	1565	1565	1565	1565	1565	1560	1565	1560	1560	1560	1565
p5.2.x	1610	1610	1595	1610	1590	1610	1610	1610	1610	1610	1600	1610	1610	1610	1610	1610	1610	1610	1610	1610	1610	1610	1610	1610	1610	1610	1610	1610
p5.2.y	1645	1630	1635	1635	1635	1635	1645	1645	1645	1645	1630	1630	1645	1645	1645	1645	1645	1645	1645	1645	1645	1645	1640	1645	1645	1645	1645	1645
p5.2.z	1680	1665	1670	1680	1670	1670	1680	1680	1680	1680	1665	1680	1670	1680	1680	1680	1680	1680	1680	1680	1680	1680	1680	1680	1680	1680	1680	1680
p5.3.k	495	495	495	495	495	495	495	495	495	495	495	470	495	495	495	495	495	495	495	495	495	495	495	495	495	495	495	495
p5.3.l	595	575	595	595	595	595	595	595	595	595	595	545	595	595	595	595	595	595	595	595	595	595	595	595	595	595	595	595
p5.3.n	755	755	755	755	755	755	755	755	755	755	755	720	755	755	755	755	755	755	755	755	755	755	755	755	755	755	755	755
p5.3.o	870	835	870	870	870	870	870	870	870	870	870	870	870	870	870	870	870	870	870	870	870	870	870	870	870	870	870	870
p5.3.q	1070	1065	1070	1070	1070	1070	1070	1065	1065	1065	1065	1045	1070	1070	1070	1070	1070	1070	1070	1070	1070	1070	1070	1070	1070	1070	1070	1070
p5.3.r	1125	1115	1110	1125	1125	1125	1125	1120	1125	1125	1125	1090	1125	1125	1125	1125	1125	1125	1125	1125	1125	1125	1125	1125	1125	1125	1125	1125
p5.3.s	1190	1175	1185	1190	1190	1190	1190	1190	1190	1185	1185	1145	1185	1190	1190	1190	1190	1190	1190	1190	1190	1190	1190	1190	1190	1190	1190	1190
p5.3.t	1260	1240	1250	1260	1260	1260	1260	1250	1255	1260	1260	1240	1260	1260	1260	1260	1260	1260	1260	1260	1260	1260	1260	1345	1260	1260	1260	1260
p5.3.u	1345	1330	1340	1345	1345	1345	1345	1330	1335	1335	1345	1305	1335	1345	1345	1345	1345	1345	1345	1345	1345	1345	1345	1345	1345	1345	1345	1345
p5.3.v	1425	1410	1420	1425	1425	1425	1425	1425	1425	1420	1425	1425	1420	1425	1425	1425	1425	1425	1425	1425	1425	1425	1425	1425	1425	1425	1425	1425
p5.3.w	1485	1465	1485	1485	1485	1485	1485	1465	1465	1465	1475	1460	1465	1485	1485	1485	1485	1485	1485	1485	1485	1485	1480	1485	1470	1480	1485	1485
p5.3.x	1555	1530	1555	1555	1555	1555	1540	1535	1540	1540	1535	1520	1540	1550	1555	1555	1555	1555	1555	1555	1555	1555	1535	1555	1545	1550	1555	1555
p5.3.y	1595	1580	1590	1595	1595	1595	1590	1590	1590	1590	1580	1590	1590	1590	1590	1595	1595	1595	1590	1595	1595	1595	1590	1595	1590	1595	1595	1595
p5.3.z	1635	1635	1625	1635	1635	1635	1635	1635	1635	1635	1635	1635	1635	1635	1635	1635	1635	1635	1635	1635	1635	1635	1635	1635	1620	1635	1635	1635
p5.4.m	555	555	555	555	555	555	555	555	555	555	550	550	555	555	555	555	555	555	555	555	555	555	555	555	555	555	555	555
p5.4.o	690	680	690	690	690	690	690	690	690	690	690	680	690	690	690	690	690	690	690	690	690	690	690	690	690	690	690	690
p5.4.p	765	760	765	765	765	765	765	760	760	760	760	760	760	760	760	765	765	765	765	765	765	765	765	765	765	765	765	765
p5.4.q	860	860	860	860	860	860	860	860	860	860	835	830	860	860	860	860	860	860	860	860	860	860	860	860	860	860	860	860
p5.4.r	960	960	960	960	960	960	960	960	960	960	960	890	960	960	960	960	960	960	960	960	960	960	960	960	960	960	960	960
p5.4.s	1030	1000	1025	1030	1030	1030	1030	1030	1030	1030	1020	1020	1005	1025	1030	1030	1030	1030	1030	1030	1030	1030	1030	1030	1025	1030	1030	1030
p5.4.t	1160	1100	1160	1160	1160	1160	1160	1160	1160	1160	1160	1160	1160	1160	1160	1160	1160	1160	1160	1160	1160	1160	1160	1160	1160	1160	1160	1160
p5.4.u	1300	1275	1300	1300	1300	1300	1300	1300	1300	1300	1300	1300	1300	1300	1300	1300	1300	1300	1300	1300	1300	1300	1300	1300	1300	1300	1300	1300
p5.4.v	1320	1310	1320	1320	1320	1320	1320	1320	1320	1320	1320	1245	1320	1320	1320	1320	1320	1320	1320	1320	1320	1320	1320	1320	1320	1320	1320	1320
p5.4.w	1390	1380	1375	1390	1390	1390	1390	1380	1390	1380	1380	1330	1380	1390	1380	1385	1390	1390	1390	1390	1390	1390	1390	1390	1380	1385	1390	1390
p5.4.x	1450	1410	1440	1450	1450	1450	1450	1450	1450	1450	1440	1410	1430	1450	1450	1450	1450	1450	1450	1450	1450	1450	1450	1450	1435	1445	1450	1450
p5.4.y	1520	1520	1520	1520	1520	1520	1520	1510	1510	1500	1500	1485	1520	1520	1520	1520	1520	1520	1520	1520	1520	1520	1520	1520	1485	1520	1520	1520
p5.4.z	1620	1575	1620	1620	1620	1620	1620	1620	1575	1580	1600	1590	1620	1620	1620	1620	1620	1620	1620	1620	1620	1620	1620	1620	1620	1620	1620	1620

Table 5: Results for set 6 of the small-scale benchmark

Ins	BKS	TMH	GTP	GTF	FVNS	SVNS	SACO	DACO	RACO	SiACO	SkVNS	GLS	FPR	SPR	MA	PSOMA	AuLNS	PSOiA	MSA	UHGS	UHGS-f	MS-ILS	MS-LS	PMA	SHHS	SHHS2	LNS	HALNS
p6.2.d	192	192	192	192	192	192	192	192	192	192	192	180	192	192	192	192	192	192	192	192	192	192	192	192	192	192	192	192
p6.2.j	948	936	948	948	948	948	948	948	948	948	948	948	942	948	948	948	948	948	948	948	948	948	948	948	948	948	948	948
p6.2.l	1116	1116	1098	1110	1116	1116	1116	1110	1116	1116	1116	1104	1110	1116	1116	1116	1116	1116	1116	1116	1116	1116	1116	1116	1116	1116	1116	1116
p6.2.m	1188	1188	1164	1188	1188	1188	1188	1188	1188	1188	1188	1164	1188	1188	1188	1188	1188	1188	1188	1188	1188	1188	1188	1188	1188	1188	1188	1188
p6.2.n	1260	1260	1242	1260	1260	1260	1260	1260	1254	1260	1248	1254	1260	1260	1260	1260	1260	1260	1260	1260	1260	1260	1254	1260	1260	1260	1260	1260
p6.3.g	282	282	282	282	282	282	282	282	282	282	276	264	282	282	282	282	282	282	282	282	282	282	282	282	282	282	282	282
p6.3.h	444	444	444	444	444	444	444	444	438	438	444	444	444	444	444	444	444	444	444	444	444	444	444	444	444	444	444	444
p6.3.i	642	612	642	642	642	642	642	642	642	642	642	642	642	642	642	642	642	642	642	642	642	642	642	642	642	642	642	642
p6.3.k	894	876	894	894	894	894	894	888	888	894	894	882	894	894	894	894	894	894	894	894	894	894	894	894	894	894	894	894
p6.3.l	1002	990	1002	1002	1002	1002	1002	1002	1002	1002	996	990	1002	1002	1002	1002	1002	1002	1002	1002	1002	1002	1002	1002	1002	1002	1002	1002
p6.3.m	1080	1080	1080	1080	1080	1080	1080	1074	1080	1080	1080	1068	1080	1080	1080	1080	1080	1080	1080	1080	1080	1080	1080	1080	1080	1080	1080	1080
p6.3.n	1170	1152	1170	1170	1170	1170	1170	1164	1164	1164	1152	1140	1164	1170	1170	1170	1170	1170	1170	1170	1170	1170	1170	1170	1170	1170	1170	1170
p6.4.j	366	366	366	366	366	366	366	366	366	366	366	360	366	366	366	366	366	366	366	366	366	366	366	366	366	366	366	366
p6.4.k	528	522	528	528	528	528	528	528	528	528	528	528	528	528	528	528	528	528	528	528	528	528	528	528	528	528	528	528
p6.4.l	696	696	696	696	696	696	696	696	696	696	678	678	696	696	696	696	696	696	696	696	696	696	696	696	696	696	696	696

Table 6: Results for set 7 of the small-scale benchmark

Ins	BKS	TMH	GTP	GTF	FVNS	SVNS	SACO	DACO	RACO	SiACO	SkVNS	GLS	FPR	SPR	MA	PSOMA	AuLNS	PSOiA	MSA	UHGS	UHGS-f	MS-ILS	MS-LS	PMA	SHHS	SHHS2	LNS	HALNS
p7.2.d	190	190	190	190	190	190	190	190	190	190	182	190	190	190	190	190	190	190	190	190	190	190	190	190	190	190	190	190
p7.2.e	290	290	290	290	289	290	290	290	290	290	289	279	290	290	290	290	290	290	290	290	290	290	290	290	290	290	290	290
p7.2.f	387	382	387	387	387	387	387	387	387	387	387	340	387	387	387	387	387	387	387	387	387	387	387	387	384	387	387	387
p7.2.g	459	459	456	459	459	459	459	459	459	459	457	440	459	459	459	459	459	459	459	459	459	459	459	459	453	459	459	459
p7.2.h	521	521	520	520	521	521	521	521	521	521	521	517	521	521	521	521	521	521	521	521	521	521	521	521	520	521	521	521
p7.2.i	580	578	579	579	575	579	580	579	579	579	579	568	578	580	580	580	580	580	579	580	580	580	580	580	572	579	580	580
p7.2.j	646	638	643	644	643	644	646	646	646	646	632	633	646	646	646	646	646	646	646	646	646	646	646	646	637	641	646	646
p7.2.k	705	702	702	705	704	705	705	704	704	704	700	691	702	705	705	705	705	705	705	705	705	705	705	705	678	700	705	705
p7.2.l	767	767	758	767	759	767	767	767	767	767	758	748	759	767	767	767	767	767	767	767	767	767	767	767	742	761	767	767
p7.2.m	827	817	827	824	824	827	827	827	827	827	827	798	816	827	827	827	827	827	827	827	827	827	827	827	794	827	827	827
p7.2.n	888	864	884	888	883	888	888	878	878	878	866	861	888	888	888	888	888	888	888	888	888	888	884	888	859	878	888	888
p7.2.o	945	914	933	945	945	945	945	945	940	941	928	897	932	945	945	945	945	945	945	945	945	945	945	945	925	941	945	945
p7.2.p	1002	987	1000	1002	1002	1002	1002	991	993	993	955	954	993	1002	1002	1002	1002	1002	1002	1002	1002	1002	1002	1002	972	997	1002	1002
p7.2.q	1044	1017	1041	1043	1038	1044	1043	1042	1043	1043	1029	1031	1043	1044	1044	1044	1044	1044	1043	1044	1044	1044	1044	1044	1005	1043	1044	1044
p7.2.r	1094	1067	1091	1088	1094	1094	1094	1093	1088	1094	1069	1075	1076	1094	1094	1094	1094	1094	1093	1094	1094	1094	1085	1094	1052	1080	1094	1094
p7.2.s	1136	1116	1123	1128	1136	1136	1136	1136	1134	1131	1118	1102	1125	1136	1136	1136	1136	1136	1135	1136	1136	1136	1133	1136	1094	1125	1136	1136
p7.2.t	1179	1165	1172	1174	1168	1179	1179	1179	1179	1179	1154	1142	1168	1175	1179	1179	1179	1179	1172	1179	1179	1179	1170	1179	1128	1160	1179	1179
p7.3.h	425	416	425	425	425	425	425	425	425	425	425	418	425	425	425	425	425	425	425	425	425	425	425	425	425	425	425	425
p7.3.i	487	481	487	487	487	487	487	487	486	487	480	480	485	487	487	487	487	487	487	487	487	487	487	487	487	487	487	487
p7.3.j	564	563	564	564	562	564	564	564	564	564	543	539	560	564	564	564	564	564	564	564	564	564	564	564	558	564	564	564
p7.3.k	633	632	633	633	632	633	633	632	633	633	633	586	633	633	633	633	633	633	633	633	633	633	633	633	632	633	633	633
p7.3.l	684	681	683	679	681	681	684	683	684	684	681	668	684	684	684	683	684	684	684	684	684	684	684	684	681	684	684	684
p7.3.m	762	756	749	755	745	762	762	762	762	762	743	735	762	762	762	762	762	762	762	762	762	762	762	762	762	762	762	762
p7.3.n	820	789	810	811	814	820	820	819	819	820	804	789	813	820	820	820	820	820	820	820	820	820	820	820	809	819	820	820
p7.3.o	874	874	873	865	871	874	874	874	874	874	841	833	859	874	874	874	874	874	874	874	874	874	874	874	856	874	874	874
p7.3.p	929	922	917	923	926	927	929	925	926	925	918	912	925	927	929	927	929	929	927	929	929	929	925	929	915	926	929	929
p7.3.q	987	966	976	987	978	987	987	987	987	987	966	945	970	987	987	987	987	987	987	987	987	987	984	987	972	987	987	987
p7.3.r	1026	1011	1018	1022	1024	1022	1026	1024	1021	1022	1009	1015	1017	1021	1026	1026	1026	1026	1026	1026	1026	1026	1024	1026	1003	1019	1026	1026
p7.3.s	1081	1061	1081	1081	1079	1079	1081	1081	1081	1077	1070	1054	1076	1081	1081	1081	1081	1081	1081	1081	1081	1081	1074	1081	1056	1076	1081	1081
p7.3.t	1120	1098	1114	1116	1112	1115	1118	1117	1103	1117	1109	1080	1111	1118	1120	1120	1120	1120	1119	1120	1120	1120	1113	1120	1092	1114	1120	1120
p7.4.g	217	217	217	217	217	217	217	217	217	217	217	209	217	217	217	217	217	217	217	217	217	217	217	217	217	217	217	217
p7.4.h	285	285	285	285	285	285	285	285	285	285	283	285	285	285	285	285	285	285	285	285	285	285	285	285	285	285	285	285
p7.4.i	366	359	366	366	366	366	366	366	366	366	364	359	366	366	366	366	366	366	366	366	366	366	366	366	366	366	366	366
p7.4.k	520	503	520	520	518	520	520	520	520	520	518	511	518	518	520	520	520	520	520	520	520	520	520	520	518	518	520	520
p7.4.l	590	576	590	588	588	590	590	590	590	590	575	573	581	590	590	590	590	590	590	590	590	590	590	590	576	590	590	590
p7.4.m	646	643	644	646	646	646	646	644	646	646	639	638	646	646	646	646	646	646	646	646	646	646	646	646	646	646	646	646
p7.4.n	730	726	723	721	715	730	730	725	725	726	723	698	723	730	726	726	730	730	730	730	730	730	726	730	726	726	730	730
p7.4.o	781	776	772	778	770	781	781	778	781	778	778	761	780	780	781	781	781	781	781	781	781	781	777	781	776	779	781	781
p7.4.p	846	832	841	839	846	846	846	846	838	842	841	803	842	846	846	846	846	846	846	846	846	846	846	846	834	846	846	846
p7.4.q	909	905	902	898	899	906	909	909	909	909	896	899	902	907	909	909	909	909	909	909	909	909	904	909	899	909	909	909
p7.4.r	970	966	970	969	970	970	970	970	970	970	964	937	961	970	970	970	970	970	970	970	970	970	970	970	952	970	970	970
p7.4.s	1022	1019	1021	1020	1021	1022	1022	1019	1021	1019	1019	1005	1022	1022	1022	1022	1022	1022	1022	1022	1022	1022	1022	1022	1016	1022	1022	1022
p7.4.t	1077	1067	1071	1071	1077	1077	1077	1072	1077	1077	1073	1020	1066	1077	1077	1077	1077	1077	1077	1077	1077	1077	1077	1077	1063	1077	1077	1077

Table 7: Number of best solutions found for each data set of the small-scale benchmark instances

Set	#	TMH	GTP	GTF	FVNS	SVNS	SACO	DACO	RACO	SiACO	SkVNS	GLS	FPR	SPR	MA	PSOMA	AuLNS	PSOiA	MSA	UHGS	UHGS-f	MS-ILS	MS-LS	PMA	SHHS	SHHS2	LNS	HALNS
4	54	4	16	16	22	35	30	12	12	13	7	6	17	35	49	48	54	54	38	52	47	50	30	54	10	26	54	54
5	45	12	26	44	40	42	42	31	31	30	21	9	33	40	40	43	45	45	44	45	45	45	40	45	34	40	44	45
6	15	9	12	14	15	15	15	11	11	13	10	4	12	15	15	15	15	15	15	15	15	15	14	15	15	15	15	15
7	43	8	15	20	17	35	41	26	27	28	6	2	16	36	42	40	43	43	36	43	43	43	31	43	9	24	43	43
All	157	33	69	94	94	127	128	80	81	84	44	21	78	126	146	146	157	157	133	155	150	153	115	157	68	105	156	157

Table 8: Average gap to the BKS for each data set of the small-scale benchmark instances

Set	#	TMH	GTP	GTF	FVNS	SVNS	SACO	DACO	RACO	SiACO	SkVNS	GLS	FPR	SPR	MA	PSOMA	AuLNS	PSOiA	MSA	UHGS	UHGS-f	MS-ILS	MS-LS	PMA	SHHS	SHHS2	LNS	HALNS
4	54	2.11038	0.89395	0.49242	0.28600	0.11843	0.31428	0.91739	1.00710	0.78589	1.51239	3.13540	0.73594	0.11439	0.02864	0.02441	0.00000	0.00000	0.07342	0.00431	0.02243	0.01153	0.20600	0.00000	1.67417	0.26800	0.00000	0.00000
5	45	1.42264	0.34805	0.01359	0.06938	0.03402	0.03576	0.23285	0.26022	0.28773	0.62163	2.51528	0.23033	0.04674	0.06154	0.01514	0.00000	0.00000	0.00699	0.00000	0.00000	0.00000	0.05735	0.00000	0.23166	0.03751	0.00710	0.00000
6	15	0.81086	0.34337	0.03604	0.00000	0.00000	0.00000	0.15269	0.20263	0.12569	0.53035	1.85077	0.11286	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.00000	0.03190	0.00000	0.00000	0.00000	0.00000	0.00000
7	43	1.18224	0.43780	0.30066	0.40590	0.05804	0.00639	0.14690	0.18932	0.15100	1.34151	3.22574	0.54639	0.04557	0.01281	0.02124	0.00000	0.00000	0.03141	0.00000	0.00000	0.00000	0.14298	0.00000	1.56445	0.27579	0.00000	0.00000
Avg	157	1.52145	0.56060	0.25971	0.23009	0.06704	0.12076	0.43776	0.49285	0.40681	1.11712	2.86034	0.48024	0.06588	0.03166	0.01921	0.00000	0.00000	0.03651	0.00213	0.00837	0.00462	0.13016	0.00000	1.07138	0.17912	0.00203	0.00000

Table 9: Average CPU time for each data set of the small-scale benchmark instances

Set	TMH	GTP	GTF	FVNS	SVNS	SACO	DACO	RACO	SiACO	SkVNS	GLS	FPR	SPR	MA	PSOMA	AuLNS	PSOiA	MSA	UHGS	UHGS-f	MS-ILS	MS-LS	PMA	SHHS	SHHS2	LNS	HALNS
4	796.70	105.29	282.92	22.52	457.89	370.90	317.90	307.40	320.40	7.40	11.40	8.60	367.40	182.36	83.89	77.30	218.58	81.00	236.35	81.62	202.68	15.90	109.30	107.30	109.70	218.02	**32.24**
5	71.30	69.45	26.55	34.17	158.93	173.60	150.60	143.30	151.30	1.50	3.50	2.90	119.90	35.33	14.72	22.10	49.50	6.60	138.02	52.86	89.34	3.36	22.90	30.40	28.10	66.39	**11.63**
6	45.70	66.29	20.19	8.74	147.88	161.10	140.80	135.20	141.70	1.90	4.30	2.10	89.60	39.07	7.59	12.30	47.08	1.40	91.02	37.09	56.35	1.97	36.40	14.5	12.20	42.79	**9.67**
7	432.60	158.97	256.76	10.34	309.87	303.50	245.90	233.10	246.50	4.30	12.10	6.30	272.80	112.75	49.09	66.80	97.47	32.20	228.01	81.03	201.87	9.76	54.60	82.50	82.50	152.95	**30.89**
Avg_{set}	336.58	100.00	146.61	18.94	268.64	252.28	213.80	204.75	214.98	3.78	7.83	4.98	212.43	92.38	38.82	44.63	103.16	30.30	173.35	63.15	137.56	7.75	55.80	58.68	58.13	120.04	**21.10**
Avg	445.70	113.20	189.22	22.65	322.59	294.61	249.18	238.77	250.58	4.63	9.24	6.10	260.61	114.77	50.46	55.96	128.76	41.34	192.00	68.96	156.00	9.29	66.85	74.33	74.28	140.00	**23.80**

Table 10: Results for the large-scale benchmark instances with up to 230 nodes

Instance	Best Known		PSOiA			PMA			HALNS		
	Solution	CPU (s)	Best	Mean	CPU (s)	Best	Mean	CPU (s)	Best	Mean	CPU (s)
eil101b_m3	916	69.44	916	913.8	134.39	916	914.6	160.60	916	915.8	69.44
eil101c_m2	1305	153.85	1305	1304.8	452.79	1305	1304.8	250.50	1305	1304.7	153.85
eil101c_m3	1251	95.00	1251	1244.1	227.61	1251	1244.2	95.00	1251	1248.6	116.15
cmt101c_m3	1300	23.10	1300	1299.0	111.11	1300	1299.0	42.00	1300	1299.4	23.10
bier127_gen1_m2	106	207.20	106	104.8	1153.87	106	105.0	673.50	106	105.5	207.20
bier127_gen1_m3	103	209.55	103	102.4	591.89	103	102.5	470.10	103	102.6	209.55
bier127_gen2_m2	5464	301.33	5464	5446.8	1132.57	5464	5454.9	398.80	5464	5460.2	301.33
bier127_gen2_m3	5393	227.51	5393	5376.2	648.08	5393	5386.9	359.30	5393	5390.1	227.51
bier127_gen2_m4	5123	355.99	5122	5119.2	657.57	5123	5120.2	383.80	5123	5121	355.99
bier127_gen3_m2	2885	184.99	2885	2884.3	1301.27	2885	2884.7	296.70	2885	2884.5	184.99
bier127_gen3_m3	2706	69.36	2706	2703.8	711.74	2706	2705.2	509.60	2706	2705.6	69.36
bier127_gen3_m4	2402	222.40	2402	2384.6	680.79	2402	2398.6	227.70	2402	2399.4	222.40
pr136_gen1_m2	63	70.72	63	62.7	451.13	63	62.8	107.50	63	62.9	70.72
pr136_gen2_m2	3646	95.62	3641	3631.8	601.31	3646	3637.4	281.60	3646	3642.8	95.62
kroA150_gen2_m2	4335	392.30	4335	4334.4	892.98	4335	4335.0	495.90	4335	4333.2	392.30
kroA150_gen3_m3	2726	168.81	2726	2719.6	538.01	2726	2725.2	597.20	2726	2725.8	168.81
cmt151b_m3	1385	164.65	1385	1373.8	754.01	1385	1374.5	169.90	1385	1384.0	164.65
cmt151c_m2	1964	131.96	1963	1962.0	1799.64	1964	1962.0	368.50	1964	1962.2	131.96
cmt151c_m3	1916	355.61	1916	1909.1	1376.24	1916	1909.2	441.50	1916	1914.4	355.61
cmt151c_m4	1880	107.89	1880	1875.6	881.11	1880	1877.6	826.50	1880	1878.2	107.89
rat195_gen2_m2	5148	335.53	5148	5145.6	2156.98	5148	5145.9	886.60	5148	5147.6	335.53
rat195_gen3_m3	2574	276.84	2574	2571.2	721.82	2574	2569.9	369.50	2574	2570.2	276.84
cmt200b_m2	2096	183.28	2096	2088.2	4180.99	2096	2086.8	669.40	2096	2090.2	183.28
cmt200b_m3	2019	386.33	2019	2005.0	2711.66	2019	2009.4	1351.70	2019	2004.4	386.33
cmt200b_m4	1894	324.31	1894	1889.7	1515.19	1894	1891.6	974.50	1894	1891.0	324.31
cmt200c_m2	2818	368.42	2818	2810.1	7320.26	2818	2810.6	1048.30	2818	2810.0	368.42
cmt200c_m3	2766	273.01	2766	2751.2	4217.29	2766	2751.8	1200.00	2766	2751.8	273.01
cmt200c_m4	2712	395.91	2712	2700.6	3004.10	2712	2703.3	1411.40	2712	2704.4	395.91
kroA200_gen1_m4	81	114.61	81	80.4	560.29	81	80.6	503.40	81	81.0	114.61
kroB200_gen1_m2	111	295.30	111	110.4	2344.53	111	110.3	663.90	111	111.0	295.30
kroB200_gen2_m2	6185	426.70	6185	6182.2	3467.26	6185	6183.4	426.70	6185	6184.8	438.32
kroB200_gen2_m4	4944	360.44	4944	4942.2	640.66	4944	4942.4	582.40	4944	4942.5	360.44
kroB200_gen3_m2	4765	470.10	4765	4757.8	6306.62	4765	4762.4	513.90	4765	4765.0	470.10
kroB200_gen3_m3	3028	547.22	3028	3016.0	1713.88	3028	3022.4	741.50	3028	3022.9	547.22
ts225_gen2_m2	5859	686.54	5859	5858.5	2998.43	5859	5859.0	759.60	5859	5858.6	686.54
gr229_gen1_m4	223	46.60	223	220.8	11922.02	223	220.3	46.60	223	221.0	344.28
gr229_gen2_m3	11566	441.80	11566	11551.3	14197.21	11566	11557.8	1665.30	11566	11559.2	441.80
gr229_gen2_m4	11355	377.28	11355	11255.3	18799.50	11355	11328.1	2272.00	11355	11355	377.28
gr229_gen3_m3	8056	938.75	8056	8051.6	14090.06	8056	8052.2	1065.30	8056	8055.4	938.75
gr229_gen3_m4	7651	781.50	7621	7600.0	11399.71	7651	7610.5	781.50	7651	7622.9	828.68

BKS with an average computational time of 18,456.64 and 1,363.10 seconds, respectively. HALNS was however faster than PMA to identify the BKS for 28 instances.

Table 12 summarizes the results obtained for the 82 large-scale instances. It reports for each heuristic, the number of identified BKS, the average gap to the BKS (in percentage), the number of Best Mean Solutions (BMS), the number of instances for which the heuristic

Table 11: Results for the large-scale benchmark instances with up to 401 nodes

Instance	Best Known		PSOiA			PMA			HALNS		
	Solution	CPU (s)	Best	Mean	CPU (s)	Best	Mean	CPU (s)	Best	Mean	CPU (s)
gil262a_m2	4078	451.76	4078	4056.4	5907.29	4078	4066.3	2100.00	4078	4068.8	451.76
gil262a_m4	3175	133.79	3175	3174.2	271.83	3175	3175.0	222.60	3175	3175.0	133.79
gil262b_m2	8081	545.17	8081	8061.1	7473.18	8081	8074.1	1267.80	8081	8078.5	545.17
gil262b_m3	7585	463.19	7585	7574.9	7276.80	7585	7566.6	1027.20	7585	7569.0	463.19
gil262b_m4	6781	329.10	6781	6742.0	4878.64	6781	6756.7	912.60	6781	6761.3	329.10
gil262c_m2	11030	731.25	11030	11020.0	27500.87	11030	11016.5	1309.00	11030	11022.4	731.25
gil262c_m3	10757	650.87	10757	10714.6	14553.76	10757	10715.2	1375.60	10757	10713.1	650.87
gil262c_m4	10281	516.50	10281	10259.4	8472.01	10281	10267.3	1997.00	10281	10262.8	516.50
gil262_gen1_m3	101	482.60	101	100.9	1769.31	101	100.2	482.60	101	100.8	709.66
gil262_gen1_m4	78	76.98	78	77.1	155.76	78	77.0	123.50	78	77.95	76.98
gil262_gen2_m2	7498	387.00	7498	7457.8	7356.65	7498	7458.4	742.10	7498	7466.2	387.00
gil262_gen2_m3	5615	352.39	5615	5608.2	3304.55	5615	5604.9	1163.80	5615	5609.7	352.39
gil262_gen3_m2	7183	284.20	7183	7182.8	9129.30	7183	7180.0	284.20	7183	7182.5	532.11
gil262_gen3_m4	2507	227.15	2507	2499.8	276.42	2507	2500.1	308.80	2507	2501.2	227.15
pr264_gen1_m4	107	195.83	107	106.6	503.07	107	106.7	289.80	107	106.6	195.83
pr264_gen2_m2	6635	642.98	6635	6634.2	2048.20	6635	6632.0	719.00	6635	6634.4	642.98
pr264_gen2_m3	6420	527.27	6420	6410.7	938.39	6420	6417.0	859.00	6420	6418.0	527.27
pr264_gen2_m4	5584	294.70	5584	5564.5	590.79	5584	5565.1	663.20	5584	5566.2	294.70
pr264_gen3_m3	2772	922.50	2772	2770.0	1037.51	2772	2769.8	922.50	2772	2770.0	1490.82
pr299_gen1_m2	139	355.26	139	138.5	4775.93	139	138.3	573.40	139	138.8	355.26
pr299_gen1_m3	111	413.78	111	110.1	1303.73	111	109.2	506.00	111	110.2	413.78
pr299_gen1_m4	84	191.29	84	83.6	383.48	84	83.6	340.30	84	83.4	191.29
pr299_gen2_m3	6018	690.41	6018	5966.7	1446.05	6018	5979.2	909.70	6018	5978.5	690.41
pr299_gen2_m4	4457	257.75	4457	4453.0	593.41	4457	4455.2	767.70	4457	4454.8	257.75
pr299_gen3_m2	5729	692.12	5729	5728.6	11872.55	5729	5709.8	1489.50	5729	5729.0	692.12
pr299_gen3_m3	3655	644.38	3655	3611.0	2705.82	3655	3611.6	1058.20	3655	3648.2	644.38
pr299_gen3_m4	2268	271.94	2268	2258.0	455.64	2268	2261.8	402.40	2268	2262.3	271.94
lin318_gen1_m2	180	1168.30	180	170.1	20667.24	180	175.3	1168.30	180	174.5	2702.83
lin318_gen1_m3	149	203.51	149	148.6	9014.64	149	147.9	721.40	149	148.4	203.51
lin318_gen2_m2	9544	1924.60	9544	9533.8	23804.82	9544	9537.5	1924.60	9544	9539.2	1962.14
lin318_gen2_m3	7807	1413.50	7786	7782.1	9773.63	7807	7769.6	1413.50	7807	7775.8	1600.58
lin318_gen3_m2	7936	581.27	7936	7905.6	44029.00	7936	7923.3	1547.30	7936	7919.2	581.27
lin318_gen3_m4	3797	128.84	3797	3796.4	1446.26	3797	3795.5	970.70	3797	3796.0	128.84
rd400_gen2_m2	13045	3220.60	12993	12787.5	77049.22	13045	12873.0	3220.60	13045	12872.4	4035.08
rd400_gen2_m3	12646	2814.58	12645	12372.1	53707.14	12645	12543.9	2852.70	12646	12555.8	2814.58
rd400_gen2_m4	12032	3299.30	12032	11953.5	42001.58	12032	11969.7	3299.30	12032	11981.2	4070.44
rd400_gen1_m2	232	3066.60	230	227.8	56767.29	232	228.5	3066.60	232	227.9	3925.58
rd400_gen1_m3	224	2844.00	222	221.7	62476.08	224	221.3	2844.00	224	221.4	4178.00
rd400_gen1_m4	213	1985.40	213	210.6	34744.80	213	209.9	1985.40	213	210.8	3000.90
rd400_gen3_m2	12431	2418.40	12428	12274.1	96178.70	12431	12312.2	2418.40	12431	12308.0	2814.66
rd400_gen3_m3	11639	3500.00	11639	11629.5	68074.77	11639	11549.8	3500.00	11639	11609.8	3811.24
rd400_gen3_m4	10436	3500.00	10417	10383.1	48462.77	10436	10345.4	3500.00	10436	10392.6	3615.46

was the fastest to identify the BKS, and the average computational time required to solve all the instances. It also reports for each algorithm the average deviation of its best identified solution value with regard to its mean solution value ($\frac{Best-Mean}{Mean}$). This is done to highlight the stability of an algorithm performance over run replications.

The results of Table 12 clearly show that PMA and HALNS outperform PSOiA for the large-scale benchmark instances: they identify more BKS in shorter computational times. When compared to PMA, our HALNS identified all the BKS solutions, one BKS more than PMA. For the 81 instances where our heuristic and PMA reach the BKS, HALNS was faster than PMA for 63 instances. Observe that for these 63 instances, HALNS was faster by 60 to 300 seconds for 24 instances, by 300 to 600 seconds for 16 instances and by more than 600 seconds for 16 instances. It is noteworthy that the difference in computing times of our HALNS and PMA could be due to several factors such as the stopping criteria used by each method. For example, PMA fixes the maximum time limit to 3,600 seconds and the maximum number of iterations without improvement to 3,000 (Ke et al., 2016) whereas our HALNS considers a time limit of 60 seconds to solve each of the SPP and the SROP and a maximum number of iterations without improvement of 5,000. When considering the average results over run replications (20 replications for HALNs and 10 for PMA and PSOiA), HALNS obtained 55 of the best mean solution values compared to 20 and 12 BMS for PMA and PSOiA, respectively. The three heuristics show slight variations of solution values over replications, the lowest variation being obtained for HALNS (0.26%), then for PMA (0.35%) and PSOiA (0.42%).

Table 12: Summary results for the large-scale benchmark instances

Evaluation criteria	**Solution method**	**PSOiA**	**PMA**	**HALNS**
Solution quality	**# BKS**	71	81	**82**
	Average gap to the BKS (%)	0.03971	0.00010	**0.00000**
	# BMS	12	20	**55**
	Variability over runs (%)	0.41827	0.34517	**0.26049**
CPU	**# Best CPU for the BKS**	0	18	**64**
	Average CPU (s)	11,031.04	1,004.15	**783.36**

5.5. HALNS parameters analysis

This section aims to point out the features of the proposed HALNS that make it perform well for the TOP. To this end, we evaluate the impacts of varying the values of some parameters and the impacts of disabling a number of components on our algorithm's performance. The analysis are conducted on a sample of 48 instances that are randomly selected from the set of benchmark instances previously solved: 36 instances are randomly selected from the small-scale benchmark instances and 12 instances from the large-scale ones.

5.5.1. *Impact of varying the value of the reaction factor λ*

This section analyzes the impact of varying the value of the reaction factor λ (used for adjusting node selection strategies and operators weights) on the overall performance of the algorithm. Three values of λ are considered $(0.80, 0.85,$ and $0.90)$. All the other HALNS parameters are fixed at their initial values as presented in Table 2.

Table 13 reports for each value of λ, the average percentage of time each node selection strategy, and each removal and insertion operator is used, the number of identified BKS and the average computational time. The results show that varying the value of the parameter λ has an impact on the algorithm's performance in terms of solution quality and computational time. With a value of $\lambda = 0.80$, HALNS identifies 46 over the 48 BKS. Increasing the value of λ to 0.85 and 0.90 allows to identify all the 48 BKS while slightly increasing the average computing time. The results of Table 13 also show that the percentage of time a node selection strategy or a removal/insertion operator is used is slightly affected when the value of λ changes. Observe that among our four node selection strategies, the most selected one is the dynamic profit per travel time which we designed for the TOP. The lowest profit removal operator was the most selected whereas the route and sequence removal operators were the least selected. Insertion operators are almost equally used.

Table 13: Impact of varying the value of λ on the HALNS performance

Strategy/operator	**Reaction factor (λ)**	**0.80**			**0.85**			**0.90**		
	Name/Evaluation	**Selected (%)**	**BKS**	**Average CPU (s)**	**Selected (%)**	**BKS**	**Average CPU (s)**	**Selected (%)**	**BKS**	**Average CPU (s)**
Node Selection	Dynamic profit per travel time	**29.20**	46	270.69	**30.27**	48	278.40	**31.90**	48	281.72
	Highest profit	25.66			26.00			25.82		
	Random	23.09			21.97			20.38		
	Last removed first inserted	22.05			21.77			21.90		
Removal	Random	21.27			21.00			21.35		
	Lowest Profit	**32.96**			**32.08**			**32.35**		
	Largest Saving in travel time	25.10			25.97			25.22		
	Route	9.99			10.10			10.30		
	Sequence	10.68			10.86			10.78		
Insertion	First available position	19.78			19.74			19.81		
	Last available position	**20.43**			20.72			20.15		
	Random available position	19.88			18.09			18.60		
	Best overall position	20.00			**21.65**			**20.85**		
	Best position	19.91			19.80			20.34		

5.5.2. Impact of the adaptive weighting

Adaptiveness is one of the main characteristics of ALNS heuristics. In this section, we evaluate the impact of running our HALNS without updating the operators/strategies scores (line 39 of Algorithm 1). This implies that all our strategies/operators have the same probability of being selected. Table 14 reports the impact of not considering the adaptive weighting on the heuristic's performance in terms of the number of identified BKS, the average gap to the BKS and the average computational time. The obtained results show that removing adaptiveness from our heuristic deteriorates the quality of the solution obtained (the number of BKS is reduced from 42 to 35 and the average gap to BKS is 2.49%) versus a relatively small saving in average computing times.

Table 14: Impact of the adaptive weighting

Heuristic variant	**#BKS**	**Average gap (%)**	**Average CPU (s)**
HALNS	42	0.00	278.40
HALNS without adaptive/weighting	35	2.49	259.51

5.5.3. Impact of the node selection strategies

In this section, we analyze the impact of our four proposed node selection strategies on the HALNS performance. To do so, we run the algorithm four times by considering at each time only one of these node selection strategies. Table 15 reports for each node selection strategy: the number of BKS obtained when it is the only one considered within the HALNS, the average gap with respect to the BKS and the average computational time.

Table 15: Impact of the proposed node selection strategies on the solution quality

Node selection strategy	**#BKS**	**Average gap (%)**	**Average CPU (s)**
Dynamic profit per travel time	27	0.40	297.98
Highest profit	20	0.72	291.15
Random	5	7.65	250.01
Last removed first inserted	17	5.80	255.43

The results of Table 15 show that the dynamic profit per travel time strategy outperforms the other node selection strategies: it identifies the largest number of BKS (27 over

48) with an average gap of 0.40%. The random selection strategy is the worst one and was considered to diversify the search. Even if it identified five BKS, this was probably due to the good quality of our insertion operators and local search procedures that allow to improve even bad quality solutions. Regarding computational times, the dynamic profit per travel time and the highest profit strategies required relatively larger computational times, on average, when compared to the random and last removed first inserted strategies. The difference in CPU times remains however relatively small (less than 42 seconds). Finally, it is important to mention that among the identified BKS, none was for the large-scale benchmark instances. So, it is the combination of our four node selection strategies that improves the overall performance of our algorithm. Observe that this result was also established in Section 5.5.1 through the percentage of utilization of the node selection strategies reported in Table 13 (all the node selection strategies were used with a percentage larger than 20%).

5.5.4. Impact of using SROP and SPP

This section aims to highlight the relevance of incorporating the SROP and the SPP components within the HALNS framework. Table 16 reports the number of BKS and the average computational time obtained with three variants of the HALNS: a variant where the SROP is not considered, a variant where the SPP is not considered, and the variant where both SROP and SPP are considered (i.e., our proposed algorithm).

Table 16: Impact of considering the SROP and SPP components on the HALNS performace

HALNS	**Without SROP**	**Without SPP**	**With SROP + SPP**
#BKS	43	39	**48**
Average gap (%)	0.40	0.67	0.00
Average CPU (s)	264.25	204.59	278.40

The results of Table 16 show that running the HALNS algorithm without either the SROP or the SPP components deteriorates the quality of the solutions obtained. Disabling the SPP component has more impact and reduces the number of BKS from 48 to 39. A variant without the SROP component identifies 43 BKS compared to 48 for the variant with both SROP and SPP but without a notable decrease in CPU times: 264.25 seconds on average compared to 278.40 seconds.

In summary, our analysis show that the good performance of our proposed HALNS can be explained by the suitable combination of a number of components and parameters tuning. It also shows that the new designed strategies and operators, the SROP and the SPP components and the algorithm's adaptiveness play a major role in obtaining such good results.

6. Conclusion

In this paper, we have introduced an efficient Hybrid Adaptive Large Neighborhood Search (HALNS) solution approach for the Team Orienteering Problem (TOP), an intensively studied variant of the vehicle routing problem with profits (VRPP). Computational results on two sets of standard benchmark instances for the TOP showed that our heuristic outperforms all state-of-the-art algorithms in terms of solution quality and/or computational time. The HALNS was able to provide the best known solution (BKS) for 387 small-scale instances over 387 and 333 BKS for the 333 large-scale instances with a new BKS. A future research avenue would be to adapt our solution approach to solve more VRPP variants, for example those dealing with time windows or arc routing variants.

Acknowledgments

This project was funded by the Canadian Natural Sciences and Engineering Research Council (NSERC) under grants and 2016-04482 and 2019-00094. This support is greatly acknowledged. We thank Ruslan Sadykov, Eduardo Uchoa and Thibault Vidal for sharing detailed information about their algorithms, computational environments and solutions. We thank the editor-in-chief, the area editor and two anonymous referees for their valuable comments on an earlier version of this article.

Alvarenga, G. B., Mateus, G. R., De Tomi, G., 2007. A genetic and set partitioning two-phase approach for the vehicle routing problem with time windows. Computers & Operations Research 34 (6), 1561–1584.

Angelelli, E., Archetti, C., Vindigni, M., 2014. The clustered orienteering problem. European Journal of Operational Research 238 (2), 404–414.

Archetti, C., Corberán, Á., Plana, I., Sanchis, J. M., Speranza, M. G., 2015. A matheuristic for the team orienteering arc routing problem. European Journal of Operational Research 245 (2), 392–401.

Archetti, C., Corberán, Á., Plana, I., Sanchis, J. M., Speranza, M. G., 2016. A branch-and-cut algorithm for the orienteering arc routing problem. Computers & Operations Research 66, 95–104.

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

Archetti, C., Speranza, M. G., 2015. Arc routing problems with profits. In: Laporte, G., Corberán, Á. (Eds.), Arc Routing: Problems, Methods, and Applications. Vol. 20 of MOS-SIAM Series on Optimization. Society for Industrial and Applied Mathematics, Philadelphia, PA, pp. 281–299.

Archetti, C., Speranza, M. G., Corberán, Á., Sanchis, J. M., Plana, I., 2013. The team orienteering arc routing problem. Transportation Science 48 (3), 442–457.

Archetti, C., Speranza, M. G., Vigo, D., 2014. Vehicle routing problems with profits. In: Vigo, D., Toth, P. (Eds.), Vehicle Routing: Problems, Methods, and Applications. Vol. 18 of Monographs on Discrete Mathematics and Applications. Society for Industrial and Applied Mathematics, Philadelphia, PA, pp. 273–297.

Arkin, E. M., Mitchell, J. S., Narasimhan, G., 1998. Resource-constrained geometric network optimization. In: Symposium on Computational Geometry. pp. 307–316.

Bianchessi, N., Mansini, R., Speranza, M. G., 2018. A branch-and-cut algorithm for the team orienteering problem. International Transactions in Operational Research 25 (2), 627–635.

Bouly, H., Dang, D.-C., Moukrim, A., 2010. A memetic algorithm for the team orienteering problem. 4OR 8 (1), 49–70.

Boussier, S., Feillet, D., Gendreau, M., 2007. An exact algorithm for team orienteering problems. 4OR 5 (3), 211–230.

Burke, E. K., Gendreau, M., Hyde, M., Kendall, G., Ochoa, G., Özcan, E., Qu, R., 2013. Hyper-heuristics: A survey of the state of the art. Journal of the Operational Research Society 64 (12), 1695–1724.

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

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

Christofides, N., Mingozzi, A., Toth, P., 1979. The vehicle routing problem. In: Christofides, N. (Ed.), Combinatorial optimization. John Wiley & Sons Canada, pp. 315–338.

Croes, G. A., 1958. A method for solving traveling-salesman problems. Operations Research 6 (6), 791–812.

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

Dang, D.-C., Guibadj, R. N., Moukrim, A., 2011. A PSO-based memetic algorithm for the team orienteering problem. In: European Conference on the Applications of Evolutionary Computation. Springer, pp. 471–480.

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

Demir, E., Bektaş, T., Laporte, G., 2012. An adaptive large neighborhood search heuristic for the pollution-routing problem. European Journal of Operational Research 223 (2), 346–359.

Desrochers, M., Desrosiers, J., Solomon, M., 1992. A new optimization algorithm for the vehicle routing problem with time windows. Operations Research 40 (2), 342–354.

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

Evers, L., Glorie, K., Van Der Ster, S., Barros, A. I., Monsuur, H., 2014. A two-stage approach to the orienteering problem with stochastic weights. Computers & Operations Research 43, 248–260.

Feillet, D., Dejax, P., Gendreau, M., 2005. Traveling salesman problems with profits. Transportation Science 39 (2), 188–205.

Ferreira, J., Quintas, A., Oliveira, J. A., Pereira, G. A., Dias, L., 2014. Solving the team orienteering problem: developing a solution tool using a genetic algorithm approach. In: Snášel, V., Krömer, P., Koeppen, M., Schaefer, G. (Eds.), Soft Computing in Industrial

Applications. Vol. 223 of Advances in Intelligent Systems and Computing. Springer, pp. 365–375.

Fialho, Á., Da Costa, L., Schoenauer, M., Sebag, M., 2008. Extreme value based adaptive operator selection. In: International Conference on Parallel Problem Solving from Nature. Springer, pp. 175–184.

Fischetti, M., Gonzalez, J. J. S., Toth, P., 1998. Solving the orienteering problem through branch-and-cut. INFORMS Journal on Computing 10 (2), 133–148.

Golden, B. L., Levy, L., Vohra, R., 1987. The orienteering problem. Naval Research Logistics 34 (3), 307–318.

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

Hammami, F., Rekik, M., Coelho, L. C., 2019. Exact and heuristic solution approaches for the bid construction problem in transportation procurement auctions with a heterogeneous fleet. Transportation Research Part E: Logistics and Transportation Review 127, 150–177.

Hemmelmayr, V. C., Cordeau, J.-F., Crainic, T. G., 2012. An adaptive large neighborhood search heuristic for two-echelon vehicle routing problems arising in city logistics. Computers & Operations Research 39 (12), 3215–3228.

Ilhan, T., Iravani, S. M., Daskin, M. S., 2008. The orienteering problem with stochastic profits. IIE Transactions 40 (4), 406–421.

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

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

Keller, J. M., Gray, M. R., Givens, J. A., 1985. A fuzzy k-nearest neighbor algorithm. IEEE Transactions on Systems, Man, and Cybernetics SMC-15 (4), 580–585.

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

Kim, B.-I., Li, H., Johnson, A. L., 2013. An augmented large neighborhood search method for solving the team orienteering problem. Expert Systems with Applications 40 (8), 3065–3072.

Lahyani, R., Gouguenheim, A.-L., Coelho, L. C., 2019. A hybrid adaptive large neighbourhood search for multi-depot open vehicle routing problems. International Journal of Production Research, 1–14.

Laporte, G., Martello, S., 1990. The selective travelling salesman problem. Discrete Applied Mathematics 26 (2-3), 193–207.

Laporte, G., Martín, I. R., 2007. Locating a cycle in a transportation or a telecommunications network. Networks 50 (1), 92–108.

Li, K., Fialho, A., Kwong, S., Zhang, Q., 2013. Adaptive operator selection with bandits for a multiobjective evolutionary algorithm based on decomposition. IEEE Transactions on Evolutionary Computation 18 (1), 114–130.

Li, Y., Chen, H., Prins, C., 2016. Adaptive large neighborhood search for the pickup and delivery problem with time windows, profits, and reserved requests. European Journal of Operational Research 252 (1), 27–38.

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

Maturana, J., Fialho, Á., Saubion, F., Schoenauer, M., Sebag, M., 2009. Extreme compass and dynamic multi-armed bandits for adaptive operator selection. In: 2009 IEEE Congress on Evolutionary Computation. IEEE, pp. 365–372.

Naccache, S., Côté, J.-F., Coelho, L. C., 2018. The multi-pickup and delivery problem with time windows. European Journal of Operational Research 269 (1), 353–362.

Orlis, C., Bianchessi, N., Roberti, R., Dullaert, W., 2020. The team orienteering problem with overlaps: An application in cash logistics. Transportation Science 54 (2), 470–487.

Özcan, E., Bilgin, B., Korkmaz, E. E., 2008. A comprehensive analysis of hyper-heuristics. Intelligent Data Analysis 12 (1), 3–23.

Pessoa, A., Sadykov, R., Uchoa, E., Vanderbeck, F., 2019. A generic exact solver for vehicle routing and related problems. In: Lodi, A., Nagarajan, V. (Eds.), Integer Programming and Combinatorial Optimization. Springer International Publishing, Cham, Switzerland, pp. 354–369.

Pessoa, A., Uchoa, E., Poggi de Aragão, M., 2009. A robust branch-cut-and-price algorithm for the heterogeneous fleet vehicle routing problem. Networks: An International Journal 54 (4), 167–177.

Pisinger, D., Ropke, S., 2007. A general heuristic for vehicle routing problems. Computers & Operations Research 34 (8), 2403–2435.

Poggi de Aragão, M., Viana, H., Uchoa, E., 2010. The team orienteering problem: Formulations and branch-cut and price. In: Erlebach, T., Lübbecke, M. (Eds.), 10th Workshop on Algorithmic Approaches for Transportation Modelling, Optimization, and Systems. Vol. 14 of OpenAccess Series in Informatics (OASIcs). Dagstuhl, Germany, pp. 142–155.

Prins, C., 2009. A GRASP × evolutionary local search hybrid for the vehicle routing problem. In: Pereira, F. B., Tavares, J. (Eds.), Bio-inspired algorithms for the vehicle routing problem. Vol. 161 of Studies in Computational Intelligence. Springer-Verlag Berlin Heidelberg, pp. 35–53.

Reinelt, G., 1991. TSPLIB – A traveling salesman problem library. ORSA Journal on Computing 3 (4), 376–384.

Renaud, J., Boctor, F. F., Laporte, G., 1996. An improved petal heuristic for the vehicle routeing problem. Journal of the operational Research Society 47 (2), 329–336.

Rochat, Y., Taillard, É. D., 1995. Probabilistic diversification and intensification in local search for vehicle routing. Journal of Heuristics 1 (1), 147–167.

Ropke, S., Pisinger, D., 2006. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science 40 (4), 455–472.

Sacramento, D., Pisinger, D., Ropke, S., 2019. An adaptive large neighborhood search metaheuristic for the vehicle routing problem with drones. Transportation Research Part C: Emerging Technologies 102, 289–315.

Shaw, P., 1998. Using constraint programming and local search methods to solve vehicle routing problems. In: International Conference on Principles and Practice of Constraint Programming. Springer, pp. 417–431.

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

Tang, H., Miller-Hooks, E., 2005. A tabu search heuristic for the team orienteering problem. Computers & Operations Research 32 (6), 1379–1407.

Tsakirakis, E., Marinaki, M., Marinakis, Y., Matsatsinis, N., 2019. A similarity hybrid harmony search algorithm for the team orienteering problem. Applied Soft Computing 80, 776–796.

Tsiligirides, T., 1984. Heuristic methods applied to orienteering. Journal of the Operational Research Society 35 (9), 797–809.

van Laarhoven, P. J. M., Aarts, E. H. L., 1987. Simulated annealing. Springer Netherlands, Dordrecht, pp. 7–15.

Vansteenwegen, P., Gunawan, A., 2019a. Other orienteering problem variants. In: Speranza, M. G., Oliveira, J. F. (Eds.), Orienteering Problems: Models and Algorithms for Vehicle Routing Problems with Profits. EURO Advanced Tutorials on Operational Research. Springer International Publishing, Cham, Switzerland, pp. 95–112.

Vansteenwegen, P., Gunawan, A., 2019b. State-of-the-art solution techniques for OP and TOP. In: Speranza, M. G., Oliveira, J. F. (Eds.), Orienteering Problems: Models and Algorithms for Vehicle Routing Problems with Profits. EURO Advanced Tutorials on Operational Research. Springer International Publishing, Cham, Switzerland, pp. 41–66.

Vansteenwegen, P., Souffriau, W., Berghe, G. V., Van Oudheusden, D., 2009a. A guided local search metaheuristic for the team orienteering problem. European Journal of Operational Research 196 (1), 118–127.

Vansteenwegen, P., Souffriau, W., Berghe, G. V., Van Oudheusden, D., 2009b. Metaheuristics for tourist trip planning. In: Geiger, M. J., Habenicht, W., Sevaux, M., Sörensen, K. (Eds.), Metaheuristics in the Service Industry. Vol. 624 of Lecture Notes in Economics and Mathematical Systems. Springer-Verlag Berlin Heidelberg, Berlin, Germany, pp. 15–31.

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

Vidal, T., Crainic, T. G., Gendreau, M., Prins, C., 2014. A unified solution framework for multi-attribute vehicle routing problems. European Journal of Operational Research 234 (3), 658–673.

Vidal, T., Maculan, N., Ochi, L. S., Vaz Penna, P. H., 2016. Large neighborhoods with implicit customer selection for vehicle routing problems with profits. Transportation Science 50 (2), 720–734.