CIRRELT

CIRRELT-2019-51

A Hybrid Adaptive Large Neighborhood Search Heuristic for the Team Orienteering Problem

Farouk Hammami
Monia Rekik
Leandro C. Coelho

November 2019

Document de travail également publié par la Faculté des sciences de l'administration de l'Université Laval, sous le numéro FSA-2019-016.

Bureau de Montréal
Université de Montréal
C.P. 6128, succ. Centre-Ville
Montréal (Québec) H3C 3J7
Tél. : 1-514-343-7575
Télécopie : 1-514-343-7121

Bureau de Québec
Université Laval,
2325, rue de la Terrasse
Pavillon Palasis-Prince, local 2415
Québec (Québec) G1V 0A6
Tél. : 1-418-656-2073
Télécopie : 1-418-656-2624

A Hybrid Adaptive Large Neighborhood Search Heuristic for the Team Orienteering Problem

Farouk Hammami*, Monia Rekik, Leandro C. Coelho

[1] Interuniversity Research Centre on Enterprise Networks, Logistics and Transportation (CIRRELT) and Department of Operations and Decision Systems, Université Laval, 2325 de la Terrasse, Québec, Canada G1V 0A6

Abstract. The Team Orienteering Problem (TOP) is a well-known NP-Hard problem in which one maximizes the collected profits for visiting some nodes. In this paper, we propose a new 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 when compared to competing algorithms in the literature on a large set of benchmark instances. HALNS identifies 386 over the 387 best known solutions (BKS) from the literature on a first dataset including small-scale instances and all the BKS for large-scale instances within very short computational times. Moreover, we prove 3 new optimal solutions for small-scale instances.

Keywords. Team orienteering problem, adaptive large neighborhood search, hybrid heuristic, homogeneous fleet, routing.

Acknowledgements. This project was funded by the Natural Sciences and Engineering Research Council of Canada (NSERC) under grants 2016-04482 and 2019-00094. This support is greatly acknowledged.

---

* Corresponding author: Farouk.Hammami@cirrelt.ca

Dépôt légal – Bibliothèque et Archives nationales du Québec
Bibliothèque et Archives Canada, 2019

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 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 21 existing state-of-the-art heuristics in terms of solution quality and/or computational time for the small-scale instances. To the best of our knowledge, only two methods from the literature have been tested on the large instances. Here, our algorithm finds all 82 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. In addition to that, we prove 3 new optimal solutions for small-scale instances.

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 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 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 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 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. Although the conducted computational experiments show that the developed solver outperforms the approach of Bianchessi et al. (2018) when tested on a 60 instances of the TOP in terms of computational time, they did not report solution values.

Although exact solution approaches allowed to solve 338 out of the 387 benchmark instances, they remain very time- and resource-consuming. 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 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.

Ke et al. (2016) proposed a new algorithm called Pareto Mimic Algorithm (PMA) for the TOP. The developed algorithm follows the general framework of the population-based meta-heuristic (Talbi, 2009). Numerical results shows 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, 21 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); Ke et al. (2016). Optimality is known for 338 of these instances, as reported by Bianchessi et al. (2018). On the large-scale benchmark instances set, BKS are provided by Ke et al. (2016).

3 Problem definition and mathematical formulation

We consider a directed graph $G = (V, A)$ where $V = \{1, \dots, N\} = V^* \cup \{1, N\}$ 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 can be formulated as follows:

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

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

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

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

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

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

$$B_i + \sum_{l \in L} x_{ij}^l - |V| \left(1 - \sum_{l \in L} x_{ij}^l \right) \leq B_j \qquad \forall (i,j) \in A \tag{7}$$

$$B_1 = 0 \tag{8}$$

$$B_N = \sum_{l \in L} \sum_{i \in V^*} y_i^l + 1 \tag{9}$$

$$B_i \leq B_N - 1 \qquad \forall i \in V^* \tag{10}$$

$$x_{ij}^l \in \{0,1\} \qquad \forall (i,j) \in A, l \in L \tag{11}$$

$$B_i \geq 0 \qquad \forall i \in V \tag{12}$$

$$y_i^l \in \{0,1\} \qquad \forall i \in V^*, l \in L. \tag{13}$$

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. Constraints (7) eliminate sub-tours and impose an order for visiting nodes. Constraint (8) implies that each route starts from the depot. Constraint (9) imply that order of the end depot is equal to the number of visited notes. Constraints (10) impose an upper bound on the order of visit of each node and strengthen the formulation. Finally, constraints (11)–(13) 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) and 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 particularity to be a VRP variant in which no node is mandatory to be visited (except the depots). 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 sub-route optimization problem where the objective is to find a more profitable sequence of nodes to replace a less profitable one. 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 sub-route optimization problem and the SPP are solved exactly by a commercial solver.

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 $t_{1i} + t_{iN} > D_{max}$.

After reducing the size of the problem, an initial solution is constructed using the nearest neighbor algorithm (Keller et al., 1985). Lines 3 and 4 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 iterations' counter. 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 N_{seg}, 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 $e^{\frac{f(s)-f(s_{adm})}{T}}$

Algorithm 1 Structure of the HALNS heuristic for the TOP

Apply a node elimination procedure
Construct initial solution s using the nearest neighbor algorithm
$s_{best} \leftarrow s$, $s_{adm} \leftarrow s$, $T \leftarrow T_0$, $iteration \leftarrow 0$, $iteration_{best} \leftarrow 0$
Initialize the scores of the node selection strategy and the removal and insertion operators
while $(iteration < iteration_{max}$ and $iteration_{best} < iteration_{best_{max}})$ **do**
    $j \leftarrow 0$
    while $(j < N_{seg})$ **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**
                Generate and solve a sub-route optimization problem
                $s_{best} \leftarrow s$, $iteration_{best} \leftarrow 0$
            else
                $iteration_{best} \leftarrow iteration_{best} + 1$
            end if
            $s_{adm} \leftarrow s$
        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
        $T \leftarrow T \times c$, $j \leftarrow j + 1$
    end while
    Update scores of the ALNS operators and insertion strategies
    $iteration \leftarrow iteration + 1$
end while
Generate and solve a SPP
Update and return s_{best}

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 sub-route optimization problem 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 iterations ($iteration_{max}$) 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. Each node selection strategy and removal/insertion operator k has a score $\pi_{k,q}$ in each run segment q (a finite number of iterations) and set to zero 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 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 is newly designed for the TOP. It 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 forming seq. Afterwards, a sub-route optimization problem is introduced and formulated as follows.

We model the problem with two sets of variables: binary variables χ_{ij} determining if arc

$(i, j) \in V_- \times V_-$ is served and continuous variables ζ_i to indicate the order of visit of node $i \in V_-$. The mathematical model for the sub-route optimization problem can be formulated as follows:

$$\max \sum_{i \in V_-} \sum_{j \in V_-} p_i \chi_{ij} \tag{14}$$

$$\text{s.t.} \sum_{j \in V_- : j \neq i} \chi_{ij} \leq 1 \qquad \forall i \in V_- \setminus \{d\} \tag{15}$$

$$\sum_{j \in V_- : j \neq o} \chi_{oj} = \sum_{i \in V_- : i \neq d} \chi_{id} = 1 \tag{16}$$

$$\sum_{j \in V_- : j \neq i} \chi_{ji} = \sum_{j \in V_- : j \neq i} \chi_{ij} \qquad i \in V_- \setminus \{o, d\} \tag{17}$$

$$\sum_{i \in V_- \setminus \{d\}} \sum_{j \in V_- \setminus \{o\}} t_{ij} \chi_{ij} \leq \mathfrak{D}_{seq} \tag{18}$$

$$\zeta_i + \chi_{ij} (1 + |V_-|) - |V_-| \leq \zeta_j \qquad \forall i \in V_-, j \in V_- \tag{19}$$

$$\zeta_o = 0 \tag{20}$$

$$\chi_{ij} \in \{0, 1\} \qquad \forall i \in V_-, j \in V_- \tag{21}$$

$$\zeta_i \geq 0 \qquad \forall i \in V_-. \tag{22}$$

The objective function (14) is to maximize the total collected profit. Constraints (15) imply that each node is visited at most once. Constraint (16) implies that the route starts from node o and ends at node d. Constraints (17) ensure the connectivity of the route. Constraint (18) ensures the route duration. Constraints (19) eliminate sub-tours. Constraint (20) implies that route must start from node o. Finally, constraints (21)–(22) define the domain of decision variables. One should note that *seq* is used as an initial solution when solving the sub-route optimization problem.

4.5 Set packing problem

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 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. 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:

$$\max \quad \sum_{r \in \mathfrak{R}} \varphi_r z_r \tag{23}$$

$$\text{s.t.} \quad \sum_{r \in \mathfrak{R}} a_{ri} z_r \leq 1 \qquad \forall i \in V^* \tag{24}$$

$$\sum_{r \in \mathfrak{R}} z_r = |L| \tag{25}$$

$$z_r \in \{0, 1\} \qquad \forall r \in \mathfrak{R}. \tag{26}$$

The objective function (23) maximizes the global profit. Constraints (24) imply that each node $i \in V^*$ can be served at most once. Constraint (25) imposes that the number of selected routes is equal to $|L|$. Finally, constraints (26) 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 i7 processor 7700-HQ, 3.80 GHz with 8 threads and 16 GB of RAM. CPLEX 12.9 was used as MIP solver to solve the SPP and the sub-route optimization problem. 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 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 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 21 state-of-the-art heuristics described in Table 1.

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

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

Name	Description	Reference
TMH	Tabu search	Tang and Miller-Hooks (2005)
GTP	Tabu search with penalty strategy	Archetti et al. (2007)
GTF	Tabu search with feasible strategy	Archetti et al. (2007)
FVNS	Fast Variable Neighborhood Search	Archetti et al. (2007)
SVNS	Slow Variable Neighborhood Search	Archetti et al. (2007)
SACO	Sequential Ant Colony Optimization	Ke et al. (2008)
DACO	Deterministic variant of Ant Colony Optimization	Ke et al. (2008)
RACO	Random variant of Ant Colony Optimization	Ke et al. (2008)
SiACO	Simultaneous variant of Ant Colony Optimization	Ke et al. (2008)
SkVNS	Skewed Variable Neighborhood Search	Vansteenwegen et al. (2009b)
GLS	Guided Local Search	Vansteenwegen et al. (2009a)
FPR	Fast variant of Path Relinking	Souffriau et al. (2010)
SPR	Slow variant of Path Relinking	Souffriau et al. (2010)
MA	Memetic Algorithm	Bouly et al. (2010)
PSOMA	Particle Swarm Optimization-based MA	Dang et al. (2011)
AuLNS	Augmented Large Neighborhood Search	Kim et al. (2013)
PSOiA	PSO-inspired Algorithm	Dang et al. (2013b)
MSA	Multi-start Simulated Annealing	Lin (2013)
PMA	Pareto Mimic Algorithm	Ke et al. (2016)
SHHS	Similarity Hybrid Harmony Search	Tsakirakis et al. (2019)
SHHS2	Second version of SHHS	Tsakirakis et al. (2019)

instances of the Traveling Salesman Problem (Reinelt, 1991) in which customers' profits were generated on the basis of three different methods. We refer the reader to Dang et al. (2013b) for more details. The methods of Dang et al. (2013b); 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 5,000 iterations without any improvement is considered. 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.

Table 2: Parameter tuning of our HALNS heuristic

Parameter	Description	Value
$iteration_{max}$	Maximum number of iterations per run segment	1,000/5,000 for small/large-scale instances
$iteration_{best_{max}}$	Maximum number of iterations without improvement	5,000
N_{seg}	Number of run segments	100
β	Number of nodes to remove	$\beta \in [1, 0.25\|V_+\|]$
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	$[2, \ldots, 15\%\|V_+^l\|]$

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 sub-route optimization problem, we set the time limit to 60 seconds.

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. The best obtained solution for each instance is reported under the column "Best".

As depicted in Table 3, HALNS is able to obtain the BKS for 53 instances over 54. Three heuristics were able to obtain the same results: AuLNS, PSOiA and PMA proposed respectively by Kim et al. (2013), Dang et al. (2013b) and Ke et al. (2016). Although 4 methods reported the same results, our HALNS beats them with an average computational time of 50.24 seconds versus 77.30, 218.58 and 109.30 seconds respectively required by AuLNS, PSOiA and PMA. Observe from Table 3 that the BKS for instance p4.4.n was obtained by TMH proposed by Tang and Miller-Hooks (2005). The authors report a BKS of 977 whereas our HALNS and four other methods report a solution value of 976.

For set 5, our HALNS identified the BKS for all 45 instances. This was also the case for 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	PMA	SHHS	SHHS2	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**
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**
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**
p4.2.d	**531**	517	530	**531**	**531**	**531**	**531**	**531**	530	**531**	528	526	**531**	**531**	**531**	**531**	**531**	**531**	**531**	**531**	528	**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**	611	613	**618**
p4.2.f	**687**	666	**687**	676	684	**687**	**687**	672	672	672	675	651	**687**	**687**	**687**	**687**	**687**	**687**	**687**	**687**	673	684	**687**
p4.2.g	**757**	749	751	756	750	753	**757**	756	756	756	750	734	**757**	**757**	**757**	**757**	**757**	**757**	**757**	**757**	753	**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**	816	827	**835**
p4.2.i	**918**	915	882	899	916	**918**	**918**	900	**918**	**918**	916	826	**918**	**918**	**918**	**918**	**918**	**918**	**918**	**918**	896	912	**918**
p4.2.j	**965**	914	946	962	962	962	**965**	962	962	962	962	939	962	**965**	**965**	**965**	**965**	**965**	962	**965**	936	957	**965**
p4.2.k	**1022**	963	1013	1013	1019	**1022**	**1022**	1016	1016	1016	1007	994	1013	**1022**	**1022**	**1022**	**1022**	**1022**	**1022**	**1022**	983	1010	**1022**
p4.2.l	**1074**	1022	1061	1058	1073	**1074**	1071	1070	1071	1069	1051	1051	1064	**1074**	1071	1071	**1074**	**1074**	1073	**1074**	1048	**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**	1078	1125	**1132**
p4.2.n	**1174**	1150	1169	1171	1159	1171	1168	1149	1158	1169	1124	1117	1161	1173	**1174**	**1174**	**1174**	**1174**	**1174**	**1174**	1152	1168	**1174**
p4.2.o	**1218**	1175	1180	1192	1216	**1218**	1215	1209	1198	1210	1195	1191	1206	**1218**	**1218**	**1218**	**1218**	**1218**	1217	**1218**	1171	1216	**1218**
p4.2.p	**1242**	1208	1226	1239	1239	1241	**1242**	1229	1233	1239	1237	1214	1240	**1242**	**1242**	1241	**1242**	**1242**	1241	**1242**	1211	**1242**	**1242**
p4.2.q	**1268**	1255	1252	1255	1265	1263	1263	1253	1252	1260	1239	1248	1257	1263	1267	1267	**1268**	**1268**	1259	**1268**	1245	1259	**1268**
p4.2.r	**1292**	1277	1281	1283	1283	1286	1288	1278	1278	1279	1279	1267	1278	1286	**1292**	**1292**	**1292**	**1292**	1290	**1292**	1269	1282	**1292**
p4.2.s	**1304**	1294	1296	1299	1300	1301	**1304**	**1304**	1303	**1304**	1295	1286	1293	1296	**1304**	**1304**	**1304**	**1304**	1300	**1304**	1284	1294	**1304**
p4.2.t	**1306**	**1306**	**1306**	**1306**	**1306**	**1306**	**1306**	**1306**	**1306**	**1306**	1305	1294	1299	**1306**	**1306**	**1306**	**1306**	**1306**	**1306**	**1306**	1302	**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**
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**
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**
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**
p4.3.g	**653**	646	651	652	**653**	**653**	**653**	652	**653**	652	651	644	**653**	**653**	**653**	**653**	**653**	**653**	**653**	**653**	651	**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**	715	**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**	796	**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**	854	858	**861**
p4.3.k	**919**	906	**919**	918	**919**	**919**	918	913	910	910	902	864	918	918	**919**	**919**	**919**	**919**	**919**	**919**	908	**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**	954	973	**979**
p4.3.m	**1063**	1005	1034	1049	1056	**1063**	1053	1039	1028	1046	1047	1030	1043	**1063**	**1063**	**1063**	**1063**	**1063**	**1063**	**1063**	1028	**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**	1093	1116	**1121**
p4.3.o	**1172**	1151	1156	1157	**1172**	1170	1170	1163	1167	1165	1136	1121	1165	1170	**1172**	**1172**	**1172**	**1172**	1170	**1172**	1149	1168	**1172**
p4.3.p	**1222**	1218	1207	1221	1208	**1222**	1221	1202	1207	1207	1200	1190	1209	1220	**1222**	**1222**	**1222**	**1222**	**1222**	**1222**	1213	**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**	1226	1251	**1253**
p4.3.r	**1273**	1265	1224	1269	1272	1272	1267	1263	1263	1263	1250	1239	1257	1272	**1273**	**1273**	**1273**	**1273**	1265	**1273**	1247	1269	**1273**
p4.3.s	**1295**	1282	1250	1294	1289	1293	1293	1291	1289	1291	1280	1279	1276	1287	**1295**	**1295**	**1295**	**1295**	1293	**1295**	1265	1285	**1295**
p4.3.t	**1305**	1288	1303	1304	1298	1304	**1305**	1304	1303	1304	1299	1290	1294	1299	**1305**	1304	**1305**	**1305**	1299	**1305**	1278	1302	**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**
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**
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**
p4.4.h	**571**	554	**571**	**571**	**571**	**571**	**571**	556	556	556	553	565	**571**	**571**	**571**	**571**	**571**	**571**	**571**	**571**	556	**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**	653	**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**	731	**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**	819	820	**821**
p4.4.l	**880**	875	878	878	879	**880**	**880**	877	875	875	876	852	875	879	**880**	**880**	**880**	**880**	**880**	**880**	859	878	**880**
p4.4.m	**919**	910	916	918	916	**919**	918	911	906	911	903	910	914	**919**	916	**919**	**919**	**919**	**919**	**919**	896	916	**919**
p4.4.n	**977**	**977**	972	976	968	968	961	956	956	956	948	942	953	969	969	969	976	976	975	976	948	971	976
p4.4.o	**1061**	1014	1057	1057	1051	**1061**	1036	1030	1021	1029	1030	937	1033	1057	**1061**	**1061**	**1061**	**1061**	**1061**	**1061**	1035	**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**	1112	1120	**1124**
p4.4.q	**1161**	1124	1148	1157	1160	**1161**	1145	1150	1137	1148	1149	1106	1139	1160	**1161**	**1161**	**1161**	**1161**	**1161**	**1161**	1129	1158	**1161**
p4.4.r	**1216**	1165	1203	1211	1207	1203	1200	1195	1195	1194	1193	1148	1196	1213	**1216**	**1216**	**1216**	**1216**	**1216**	**1216**	1170	**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**	1223	1252	**1260**
p4.4.t	**1285**	1255	1279	**1285**	1282	1279	1281	1281	1283	1281	1281	1250	1256	1280	**1285**	**1285**	**1285**	**1285**	**1285**	**1285**	1264	1281	**1285**

solution approaches proposed in Dang et al. (2013b); Ke et al. (2016); Kim et al. (2013) who obtained these solutions respectively within an average computational time of 49.50, 109.30 and 77.30 seconds vs only 21.10 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, 12 solutions approaches in addition to our HALNS were able to obtain all 15 BKS. 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 as reported in Table 9.

For set 7, in addition to AuLNS, PSOiA and PMA proposed respectively by Kim et al. (2013), Dang et al. (2013b) and Ke et al. (2016), our HALNS identified all 43 BKS. In

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	PMA	SHHS	SHHS2	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
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
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
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
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
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
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
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
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
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
p5.2.s	1340	1310	1330	1340	1340	1340	1340	1330	1330	1330	1325	1325	1340	1340	1330	1340	1340	1340	1340	1340	1325	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	1380	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
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
p5.2.w	1565	1560	1555	1565	1560	1560	1560	1555	1555	1555	1560	1560	1560	1560	1560	1560	1565	1565	1565	1565	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
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	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
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
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
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
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
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
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
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
p5.3.t	1260	1240	1250	1260	1260	1260	1260	1250	1255	1260	1260	1240	1260	1260	1260	1260	1260	1260	1260	1345	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
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
p5.3.w	1485	1465	1485	1485	1485	1485	1485	1465	1465	1465	1475	1460	1465	1485	1485	1485	1485	1485	1485	1485	1470	1480	1485
p5.3.x	1555	1530	1555	1555	1555	1555	1540	1535	1540	1540	1535	1520	1540	1550	1555	1555	1555	1555	1555	1555	1545	1550	1555
p5.3.y	1595	1580	1590	1595	1595	1595	1590	1590	1590	1590	1580	1590	1590	1590	1590	1595	1595	1595	1590	1595	1590	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	1620	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
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
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
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
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
p5.4.s	1030	1000	1025	1030	1030	1030	1030	1030	1030	1030	1020	1020	1005	1025	1030	1030	1030	1030	1030	1030	1025	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
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
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
p5.4.w	1390	1380	1375	1390	1390	1390	1390	1380	1390	1380	1380	1330	1380	1390	1380	1385	1390	1390	1390	1390	1380	1385	1390
p5.4.x	1450	1410	1440	1450	1450	1450	1450	1450	1450	1450	1440	1410	1430	1450	1450	1450	1450	1450	1450	1450	1435	1445	1450
p5.4.y	1520	1520	1520	1520	1520	1520	1520	1510	1510	1500	1500	1485	1520	1520	1520	1520	1520	1520	1520	1520	1485	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

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	PMA	SHHS	SHHS2	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
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
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
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
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	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
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
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
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
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
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
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
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
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
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

terms of computational time, only 46.90 seconds in average were required by our algorithm to obtain the BKS values for instances from Set 7 versus 66.80, 97.47 and 54.60 seconds required respectively by AuLNS, PSOiA and PMA to obtain 43 best solutions values over the 43 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. 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

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	PMA	SHHS	SHHS2	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**
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**
p7.2.f	**387**	382	**387**	**387**	**387**	**387**	**387**	**387**	**387**	**387**	**387**	340	**387**	**387**	**387**	**387**	**387**	**387**	**387**	**387**	384	**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**	453	**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**	520	**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**	572	579	**580**
p7.2.j	**646**	638	643	644	643	644	**646**	**646**	**646**	**646**	632	633	**646**	**646**	**646**	**646**	**646**	**646**	**646**	**646**	637	641	**646**
p7.2.k	**705**	702	702	**705**	704	**705**	**705**	704	704	704	700	691	702	**705**	**705**	**705**	**705**	**705**	**705**	**705**	678	700	**705**
p7.2.l	**767**	**767**	758	**767**	759	**767**	**767**	**767**	**767**	**767**	758	748	759	**767**	**767**	**767**	**767**	**767**	**767**	**767**	742	761	**767**
p7.2.m	**827**	817	**827**	824	824	**827**	**827**	**827**	**827**	**827**	**827**	798	816	**827**	**827**	**827**	**827**	**827**	**827**	**827**	794	**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**	859	878	**888**
p7.2.o	**945**	914	933	**945**	**945**	**945**	**945**	**945**	940	941	928	897	932	**945**	**945**	**945**	**945**	**945**	**945**	**945**	925	941	**945**
p7.2.p	**1002**	987	1000	**1002**	**1002**	**1002**	**1002**	991	993	993	955	954	993	**1002**	**1002**	**1002**	**1002**	**1002**	**1002**	**1002**	972	997	**1002**
p7.2.q	**1044**	1017	1041	1043	1038	**1044**	1043	1042	1043	1043	1029	1031	1043	**1044**	**1044**	**1044**	**1044**	**1044**	1043	**1044**	1005	1043	**1044**
p7.2.r	**1094**	1067	1091	1088	**1094**	**1094**	**1094**	1093	1088	**1094**	1069	1075	1076	**1094**	**1094**	**1094**	**1094**	**1094**	1093	**1094**	1052	1080	**1094**
p7.2.s	**1136**	1116	1123	1128	**1136**	**1136**	**1136**	**1136**	1134	1131	1118	1102	1125	**1136**	**1136**	**1136**	**1136**	**1136**	1135	**1136**	1094	1125	**1136**
p7.2.t	**1179**	1165	1172	1174	1168	**1179**	**1179**	**1179**	**1179**	**1179**	1154	1142	1168	1175	**1179**	**1179**	**1179**	**1179**	1172	**1179**	1128	1160	**1179**
p7.3.h	**25**	416	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	418	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**	**25**
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**
p7.3.j	**564**	563	**564**	**564**	562	**564**	**564**	**564**	**564**	**564**	543	539	560	**564**	**564**	**564**	**564**	**564**	**564**	**564**	558	**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**	632	**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**	681	**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**
p7.3.n	**820**	789	810	811	814	**820**	**820**	819	819	**820**	804	789	813	**820**	**820**	**820**	**820**	**820**	**820**	**820**	809	819	**820**
p7.3.o	**874**	**874**	873	865	871	**874**	**874**	**874**	**874**	**874**	841	833	859	**874**	**874**	**874**	**874**	**874**	**874**	**874**	856	**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**	915	926	**929**
p7.3.q	**987**	966	976	**987**	978	**987**	**987**	**987**	**987**	**987**	966	945	970	**987**	**987**	**987**	**987**	**987**	**987**	**987**	972	**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**	1003	1019	**1026**
p7.3.s	**1081**	1061	**1081**	**1081**	1079	1079	**1081**	**1081**	**1081**	1077	1070	1054	1076	**1081**	**1081**	**1081**	**1081**	**1081**	**1081**	**1081**	1056	1076	**1081**
p7.3.t	**1120**	1098	1114	1116	1112	1115	1118	1117	1103	1117	1109	1080	1111	1118	**1120**	**1120**	**1120**	**1120**	1119	**1120**	1092	1114	**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**
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**
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**
p7.4.k	**520**	503	**520**	**520**	518	**520**	**520**	**520**	**520**	**520**	518	511	518	518	**520**	**520**	**520**	**520**	**520**	**520**	518	518	**520**
p7.4.l	**590**	576	**590**	588	588	**590**	**590**	**590**	**590**	**590**	575	573	581	**590**	**590**	**590**	**590**	**590**	**590**	**590**	576	**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**
p7.4.n	**730**	726	723	721	715	**730**	**730**	725	725	726	723	698	723	**730**	726	726	**730**	**730**	**730**	**730**	726	726	**730**
p7.4.o	**781**	776	772	778	770	**781**	**781**	778	**781**	778	778	761	780	780	**781**	**781**	**781**	**781**	**781**	**781**	776	779	**781**
p7.4.p	**846**	832	841	839	**846**	**846**	**846**	**846**	838	842	841	803	842	**846**	**846**	**846**	**846**	**846**	**846**	**846**	834	**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**	899	**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**	952	**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**	1016	**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**	1063	**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	PMA	SHHS	SHHS2	HALNS
4	54	5	16	15	22	35	30	12	12	13	7	6	17	35	49	48	**53**	**53**	38	**53**	10	26	**53**
5	45	12	26	44	40	42	42	31	31	30	21	9	33	40	40	43	**45**	**45**	44	**45**	34	40	**45**
6	15	9	12	14	**15**	**15**	**15**	11	11	13	10	4	12	**15**	**15**	**15**	**15**	**15**	**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**	9	24	**43**
All	157	34	69	93	94	127	128	80	81	84	44	21	78	126	146	146	**156**	**156**	133	**156**	68	105	**156**

these instances to the BKS computed as follows:

$$Gap(\%) = \sum_{ins=1}^{157} \left(\frac{\frac{BKS_{ins} - S_{ins}}{S_{ins}}}{157} \right) \tag{27}$$

where BKS_{ins} is 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 for each set.

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	PMA	SHHS	SHHS2	HALNS
4	54	2.07130	0.89585	0.49432	0.28792	0.12035	0.31620	0.91932	1.00903	0.78782	1.51434	3.13737	0.73788	0.11630	0.03055	0.02632	**0.00190**	**0.00190**	0.07532	**0.00190**	1.67613	0.26990	**0.00190**
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**	0.23166	0.03751	**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**	**0.00000**
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**	1.56445	0.27579	**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	**0.00065**	**0.00065**	0.03651	**0.00065**	1.07138	0.17912	**0.00065**

Table 9: Average CPU time (s) 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	PMA	SHHS	SHHS2	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	109.30	107.30	109.70	**50.24**
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	22.90	30.40	28.10	**21.10**
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	36.40	14.50	12.20	**13.41**
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	54.60	82.50	82.50	**46.90**
Avg	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	55.80	58.68	58.13	**32.91**

5.3.1 New optimal solutions

Bianchessi et al. (2018) reported in their work that the number of open instances is equal to 49 however according to the previous published results by El-Hajj et al. (2016) the number of open instances is 45. Hence, in order to reduce this number and provide the literature with new optimal solutions, we addressed these open instances by solving them via the branch-and-cut procedure of CPLEX applied to model M_{TOP} with a time limit of 12 hours. In Table 10 we report the instances solved to optimality for the first time. We present in Table 10 for each instance its name, number of nodes, number of vehicles, the maximum time limit, the optimal solution and the computational time in seconds.

Table 10: New optimal solutions for the small-scale benchmark instances

Instance	$\|V\|$	$\|L\|$	D_{max}	Optimal	CPU (s)
p4.3.o	100	3	63.30	1172	36801.16
p4.3.p	100	3	66.70	1222	8865.11
p4.3.q	100	3	70.00	1253	16925.14

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. In Table 11, we report the solutions values reported by Dang et al. (2013b) and Ke et al. (2016) as well as those of our HALNS. Table 11 also reports for each instance the BKS and the best CPU in seconds required to identify the BKS.

The results reported in Table 11 show that the PSOiA requires more computational time in average to reach its best solution value when compared to PMA and HALNS. The average computational time of the PSOiA is 11,031.04 seconds which is very high against 1,004.15 and 843.66 seconds required respectively by the PMA et HALNS. This proves that both PMA and HALNS converge quickly when compared to PSOiA. Moreover, the best computational times reported for each instance show that the PMA and HALNS outperforms PSOiA in terms of computational time for all instances. Observe that for the 81 instances where our heuristic and PMA reach the BKS, HALNS was faster than PMA for 55 instances. In terms of solution quality, the number of BKS reported by the PSOiA, the PMA and our heuristic, over the 82 instances is respectively equal to 71, 81 and 82. Observe that our solution approach reports a new solution for the instance rd400_gen2_m3 with a value of 12,646 which proves that our heuristic is very competitive and deals well even with large-scale instances.

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 386 small-scale instances over 387 and 82 BKS for the 82 large-scale instances with a new BKS. We also proved 3 new optimal solutions. 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.

References

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

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 (ATMOS'10), Dagstuhl, Germany. pp. 142–155.

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, 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, 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. Society for Industrial and Applied Mathematics, Philadelphia, PA. volume 20, 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, 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. Society for Industrial and Applied Mathematics, Philadelphia, PA. volume 18 of *Monographs on Discrete Mathematics and Applications*, 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, 627–635.

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

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

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

Chao, I.M., Golden, B.L., Wasil, E.A., 1996. The team orienteering problem. European Journal of Operational Research 88, 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, 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, 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, 346–359.

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, 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: Soft Computing in Industrial Applications. Springer, pp. 365–375.

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

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

Gunawan, A., Lau, H.C., Vansteenwegen, P., 2016. Orienteering problem: A survey of recent variants, solution approaches and applications. European Journal of Operational Research 255, 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, 3215–3228.

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

Ke, L., Archetti, C., Feng, Z., 2008. Ants can solve the team orienteering problem. Computers & Industrial Engineering 54, 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, 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, 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, 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, 193–207.

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

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, 27–38.

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

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

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, 167–177.

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

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

Rochat, Y., Taillard, É.D., 1995. Probabilistic diversification and intensification in local search for vehicle routing. Journal of Heuristics 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, 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, 1853–1859.

Talbi, E.G., 2009. Metaheuristics: from design to implementation. volume 74. John Wiley & Sons, New Jersey, United States of America.

Tang, H., Miller-Hooks, E., 2005. Algorithms for a stochastic selective travelling salesperson problem. Journal of the Operational Research Society 56, 439–452.

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, 797–809.

Vansteenwegen, P., Gunawan, A., 2019a. Other orienteering problem variants, in: Orienteering Problems: Models and Algorithms for Vehicle Routing Problems with Profits. Springer International Publishing, Cham, Switzerland. EURO Advanced Tutorials on Operational Research, pp. 95–112.

Vansteenwegen, P., Gunawan, A., 2019b. State-of-the-art solution techniques for OP and TOP, in: Orienteering Problems: Models and Algorithms for Vehicle Routing Problems with Profits. Springer International Publishing, Cham, Switzerland. EURO Advanced Tutorials on Operational Research, 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, 118–127.

Vansteenwegen, P., Souffriau, W., Berghe, G.V., Van Oudheusden, D., 2009b. Metaheuristics for tourist trip planning, in: Metaheuristics in the Service Industry. Springer, Berlin, Germany, pp. 15–31.

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

Table 11: Results for the large-scale benchmark instances

Instance	Best Known		PSOiA		PMA		HALNS	
	Sol	CPU (s)	Sol	CPU (s)	Sol	CPU (s)	Sol	CPU (s)
cmt101c_m3	**1300**	42.00	**1300**	111.11	**1300**	42.00	**1300**	99.03
cmt151b_m3	**1385**	169.9	**1385**	754.01	**1385**	169.90	**1385**	200.20
cmt151c_m2	**1964**	291.76	1963	1799.64	**1964**	368.50	**1964**	291.76
cmt151c_m3	**1916**	410.36	**1916**	1376.24	**1916**	441.50	**1916**	410.36
cmt151c_m4	**1880**	571.69	**1880**	881.11	**1880**	826.50	**1880**	571.69
cmt200b_m2	**2096**	201.41	**2096**	4180.99	**2096**	669.40	**2096**	201.41
cmt200b_m3	**2019**	406.66	**2019**	2711.66	**2019**	1351.70	**2019**	406.66
cmt200b_m4	**1894**	360.34	**1894**	1515.19	**1894**	974.50	**1894**	360.34
cmt200c_m2	**2818**	409.36	**2818**	7320.26	**2818**	1048.30	**2818**	409.36
cmt200c_m3	**2766**	300.01	**2766**	4217.29	**2766**	1200.00	**2766**	300.01
cmt200c_m4	**2712**	403.99	**2712**	3004.10	**2712**	1411.40	**2712**	403.99
eil101b_m3	**916**	123.58	**916**	134.39	**916**	160.60	**916**	123.58
eil101c_m2	**1305**	250.50	**1305**	452.79	**1305**	250.50	**1305**	292.36
eil101c_m3	**1251**	95.00	**1251**	227.61	**1251**	95.00	**1251**	118.54
gil262a_m2	**4078**	501.96	**4078**	5907.29	**4078**	2100.00	**4078**	501.96
gil262a_m4	**3175**	139.36	**3175**	271.83	**3175**	222.60	**3175**	139.36
gil262b_m2	**8081**	605.74	**8081**	7473.18	**8081**	1267.80	**8081**	605.74
gil262b_m3	**7585**	498.05	**7585**	7276.80	**7585**	1027.20	**7585**	498.05
gil262b_m4	**6781**	342.81	**6781**	4878.64	**6781**	912.60	**6781**	342.81
gil262c_m2	**11030**	786.29	**11030**	27500.87	**11030**	1309.00	**11030**	786.29
gil262c_m3	**10757**	723.19	**10757**	14553.76	**10757**	1375.60	**10757**	723.19
gil262c_m4	**10281**	561.41	**10281**	8472.01	**10281**	1997.00	**10281**	561.41
bier127_gen1_m2	**106**	159.95	**106**	1153.87	**106**	673.50	**106**	159.95
bier127_gen1_m3	**103**	237.41	**103**	591.89	**103**	470.10	**103**	237.41
bier127_gen2_m2	**5464**	398.80	**5464**	1132.57	**5464**	398.80	**5464**	402.71
bier127_gen2_m3	**5393**	291.27	**5393**	648.08	**5393**	359.30	**5393**	291.27
bier127_gen2_m4	**5123**	383.80	5122	657.57	**5123**	383.80	**5123**	401.36
bier127_gen3_m2	**2885**	296.70	**2885**	1301.27	**2885**	296.70	**2885**	563.10
bier127_gen3_m3	**2706**	290.84	**2706**	711.74	**2706**	509.60	**2706**	290.84
bier127_gen3_m4	**2402**	227.70	**2402**	680.79	**2402**	227.70	**2402**	232.88
gil262_gen1_m3	**101**	482.60	**101**	1769.31	**101**	482.60	**101**	754.03
gil262_gen1_m4	**78**	123.50	**78**	155.76	**78**	123.50	**78**	199.11
gil262_gen2_m2	**7498**	403.12	**7498**	7356.65	**7498**	742.10	**7498**	403.12
gil262_gen2_m3	**5615**	391.54	**5615**	3304.55	**5615**	1163.80	**5615**	391.54
gil262_gen3_m2	**7183**	284.20	**7183**	9129.30	**7183**	284.20	**7183**	566.07
gil262_gen3_m4	**2507**	249.61	**2507**	276.42	**2507**	308.80	**2507**	249.61
gr229_gen1_m4	**223**	46.60	**223**	11922.02	**223**	46.60	**223**	496.20
gr229_gen2_m3	**11566**	711.56	**11566**	14197.21	**11566**	1665.30	**11566**	711.56
gr229_gen2_m4	**11355**	401.36	**11355**	18799.50	**11355**	2272.00	**11355**	401.36
gr229_gen3_m3	**8056**	998.67	**8056**	14090.06	**8056**	1065.30	**8056**	998.67
gr229_gen3_m4	**7651**	781.50	7621	11399.71	**7651**	781.50	**7651**	881.57
kroA150_gen2_m2	**4335**	431.10	**4335**	892.98	**4335**	495.90	**4335**	431.10
kroA150_gen3_m3	**2726**	174.03	**2726**	538.01	**2726**	597.20	**2726**	174.03
kroA200_gen1_m4	**81**	116.22	**81**	560.29	**81**	503.40	**81**	116.22
kroB200_gen1_m2	**111**	317.53	**111**	2344.53	**111**	663.90	**111**	317.53
kroB200_gen2_m2	**6185**	426.70	**6185**	3467.26	**6185**	426.70	**6185**	487.02
kroB200_gen2_m4	**4944**	396.09	**4944**	640.66	**4944**	582.40	**4944**	396.09
kroB200_gen3_m2	**4765**	500.11	**4765**	6306.62	**4765**	513.90	**4765**	500.11
kroB200_gen3_m3	**3028**	570.02	**3028**	1713.88	**3028**	741.50	**3028**	570.02
lin318_gen1_m2	**180**	1168.30	**180**	20667.24	**180**	1168.30	**180**	2906.27
lin318_gen1_m3	**149**	221.21	**149**	9014.64	**149**	721.40	**149**	221.21
lin318_gen2_m2	**9544**	1924.60	**9544**	23804.82	**9544**	1924.60	**9544**	2002.18
lin318_gen2_m3	**7807**	1413.50	7786	9773.63	**7807**	1413.50	**7807**	1633.24
lin318_gen3_m2	**7936**	631.81	**7936**	44029.00	**7936**	1547.30	**7936**	631.81
lin318_gen3_m4	**3797**	140.04	**3797**	1446.26	**3797**	970.70	**3797**	140.04
pr136_gen1_m2	**63**	107.50	**63**	451.13	**63**	107.50	**63**	368.14
pr136_gen2_m2	**3646**	154.75	3641	601.31	**3646**	281.60	**3646**	154.75
pr264_gen1_m4	**107**	203.99	**107**	503.07	**107**	289.80	**107**	203.99
pr264_gen2_m2	**6635**	656.10	**6635**	2048.20	**6635**	719.00	**6635**	656.10
pr264_gen2_m3	**6420**	555.02	**6420**	938.39	**6420**	859.00	**6420**	555.02
pr264_gen2_m4	**5584**	300.71	**5584**	590.79	**5584**	663.20	**5584**	300.71
pr264_gen3_m3	**2772**	922.50	**2772**	1037.51	**2772**	922.50	**2772**	1569.28
pr299_gen1_m2	**139**	366.25	**139**	4775.93	**139**	573.40	**139**	366.25
pr299_gen1_m3	**111**	431.02	**111**	1303.73	**111**	506.00	**111**	431.02
pr299_gen1_m4	**84**	201.36	**84**	383.48	**84**	340.30	**84**	201.36
pr299_gen2_m3	**6018**	704.50	**6018**	1446.05	**6018**	909.70	**6018**	704.50
pr299_gen2_m4	**4457**	263.01	**4457**	593.41	**4457**	767.70	**4457**	263.01
pr299_gen3_m2	**5729**	713.53	**5729**	11872.55	**5729**	1489.50	**5729**	713.53
pr299_gen3_m3	**3655**	678.29	**3655**	2705.82	**3655**	1058.20	**3655**	678.29
pr299_gen3_m4	**2268**	286.25	**2268**	455.64	**2268**	402.40	**2268**	286.25
rat195_gen2_m2	**5148**	345.91	**5148**	2156.98	**5148**	886.60	**5148**	345.91
rat195_gen3_m3	**2574**	288.37	**2574**	721.82	**2574**	369.50	**2574**	288.37
ts225_gen2_m2	**5859**	700.55	**5859**	2998.43	**5859**	759.60	**5859**	700.55
rd400_gen2_m2	**13045**	3220.60	12993	77049.22	**13045**	3220.60	**13045**	4203.21
rd400_gen2_m3	**12646**	2852.70	12645	53707.14	12645	2852.70	**12646**	2901.63
rd400_gen2_m4	**12032**	3299.30	**12032**	42001.58	**12032**	3299.30	**12032**	4031.71
rd400_gen1_m2	**232**	3066.60	230	56767.29	**232**	3066.60	**232**	4005.69
rd400_gen1_m3	**224**	2844.00	222	62476.08	**224**	2844.00	**224**	4263.27
rd400_gen1_m4	**213**	1985.40	**213**	34744.80	**213**	1985.40	**213**	3103.02
rd400_gen3_m2	**12431**	2418.40	12428	96178.70	**12431**	2418.40	**12431**	2901.71
rd400_gen3_m3	**11639**	3500.00	**11639**	68074.77	**11639**	3500.00	**11639**	3709.18
rd400_gen3_m4	**10436**	3500.00	10417	48462.77	**10436**	3500.00	**10436**	3766.10
Average				**11,031.04**		**1,004.15**		**843.66**