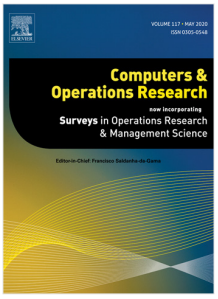
A Large Neighborhood Search-based approach to tackle the very large scale Team Orienteering Problem in industrial context

Charly Chaigneau, Nathalie Bostel, Axel Grimault

Journal Pre-proof



A Large Neighborhood Search-based approach to tackle the very large scale Team Orienteering Problem in industrial context

Charly Chaigneau, Nathalie Bostel, Axel Grimault

PII: S0305-0548(24)00426-X
DOI: https://doi.org/10.1016/j.cor.2024.106954
Reference: CAOR 106954

To appear in: *Computers and Operations Research*

Received date : 12 June 2024
Revised date : 20 November 2024
Accepted date : 11 December 2024

Please cite this article as: C. Chaigneau, N. Bostel and A. Grimault, A Large Neighborhood Search-based approach to tackle the very large scale Team Orienteering Problem in industrial context. *Computers and Operations Research* (2024), doi: https://doi.org/10.1016/j.cor.2024.106954.

A Large Neighborhood Search-based approach to tackle the very large scale Team Orienteering Problem in industrial context

CHAIGNEAU Charly[a], BOSTEL Nathalie[a], GRIMAULT Axel[b,*]

[a]*Nantes Université, LS2N, UMR 6004, F-44000, Nantes, France*
[b]*Université d'Angers, LARIS, SFR MATHSTIC, F-49000, Angers, France*

Abstract

The Team Orienteering Problem (TOP) is an optimization problem belonging to the class of Vehicle Routing Problem with Profits in which the objective is to maximize the total profit collected by visiting customers while being limited to a time limit. This paper deals with the very large scale TOP in an industrial context. In this context, computing time is decisive and classical methods may fail to provide good solutions in a reasonable computational time. To do so, we propose a Large Neighborhood Search (LNS) combined with various mechanisms in order to reduce the computational time of the method. It is applied on classical sets of instances from the literature and on a new set of very large scale instances ranging from 1001 to 5395 customers that we adapted from Kobeaga et al. (2017). On the small scale set of instances, most best-known solutions are found. On the large scale set of instances, three new best-known solutions are found while the algorithm quickly gets more than half of the other best-known solutions.

Keywords: Team Orienteering Problem, Very large scale, Vehicle Routing Problem, Clustering, Large Neighborhood Search

1. Introduction

The Orienteering Problem (OP) is an optimization problem which was first introduced in 1987 by Golden et al. (1987). Also known as the Selective Travelling Salesman Problem (Laporte and Martello, 1990), OP belongs to the class of Vehicle Routing Problem (VRP) with Profits. Like the VRP, OP is NP-Hard (Golden et al., 1987). In the OP, every customer has a profit which is collected if it is visited by the vehicle. It differs from the VRP as the vehicle is constrained by a time limit, meaning that it is usually not possible to visit every customer. Thus, the objective is then to maximize the total profit collected while respecting the time limit. The Team Orienteering Problem (TOP) is an extension of the OP where several vehicles are considered. TOP is used to modelize several applications, in particular the Tourist Trip Design Planning (TTDP)(Vansteenwegen, 2007). In TTDP, the aim is to build a tourist trip itinerary based on a list of points of interest. As tourists usually can't visit every point of interest, the goal is to give them the best itinerary which can be characterised by their location, their preferences, the period of the year and so on. A survey on this problem can be found on Gavalas et al. (2014) and Ruiz-Meza and Montoya-Torres (2022).

This paper aims at addressing an industrial problem that can be modelized as an extension of the TOP: the subsurface imaging problem. In our context, computing time is decisive as it can be necessary to solve the problem several times an hour. Thus, we seek to obtain solutions under a budget time of 10 minutes. Industrial instances can also have more customers or more vehicles than in classical literature instances, making it even harder to solve efficiently. As in Arnold et al. (2019), we define the 'very large scale' VRP as a VRP problem where we consider several thousands of customers since the term 'large scale' is used to describe problems of several hundreds of customers. The very large scale char-

*Corresponding author
Email address: axel.grimault@univ-angers.fr (GRIMAULT Axel)

acteristic has not been much tackled even if it became recently a new source of dedicated researches (Arnold et al., 2019, Accorsi and Vigo, 2021). The main aim of those articles is to obtain very good solutions in a time allowing the proposed algorithm to be used in the industrial context. To do so, heuristic approaches are used and paired with various mechanisms in order to reduce the computational time while preserving the quality of the final solution. The recent interest for those problems can be explained by the technological progress which eases the treatment of such problems. However, even if very large scale instances are tackled in some works in the OP context (Kobeaga et al., 2017), to our knowledge, this paper is the first one to address the very large scale TOP.

To deal with it, we propose a Large Neighborhood Search (LNS) combined with local search, simulated annealing, and with specific mechanisms to get very good solutions in a reasonable time. The remaining of this paper is organized as follows. First, in Section 2, we describe the industrial context. In Section 3, we discuss about the state-of-the-art algorithms for both the TOP and the very large scale VRP. In Section 4, we present one mathematical formulation of the problem. Then, in Section 5, we describe the LNS and we evaluate it in Section 6. We also propose an extension of several instances used in the OP to build a new set of very large scale instances for the TOP. Results on industrial instances are then evaluated in Section 7. Finally, we conclude this paper in Section 8 and propose some perspectives of research to better deal with such problems.

2. Industrial context

We are interested in a vehicle routing problem that arises in the subsurface imaging field. The aim of this field is to gather data that enables geologist to understand the structure of the subsurface, allowing environmental site characterization. Geophysical imaging provides essential knowledge on the structure of the subsurface. Applications can be found in geothermics or in the process of carbon capture. In the subsurface imaging field, several vehicles need to cover large areas to collect, on visited points, data related to the subsurface composition. These operations can last for several weeks with hundreds of thousands points that need to be visited to ensure a relevant cartography.

To this end, equipment getting data related to the subsurface structure are first deployed on the area of interest. Because of the size of the area to be studied, the whole area is not equiped and it is then needed to reposition the equipment throughout the study. Vehicles then go from point to point to collect geophysical data and to allow the equipment repositioning. Even if not every point is not available at the begining of the harvesting, problems of several thousands of points are still considered. One of the specificities of the problem is the number of hazards that happens during the gathering operation: vehicles may need maintenance operations, new areas may open and new available points appear regularly. Because of this, it is needed to update regularly the planning to ensure the quality of the process. This is defined as having the highest productivity possible, computed as the number of points visited during the next hour.

Thus, this problem can be modelized as a Team Orienteering Problem with multi-depot and open routes (MDOTOP), where each point has a profit of 1. The aim of our work is to propose, in a budget time of 10 minutes, solutions of high quality. For technical reasons, it is not possible to update continuously the planning, and this duration is associated to the time when routes can be updated.

3. Literature review

To our knowledge, the MDOTOP is not addressed in the literature. Thus, we review the Team Orienteering Problem. The TOP is a routing problem consisting in the maximization of the total profit collected by several vehicles on customers over a time limit. It is a well-studied problem but it still remains difficult to solve. Both exact and heuristic methods has been applied to the problem, heuristics being the main approach to tackle it. We review both of these methods applied to at least one of the two sets of instances from the literature: the small scale one proposed by Chao et al. (1996) and the large scale one introduced by Dang et al. (2013a). Then, we review the very large scale VRP. In this context, several

thousands of customers must be visited and classical methods need to be adapted to scale accordingly.

3.1. Exact methods

Few exact methods have been developed to tackle the TOP. Among these methods, Branch & Cut (BC) (Dang et al., 2013b, Bianchessi et al., 2018) has shown good results recently by finding 327 optimal solutions out of the 387 tested on the classical set of instances proposed by Chao et al. (1996). Branch & Price (BP) has also been used to tackle the TOP (Boussier et al., 2007, Keshtkaran et al., 2015) but results obtained were not among the best. Overall, the combination of both methods, called Branch & Cut & Price (BCP), is the method which is the most used among exact methods (Poggi et al., 2010, Keshtkaran et al., 2015, Pessoa et al., 2019, Orlis et al., 2020). In particular, the state-of-the-art method uses a BCP to solve the TOP with overlaps (Orlis et al., 2020). It manages to find optimality in 371 out of the 387 instances, and closes 33 open instances. Finally, cutting-plane methods (El-Hajj et al., 2016, Assunção and Mateus, 2019) have also been applied to the TOP. The latter solves 341 instances to optimality, showing that cutting-plane algorithms can also be used to tackle the TOP efficiently. Overall, exact methods have only been applied to the small scale set of instances (Chao et al., 1996) as obtaining very good results on this set is still a challenging task. Obvious perspectives will be to build efficient exact methods that can tackle problems from the large scale set of instances.

3.2. Heuristic approaches

Based on its complexity, heuristic approaches are more suitable to tackle the TOP. The first heuristic applied to the TOP is developed by Chao et al. (1996) and is compared to a modified version of Tsiligirides' heuristic (Tsiligiridis, 1984) which was applied to the stochastic OP. Results showed that the proposed heuristic was better than the adapted one.

Later, different methods were applied to the TOP. Tabu Search (TS) was one of the first method used. In Tang and Miller-Hooks (2005), the authors embedded the TS in an adapative memory procedure. According to the authors, this procedure allows diversity in the pool of solutions. The TS alternates between small and large neighborhood which enables the heuristic to be fast while not loosing quality. In their TS, greedy procedures are paired with random ones. Archetti et al. (2007) developed two variants of the TS algorithm. Both variants are mostly distinguished by the fact that one considers infeasible solutions while the other does not.

In the same paper, authors also apply a Variable Neighborhood Search (VNS) to the TOP. The VNS is considered in two versions depending on the computational time allocated to it. Overall, the VNS was better than both of their TS. A Skewed VNS (SVNS) is also proposed by Vansteenwegen et al. (2009b) and outperforms their prior Guided Local Search (GLS) algorithm (Vansteenwegen et al., 2009a). Vidal et al. (2014b) developed several heuristics based on a new neighborhood search. Among these heuristics were a multi-start local improvement heuristic and an Iterated Local Search (ILS). Both heuristics performed well in terms of solution's quality with respect to the computational time. Kim et al. (2013) proposed the first algorithm based on large neighborhood search. They used an Augmented Large Neighborhood Search (AuLNS) paired with three improvement algorithms and obtained the best known solutions of all instances in a better computational time than Dang et al. (2013a), outperforming all existing algorithms. Recently, a LNS proposed by Orlis et al. (2020) finds all but one best known solutions on the small scale set of instances. In 2020, Hammami et al. (2020) proposed an Hybrid Adaptive Large Neigbhorhood Search (HALNS). In their algorithm, the ALNS framework is paired with the resolution of a set packing problem consisting in finding the best set of routes built during the process. Results showed the efficiency of the method as it found all best known solutions for both the small and large scale sets of instances while improving one best known solution on the last set. This method is the best for both set of instances from the literature with respect to the solution's quality and the associated CPU time. Other methods are proposed as a Multi-start Simulated Annealing (MSA) algorithm (Lin, 2013) that obtained 135 best known

solutions on the small scale instances, and a Similarity Hybrid Harmony Search algorithm (SHHS) (Tsakirakis et al., 2019). In the latest, the authors proposed two versions of their algorithm, one being static while the other is dynamic. The better one, the dynamic, found 276 best known solutions out of the 328 tested instances.

Population-based algorithms have also been used to tackle the TOP. Ke et al. (2008) proposed 4 variants of Ant Colony Optimization algorithm based on the algorithm used to build candidate solutions. Out of the 4 versions, the sequential one was the best one. Bouly et al. (2008) proposed a Memetic Algorithm (MA) which consists in the combination of a Genetic Algorithm (GA) and local search operators. Results obtained were comparable to those of Archetti et al. (2007). Dang et al. (2011) proposed a Particle Swarm Optimization-based Memetic Algorithm (PSOMA) based on the work from Bouly et al. (2008). It showed very good performances as it outperformed existing algorithms. In 2013, Dang et al. (2013a) proposed a PSO-inspired Algorithm (PSOiA) based on their prior work. The proposed algorithm outperformed all existing algorithms: it was the first algorithm to find every best known solutions of the small scale set of instances. Based on their results, the authors showed that a new set of instances was needed to have a new base of research for the TOP, as already proposed in Souffriau et al. (2010). They built a new set of larger instances based on the OP instances constructed by Fischetti et al. (1998). A genetic algorithm approach was then proposed by Ferreira et al. (2014), but they only tested a limited number of small size instances and none of the new large scale instances. A population-based algorithm is also proposed by Vidal et al. (2014b). Specifically, they adapted their Unified Hybrid Genetic Search (UHGS) (Vidal et al., 2014a) framework and obtained very good results on the subset of small scale instances tested. Finally, in 2020, a Scatter Search Hybrid approach was proposed (Alkhazaleh et al., 2019) and showed good results as it got 154 best known solutions out of the 157 tested instances belonging to the small scale set. Other methods includes a Pareto Mimic Algorithm (PMA) (Ke et al., 2016) which showed very good results both on the small and large scale sets, and a Hybrid Scatter Search with steep descent (HISS-SD) approach (Alkhazaleh et al., 2019) which obtained results sligthly worse than PSOiA (Dang et al., 2013a).

Heuristic approaches are described in table 1. In particular, we report, for each method, the algorithm, the set of instance considered, the number of best known solutions found, and the average CPU (s).

3.3. Very large scale

The very large scale component has only been tackled a few number of times in the VRP. In this context, the number of customers is large and exact methods are not suitable. The size of these instances is obviously a key factor to choose or adapt a method to this context. The goal is then to obtain very good solutions in a reasonable computational time based on the instance size and the problem tackled. In Kytöjoki et al. (2007), the authors developped a VNS that was coupled with a GLS framework. Memory usage was optimized by the use of compact information avoiding the storage of the distance matrix. They also used an appropriate representation of a solution that eases classical operations like insertions and removals. The algorithm managed to solve instances up to 20 000 customers in a reasonable time. Zachariadis and Kiranoudis (2010) proposed a TS paired with Static Move Descriptors (SMD), a method that greatly reduces the computational complexity of local search operators so that they cost an almost linear computational time. Also using these SMD to diversify the exploration, they managed to solve instances up to 3 000 customers. Recently, Arnold et al. (2019) proposed a knowledge-based heuristic based on a prior analysis about the difference between (near-)optimal solutions and other solutions. Used to penalize 'bad' edges in a GLS and associated to pruning strategies and reduced information storage, they managed to solve instances up to 30 000 customers. Finally, Accorsi and Vigo (2021) proposed a new method based on an ILS. To reduce the computational time, they used SMD, granular neighborhood (Toth. and Vigo, 2003), and intensify the optimization around areas that were recently modified. The local search was designed

Method	Reference	Algorithm	Instance set	# BKS	CPU (s)
THM	Tang and Miller-Hooks (2005)	Tabu Search	Chao et al. (1996)	33/157	445.7
GTP	Archetti et al. (2007)	Tabu Search	Chao et al. (1996)	69/157	113.2
GTF	Archetti et al. (2007)	Tabu Search	Chao et al. (1996)	94/157	189.22
FVNS	Archetti et al. (2007)	Variable Neighborhood Search	Chao et al. (1996)	94/157	22.65
SVNS	Archetti et al. (2007)	Variable Neighborhood Search	Chao et al. (1996)	127/157	322.59
SACO	Ke et al. (2008)	Ant Colony	Chao et al. (1996)	128/157	294.61
DACO	Ke et al. (2008)	Ant Colony	Chao et al. (1996)	80/157	249.18
RACO	Ke et al. (2008)	Ant Colony	Chao et al. (1996)	81/157	238.77
SiACO	Ke et al. (2008)	Ant Colony	Chao et al. (1996)	84/157	250.58
SkVNS	Vansteenwegen et al. (2009b)	Skewed Variable Neighborhood Search	Chao et al. (1996)	44/157	4.63
GLS	Vansteenwegen et al. (2009b)	Guided Local Search	Chao et al. (1996)	21/157	9.24
FPR	Souffriau et al. (2010)	Path Relinking	Chao et al. (1996)	78/157	6.1
SPR	Souffriau et al. (2010)	Path Relinking	Chao et al. (1996)	126/157	260.61
MA	Bouly et al. (2008)	Memetic Algorithm	Chao et al. (1996)	146/157	114.77
PSOMA	Dang et al. (2011)	Particle Swarm	Chao et al. (1996)	146/157	50.46
AuLNS	Kim et al. (2013)	Augmented Large Neighborhood Search	Chao et al. (1996)	157/157	55.96
PSOiA	Dang et al. (2013a)	Particle Swarm	Chao et al. (1996)	157/157	128.76
			Dang et al. (2013a)	71/82	11,031.04
MSA	Lin (2013)	Multi-start Simulated Annealing	Chao et al. (1996)	133/157	41.34
UHGS	Vidal et al. (2014b)	Unified Hybrid Genetic Search	Chao et al. (1996)	155/157	192
UHGS-f	Vidal et al. (2014b)	Unified Hybrid Genetic Search	Chao et al. (1996)	150/157	68.96
MS-ILS	Vidal et al. (2014b)	Multi-start Iterated Local Search	Chao et al. (1996)	153/157	156
MS-LS	Vidal et al. (2014b)	Multi-start Local Search	Chao et al. (1996)	115/157	9.29
PMA	Ke et al. (2016)	Pareto Mimic Algorithm	Chao et al. (1996)	157/157	66.85
			Dang et al. (2013a)	81/82	1,004.15
SHHS	Tsakirakis et al. (2019)	Similarity Hybrid Harmony Search	Chao et al. (1996)	68/157	74.33
SHHS2	Tsakirakis et al. (2019)	Similarity Hybrid Harmony Search	Chao et al. (1996)	105/157	74.28
LNS	Orlis et al. (2020)	Large Neighborhood Search	Chao et al. (1996)	156/157	140
HALNS	Hammami et al. (2020)	Hybrid Adaptive Large Neighborhood Search	Chao et al. (1996)	157/157	23.8
			Dang et al. (2013a)	82/82	783.36
HISS-SD	Alkhazaleh et al. (2019)	Hybrid Scatter Search	Chao et al. (1996)	154/157	80.95

Table 1: Description of references method used. Based on the instance set, we report the number of best known solutions (BKS) found and the average CPU (s) reported.

in a Hierarchical Randomized Variable Neighborhood Descent (HRVND) to further promote the intensification and the diversification while maintaining efficient computational time. Results were better than previous works on the very large scale VRP.

With regard to the OP, Kobeaga et al. (2017) introduced a new set of very large scale instances. Authors proposed a genetic algorithm that maintains unfeasible solutions during the search. However, no specific mechanisms were introduced to address the very large scale component in their work, resulting in high computational time for the biggest instances of the set.

4. Problem formulation

We model the Multi-Depot Open Team Orienteering Problem with a graph $G = (V, A)$ where $V = \{1, ..., N\}$ represents the set of vertices, *i.e.* the customers, with $\{1, ..., m\}$ and $\{N - m, ..., N\}$ being the starting and ending points of every route, m the number of vehicles, and $A = \{(i, j) \mid i \in V, j \in V, i \neq j\}$ the set of edges. A profit p_i is associated to each customer $i \in V$ and each edge $(i, j) \in A$ has a travel time $t_{i,j}$. Travel times to the ending points have a cost of 0 as these points are introduced to model the open path of each vehicle. The set of vehicles is noted $K = \{1, 2, .., m\}$ and all vehicles share identical features. Every vehicle $k \in K$ must respect a time limit D.

We use two types of variables to model the MDOTOP: (1) binary variables $x^k_{i,j}$, which determine if the vehicle $k \in K$ uses the edge $(i, j) \in A$, and (2) binary variables y^k_i which determine if the vertice $i \in V$ is visited by the vehicle $k \in K$. Using these variables, based on El-Hajj et al. (2016), we can model the MDOTOP as follows:

$$z^\star = \max \sum_{k \in K} \sum_{i \in V} y^k_i \times p_i \tag{1}$$

$$s.t. \sum_{j \in V} x^k_{j,i} = \sum_{j \in V} x^k_{i,j} = y^k_i \qquad \forall k \in K, \ \forall i \in V \setminus \left\{\{1, ..., k\} \cup \{N-k, ..., N\}\right\} \quad (2)$$

$$\sum_{k \in K} y^k_i \leq 1 \qquad \forall i \in V \quad (3)$$

$$\sum_{j \in V} x^k_{k,j} = \sum_{j \in V} x^k_{j,N-k} = 1 \qquad \forall k \in K \quad (4)$$

$$\sum_{i \in V} \sum_{j \in V} x^k_{i,j} \times t_{i,j} \leq D \qquad \forall k \in K \quad (5)$$

$$\sum_{(i,j) \in S \times S} x^k_{i,j} \leq |S| - 1 \qquad \forall S \subseteq V \setminus \{1, N\}, |S| \geq 2, \forall k \in K \quad (6)$$

$$x^k_{i,j} \in \{0, 1\} \qquad \forall (i, j) \in A, \forall k \in k \quad (7)$$

$$y^k_i \in \{0, 1\} \qquad \forall i \in V, \forall k \in K \quad (8)$$

The objective function (1) maximizes the total profit collected on customers. Constraints (2) are the flow conservation constraints and link variables x^k_{ij} and y^k_j. Constraints (3) ensure that each customer can be visited at most once. Constraints (4) impose the starting and ending points for all vehicles. Constraints (5) guarantee that every vehicle will respect the time limit D. Finally, constraints (6) will make sure that there is no subtour in the solution.

To our knowledge, the MDOTOP was not considered in the literature. To compare our method with the literature, it is thus applied to the TOP. As no exact methods are applied to the large scale set of instances for the TOP, and because the best exact method (Orlis et al., 2020) already has important computational time on the small scale one, we propose a metaheuristic to address the problem.

5. Solution approach

In this section, we present the algorithm we have developed to tackle the very large scale TOP. As previously mentioned, the main goal of this paper is to propose an algorithm which quickly provides very good solutions so it can be used in a complex industrial context. Based on the literature, we proposed a metaheuristic which is more suitable to solve difficult and large scale routing problems in a reasonable time. We chose to develop a Large Neighborhood Search (LNS) to address the TOP. This choice is led by its high level of performances when tackling the VRP and its variants while being highly scalable to solve industrial problems (Mara et al., 2022). LNS is first introduced in 1998 (Shaw, 1998) and then extended to the ALNS by Ropke and Pisinger (2006) who applied it on a Pickup and Delivery Problem with Time Windows (PDPTW). LNS is a metaheuristic based on the repetition of destroy and repair operators. The structure of our LNS is detailed in Algorithm 1. In the following, we first present our construction heuristic and then our LNS.

Algorithm 1 Overall LNS framework used in this paper.

Require: Initial solution: S_0, initial temperature: T_0, number of non-improving iterations before increasing the size of neighborhood lists: κ, cooling factor: c, cooling step size: stepSize
$S_{Best} \leftarrow S_0$, $T \leftarrow T_0$, NoImp $\leftarrow 0$
while Non stopping criterion **do**
  for step $\leftarrow 1$ to stepSize **do**
    $S_{New} \leftarrow S_0$
    Select a removal operator D and a repairing operator R
    Generate q and remove q vertices of S_{New} using D
    Repair S_{New} using R
    $\Delta = z(S_{New}) - z(S_0)$
    if S_{New} meets the requirements **then**
      Apply Local_Search on S_{New}
    end if
    Generate $\delta \in (0, 1)$
    if S_{New} is better than S_0 **or** $e^{\frac{-\Delta}{T}} \geq \delta$ **then**
      $S_0 \leftarrow S_{New}$
    end if
    if S_{New} is better than S_{Best} **then**
      $S_{Best} \leftarrow S_{New}$
      Reset the size of the neighborhood list
      NoImp $\leftarrow 0$
    else
      NoImp $\leftarrow$ NoImp + 1
    end if
    if NoImp $\geq \kappa$ **then**
      Increase the size of the neighborhood list
      NoImp $\leftarrow 0$
    end if
  end for
  $T \leftarrow T \times c$
end while

5.1. Construction heuristic

The LNS requires an initial solution S_0 that will be improved over iterations. In routing

problems, several methods exist to build an initial solution, as among the most popular ones, the Cheapest Insertion and the Clarke & Wright heuristic (Clarke and Wright, 1964). As we aim to deal with problems of thousands of points, a cluster-first route-second method is defined to build the initial solution. By partitioning the space, the construction of the initial solution is easier and the overall process might even be sped up by providing a better solution to the metaheuristic. This method has shown success in the TSP context where millions of customers can be tackled by partitioning the space before optimizing each partition independently (Taillard and Helsgaun, 2018).

5.1.1. Clustering-first

To do the clustering, k-means has shown good performances (Mariescu-Istodor, 2021). Based on it and proposed in Kaufman and Rousseeuw (1990), we then chose k-medoids to cluster the customers. In the TOP however, all customers are usually not visited as every route is constrained by a time limit. To take this into account, we propose an adaptation of the algorithm of Bernábe-Loranca et al. (2014). In their method, in order to build balanced clusters, the authors propose to modify the traditional function optimized in k-medoids by a weighted objective function. They apply it in a geographical context and show that using the modified k-medoids function results in better-balanced clusters. This idea of balancing clusters is interesting to represent the time limit associated to each route. It has already be shown that compact routes are usually better than routes overlapping each other and these intersections must be avoided in the VRP (Arnold and Sörensen, 2019). Having compact and non-overlapping routes should then ease the overall process of finding quickly very good solutions while the balance makes sure that each partition is equally distributed. Solutions where routes are compact, contiguous and non-overlapping also tend to be accepted more easily by the user in a realistic routing plan (Bräysy and Hasle, 2014), making the interest of the clustering even bigger for our application. The k-medoids function of Bernábe-Loranca et al. (2014) is defined by:

$$F1 : \min \; w_{medoids}\left(\sum_{k=1}^{K}\sum_{i\in M_k} t_{i,C_k}\right) + w_{homogeneity}\left(\sqrt{\frac{\sum_{k=1}^{K}(|M_k| - B)^2}{N}}\right)$$

where t_{i,C_k} is the travelling cost between the customer $i \in M_k$ and the center C_k of the medoid M_k, $|M_k|$ the number of customers in the k^{th} medoid and B the balance objective equal to $\frac{N}{m}$. The first part is the classical k-medoids objective function to minimize. The second part is associated to Loranca's homogeneity function: it consists in comparing the number of customers belonging to the cluster and its balance objective to penalize the function according to this difference. An adaptation of this function is proposed to better represent the routing context. Indeed, k-medoids may associate a vehicle to a cluster which is far away from the vehicle's depot. This would induce a non-negligible cost in the vehicle's route which would not have been anticipated during the clustering process. To solve this problem, we introduce a new component to the k-medoids function proposed in Bernábe-Loranca et al. (2014):

$$F2 : \min \; w_{medoids}\left(\sum_{k=1}^{K}\sum_{i\in M_k} t_{i,C_k}\right) + w_{homogeneity}\left(\sqrt{\frac{\sum_{k=1}^{K}(|M_k| - B)^2}{N}}\right) + w_{depot}\left(\sum_{k=1}^{K}\min_{i\in M_k} t_{k,i}\right)$$

The first two parts are Bernábe-Loranca et al. (2014) function while the last part is there to make sure that the clustering function takes into account the cost needed for a vehicle to reach its cluster. Note that in the TOP context all vehicles share a common depot but the function can easily be tuned to tackle the multi-depot variant. It is also possible to consider an unbalanced version,

for example if the fleet of vehicles is heterogeneous, to build clusters of different size based on the vehicle speed.

5.1.2. Routing-second

When the clustering phase is finished, each customer is inserted into its associated route (defined by its cluster) using the Cheapest Insertion heuristic while an insertion is feasible, that is an insertion which does not violate the time limit D of the route. Finally, the local search framework, detailed in Section 5.3, is applied to improve the quality of the initial solution.

5.2. Large Neighborhood Search

To optimize our initial solution, we chose to use a LNS. As it is often done, our LNS is paired with simulated annealing (SA) to escape local optimum by accepting slightly worse solutions (Kirkpatrick et al., 1983). Our LNS also has classical stopping criteria based on the computational time, the number of iterations and the number of iterations without improvement. Specific details are discussed next: first, we start to describe both destroy and repair operators considered in this paper. Then, we describe neighborhood lists that are used to reduce the number of evaluations done in our optimisation process. Afterwards, we present the blink that is done to diversify the neighborhood exploration. Finally, the objective function used to compare solutions is presented.

5.2.1. Destroy operators

Our LNS uses several destroy operators to explore a large neighborhood. The number of destroyed customers q will be picked from an interval $[q_{min}, q_{max}]$ for operators which destroy customers, and from an interval $[q_{seq_{min}}, q_{seq_{max}}]$ for operators which destroy a sequence of consecutive customers. At each iteration, the selected operator is chosen randomly to avoid redundancy. In addition, a variant based on sequences of consecutive customers is considered for most operators. Classical operators proposed by the literature and used in our algorithm are:

- Worst Cost (Sequence) destroy: criterion associated to the cost of a (sequence of) customer in the route;
- Worst Profit (Sequence) destroy: criterion associated to the profit of a (sequence of) customer;
- Random (Sequence) destroy: destroyed (sequence of) customers are chosen randomly.
- Shaw destroy: Shaw's criterion was first introduced in Shaw (1998). It consists in destroying relative customers from the route. In our work, we consider two customers i and j as relatives if the travelling time $t_{i,j}$ between them is small enough.

Dedicated operators used are:

- Rectangle destroy: geographical position is a criterion that may be useful when tackling specific instances where some areas are widely covered by customers. Introduced in Demir et al. (2012), this operator picks randomly a seed customer from the problem and deletes every customer which is in a rectangle around him (figure 1). While the number $q \in [q_{min}, q_{max}]$ of customers to delete is not reached, the rectangle is extended.
- Worst Cost-Efficiency (Sequence) destroy: the profit is a good indicator of the relevance of a customer. However, a customer can bring a small profit while not being costly in travelling time. We then define an operator based on the cost-effectivness of a customer which deletes from the solution the $q \in [q_{min}, q_{max}]$ worse customers based on this criterion. The cost-efficiency of a customer i positioned between customers j and l, noted CE_i, is computed as follows:

$$CE_i = \frac{p_i}{c^i_{j,l}} \tag{9}$$

$$c^i_{j,l} = t_{j,i} + t_{i,l} - t_{j,l} \tag{10}$$

where p_i is the customer's profit, and $c^i_{j,l}$ its cost of visit in its route as defined in Equation 8. A cost-efficient customer is then very interesting to visit because he doesn't cost much compared to the profit he brings. This operator makes a link between the objective function, which maximizes the total collected profit, and the time limit constraint associated to the routes.

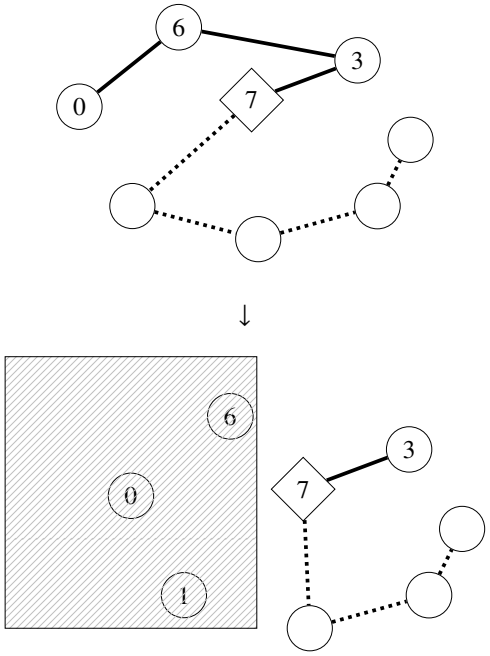


Figure 1: Figure showing the principle of Rectangle Destroy. The customer (0) is picked as the seed, thus is the center of the rectangle. Every customer in the resulting rectangle is destroyed from the solution.

5.2.2. Repair operators

Our LNS is composed of seven repair operators. We chose these operators based on their goal to avoid overlapping operators while exploring a large and diverse neighborhood. As we already mentioned, unlike in the VRP, in the TOP it is not possible to insert every customer into a vehicle's route. Those operators aim to insert the best customers possible while maintaining the feasibility of the route. To do so, after each insertion, we check that it is still possible to insert a customer. The reparation process stops when the operator can't insert a new customer. For all these operators, customers are inserted into their best feasible position, that is the position of insertion inducing the smallest increase of cost among the route. Our repair operators can be categorized in two families: global insertion operators, which is composed of operators that compare every customer at each insertion, and sorted insertion operators, which insert customers in a pre-computed order.

Global insertion operators. Global insertion operators are more costly than sorted insertion operators but they also have a better view of the solution at each step of insertion. Most of the time, they remain fast to compute but their computational time also grows with the instance size. This may be problematic in our context as we want to solve very large scale instances. To avoid this problem, they tend to be used with a smaller probability than other insertion operators. Two global insertion operators are evaluated in our LNS:

- Best Cost insertion: criterion associated to the cost related to the insertion of a customer into a route;
- Cost-Efficiency insertion: criterion associated to the cost-efficiency of a customer into a route. As the customer is not yet visited and because customers are only inserted into their best feasible position of insertion, this criterion can be computed as:

$$CE_i = \frac{p_i}{\min\limits_{\forall j,l} c^i_{j,l}} \tag{11}$$

  where $\min c^i_{j,l}$ corresponds to the minimum feasible cost insertion of i in the solution.

Sorted insertion operators. Sorted insertion operators are fast to compute as they usually start by sorting the customers based on a specific criterion and then insert them in the defined order. In opposition with global insertion operators, they have a very limited view of the solution at each step of insertion as the defined order is not reassessed after any insertion. Repair operators that use this principle are the following:

- Best Profit insertion: criterion associated to the profit of a customer;
- Random insertion: select randomly the customer to insert into a route;
- Order insertion: replace the customers that just got deleted from the solution. It aims at moving well-organized sequences of consecutive customers into a better place. Indeed, if a sequence of customers is well-organized, its cost might be low if globally taken. Nevertheless, this does not mean that

the sequence is at the best place in the solution. The Order insertion operator aims at correcting these kinds of issues.

- Reverse Order insertion: inserts customers in the reverse order of destruction. This means that the last customer deleted from the solution is the first to be replaced. The operator aims at reversing sequences of consecutive customers which might be well-organized but might be better placed if reversed.

- k-Regret Cost-Efficiency insertion: based on the classical regret insertion operator, this operator searches for customers that are the most critical in the insertion order. The k-regret associated to the cost-efficiency criterion of a customer i can be computed as:

$$\sum_{j=1}^{k} CE_i^0 - CE_i^j \tag{12}$$

where CE_i^j is the j^{th} best cost-efficiency of i. The k-Regret Cost-efficiency operator has a similar objective to the Cost-efficiency operator: to find the best customers to insert according to both the cost associated to their visit and the profit they bring. However, it does it in a smaller computational time, as customers are sorted before inserting them. In this paper, a full regret is used to compute the regret value, meaning that all feasible insertions are considered for each customer in the computation of the regret cost-efficiency value.

5.2.3. Neighborhood lists

To avoid the evaluation of too much possibilities during the process, neighborhood lists are used. We use similar ones as proposed by Toth. and Vigo (2003), which are called granular lists. These lists aim at defining a set of neighbors for every customer based on a set of 'short' edges. In our algorithm, a customer i is a neighbor of a customer j if it follows one of the following conditions:

- $t_{i,j} < \vartheta = \beta \times \frac{c_{S_0}}{\tau + m}$;
- j is at most the η^{th} nearest customers from i.

where ϑ is the granularity threshold, β is a positive sparsification parameter, c_{S_0} is the cost value of the initial heuristic solution, *i.e.* the sum of every route's cost, τ is the number of visited customers in the solution, and η is the minimum number of neighbors considered for each customer. The first condition enables the algorithm to search around 'short' arcs while the second one makes sure that every customer has a minimum number of neighbors, tackling some cases where a customer might be isolated from the others. As the initial granularity threshold value might not be the best suited for the tackled problem, it may be interesting to increase the size of our lists. For this purpose, we also define a parameter κ which corresponds to the maximum number of LNS iterations without finding a new best solution. If during the LNS process we reach this number, then the size of our lists is increased by increasing the value of β. A maximum number of neighbors per customer is also defined to avoid considering too much neighbors. These lists are used in every repair operator and in every local search operator. For example, while repairing a solution, only the insertion of a customer around its neighbors is evaluated. Similarly, only moves that imply neighbors are evaluated in the local search framework. Thus, the size of the graph considered is reduced as illustrated figure 2.

5.2.4. Blink

A blink is used to add a diversification process in our destroy & repair operators. Blink is introduced in Christiaens and Vanden Berghe (2020) and consists in skipping an evaluation or a movement based on a random criterion. Several types of blink were tested: blinking when finding a new best candidate in destroy & global repair operators, blinking to avoid the evaluation of a customer for destruction (Dumez et al., 2021), blinking to avoid the evaluation of a customer insertion, and blinking to avoid a position of insertion for a customer. Overall, only the first one has improved the algorithm performance, meaning that sometimes the algorithm ignores a new best candidate to remove or insert in the solution.

5.2.5. Objective function

Obviously, the main objective function to optimize is the total profit collected on customers.

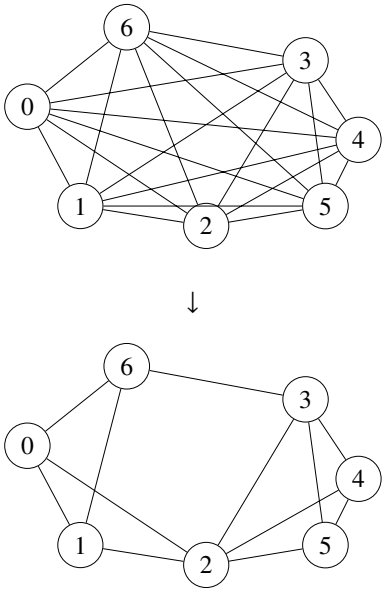


Figure 2: Figure showing an example of the reduction in the graph size while using neighborhood lists. Edges which are more costly than a defined threshold value are not explicitly considered anymore.

However, it is important not to overlook the cost of the solution associated to the collected profit. In the TOP, several solutions may have the same profit but differ in its routes cost. A smaller-cost route should be preferable over a bigger-cost route as it should be easier to add a new customer into the smallest one. This idea is used by Vidal et al. (2014b) and has already shown its interest. As such, solutions having a smaller cost are prioritized when comparing to solutions having the same profit.

5.3. Local search

The main goal of this local search framework is to quickly get to, or near, the local optimum. By pairing this framework and the destroy & repair process, we hope to cover a very wide number of local optima and, by extension, to find the global optimum. The local search framework is composed of several operators which are applied in the given order while there is an improvement: 1-1 Exchange, Replace, 3-opt, and Cross-Exchange. This order is chosen based on the computational complexity of local search operators. For all but 3-opt, the best-improvement criterion is used to determine the move to apply. For 3-opt, a first-improvement strategy is used. If the local search framework improves the solution, it tries to insert new customers using the Cost-efficient repair operator as a compromise between the additional profit gained by visiting a new customer and the remaining time available in the associated route. This process is repeated while it improves the quality of the solution.

5.3.1. Restricted candidate list

Following the same principle than the neighborhood lists, we define a Restricted Candidate List (RCL). Our choice of using it is based on Arnold et al. (2019) where authors show that, in very large scale problems, it is important not to lose time in the evaluation of already good area, while intensifying the optimisation around bad areas will lead to an overall better solution. It is then necessary to define what is a good area and a bad one in a solution. This definition might be problem dependant. In their work, Arnold and Sörensen (2017) study how different are (near-) optimal solutions from non-optimal solutions based on several metrics on a VRP problem. Using these metrics, they managed to predict the quality of a solution with an accuracy up to 93%, which shows that these metrics can be used to differentiate these solutions. They finally use the width, the cost and the depth implied by an edge to define a 'bad' edge, and use this definition to build a very competitive heuristic for the VRP. In the TOP context, profit is at least as important as the cost of a customer and these observations might not hold. However, the cost of a route is still a key factor while optimizing a solution since it might be possible to insert another customer by reducing it. In our work, we characterize customers and not edges as profit is customer-dependent. A customer i positioned between customers j and l is then evaluated by his profit p_i, his cost $c^i_{j,l}$ (see Equation 10), his depth $depth_i$ and his width $width^i_{j,l}$ that can be computed as:

$$width^i_{j,l} = width(j,i) + width(i,l) \tag{13}$$

Both the depth of a customer and its width are represented in figure 3. To compute the width $width(i,j)$, first build the parallel straight line

from $(0, G)$ going through i, with G the route's center of gravity. $width(i, j)$ is then computed as the distance between j and its projection on the previously built straight line. The resulting function used to evaluate a customer is then:

$$V_i = \left(w_p \times p_i + w_w \times width^i_{j,l} + w_c \times c^i_{j,l}\right) \times \left(\frac{depth_i}{maxDepth}\right)^{\frac{w_d}{2}} \tag{14}$$

where $maxDepth$ is the maximum depth in the route, and w_p, w_w, w_c and $w_d \in \{0, 1\}$ are the weight applied to each metrics considered. The RCL is only used in the 1-1 Exchange, Best Replace and Cross-exchange operators, to limit the number of evaluated customers while targeting the 'bad' ones. It is composed of the φ worst customers of the solution based on Equation 14.

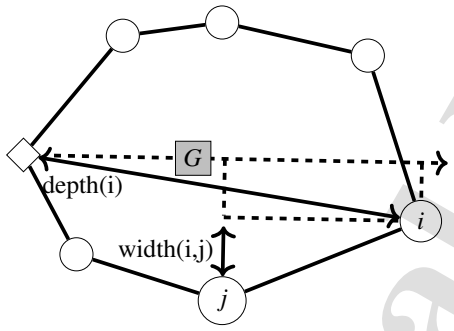


Figure 3: Illustration of the metrics used to characterize the restricted candidate list. (G) is the route's center of gravity. The depth of (i) corresponds to the distance between the depot and i.

6. Numerical results on literature instances

Our algorithm is implemented in C++11. We evaluate it on an Intel Xeon processor E5-2690, 2.60GHZ with 8 GB of RAM. The LNS is tested in the same conditions as in Hammami et al. (2020): we report the best solutions obtained from twenty independent runs which are performed on each instance. Tests are performed on two sets of existing instances, and on a new set of very large scale instances that we built for the TOP based on the instances of Kobeaga et al. (2017) for the OP. Instances are described in Section 6.1.

6.1. Instances description

To evaluate our algorithm, we apply it to popular TOP instances from the literature and on a newly introduced set of very large scale instances. These instances are classified into three different categories which are small scale, large scale and very large scale, based on their size.

6.1.1. Small scale instances

The small scale instances are those introduced in Chao et al. (1996). These instances are separated into seven sets based on their size, which vary from 21 to 102 customers. In each set, the number of customers and their position stay the same, but the time limit and the number of vehicles vary. According to Souffriau et al. (2010), out of the 387 instances of this set, only 157 are relevant as every algorithm got the same results on the remaining instances of this set. Therefore, only these 157 instances are considered in this paper.

6.1.2. Large scale instances

We also test our LNS on a larger set of instances. This set is the one reported by Dang et al. (2013a). It was introduced to give a new area of research on the TOP as PSOiA (Dang et al., 2013a) found all best known solutions when applied to the small scale set of instances. This set is composed of 333 instances which were adapted from the OP (Fischetti et al., 1998) and derived from the Capacitated VRP (CVRP) and the TSP. For the CVRP, customers'demand became their profit, whereas for the TSP instances, profits were created using three ways:

- Every customer has a profit of 1 (gen1);
- Every customer's profit is computed by a pseudo-random function so that the value can't be less than 1 and more than 100 (gen2);
- Every customer has a profit dependent on its distance from the depots. The further the customer, the bigger the profit (gen3).

These instances are larger than the previous set, ranging from 101 to 400 customers. Still, they are not big enough to model some industrial problems which can have several thousands of customers to consider.

6.1.3. Very large scale instances

The main goal of our algorithm is to tackle instances of several thousands of customers. To our knowledge, such instances are not available in the literature in the TOP context. A new set of instances based on the literature (Kobeaga et al., 2017) is built by adapting the very large scale instances from the OP. The time limit D^{TOP} of the TOP is based on the time limit D^{OP} of the OP and determined as in Chao et al. (1996): $D^{TOP} = \frac{D^{OP}}{m}$, where m is the number of vehicles considered. Instances can be found at `https://github.com/CharlyChgn/VLS-Team-Orienteering-Problem`. It consists in 24 problems associated to 3 classes of profit generation (gen1, gen2 and gen3) as in Dang et al. (2013a) (and explained previously for large instances) and with 4 variations on the number of vehicles, resulting in a total of 288 instances. As these instances are bigger, it is easier to consider more vehicles while still having a large number of customers per route. This is consistent with our goal to tackle industrial problems as the number of vehicles can be bigger in realistic problems. In this set of instances, the number of vehicles varies between 2, 3, 8 and 10 vehicles while the number of customers ranges from 1001 to 5395.

6.2. Parameters

As previously mentioned, our LNS is composed of numerous parameters. A summary of their notations and definitions can be found in table 2. To evaluate our algorithm, these parameters values need to be fixed. As our main goal is to show that our algorithm is competitive to tackle big instances, a subset of 14 instances from the large and the newly introduced very large scale set are used. These instances are selected to cover different types of instances based on the depot position, and all kinds of profit generation are considered. They range from 195 to 3795 customers. The algorithm is applied 10 times on each selected instance with different parameters values picked from the literature. Parameters'values are fixed one by one based on their impact on the solution's quality and the computational time. Final results are also summarized in table 2. A similar task is done to evaluate the interest of our destroy and repair operators. Table 3 sums the remaining operators used in the final algorithm. It is noteworthy that, even if random operators manage to bring slightly better solutions, we chose not to use them as they have low chances to be of interest when applied to the very large scale set of instances.

According to the results, local search is applied differently based on the set of instances tackled: it is only applied when a new best solution is found for the small and large scale set of instances, while it is applied each time the obtained solution is better than the solution before the destroy & repair process for the very large scale set. To ensure consistency in computational time, the stopping criterion is 250 000 iterations or 100 000 iterations without improvement for the small and large scale set, and 100 000 iterations for the very large scale set. This high number of iterations without improvement is explained by the time needed to extend the neighborhood lists which enables a larger exploration. The maximal computational time allowed is one hour but we aim to obtain very good solutions in a budget time of 10 minutes.

6.2.1. Clustering

The proposed clustering is based on three terms associated to the compactness (corresponding to the classical k-medoids function), the balance and the cost for a vehicle to reach its associated cluster. Figure 4 shows two examples of differences between the clustering method proposed by Bernábe-Loranca et al. (2014) and our adapted method. As can be seen, vehicles are closer to their depot using our method than the clustering function used by Bernábe-Loranca et al. (2014). To have a better view of the impact of each part of the proposed function, we apply our clustering algorithm to the large scale set of instances. We ignore instances that only differs in the profit generation as the clustering function would then yield the same clusters. Results reported in table 4 show the average value of each part and for different weights. These values need to be the small-

Parameters		Description	Value
Clustering	$w_{medoids}$	Weight of the medoids cost in the k-medoids function	0.2
	$w_{homogeneity}$	Weight of the balance cost in the k-medoids function	0.7
	w_{depot}	Weight of the depot-to-cluster cost in the k-medoids function	0.1
Destroy operators	$[q_{seq_{min}}, q_{seq_{max}}]$	Number of customers removed by destroy operators (%)	[1,20]
	$[q_{seq_{min}}, q_{seq_{max}}]$	Number of customer removed by sequential destroy operators (%)	[1,20]
SA framework	δ	Degradation used to compute the initial temperature (%)	1
	c	Cooling factor	0.99975
	$stepSize$	Cooling step size	50
Stopping criterion	CPU_{max}	Maximum computational time (s)	3600
		Maximum number of iterations for small scale and large instances	250 000
	$itermax$	Maximum number of iterations for very large scale instances	100 000
	$NoImp_{max}$	Maximum number of iterations without improvement	100 000
Neighborhood lists	β_0	Initial sparsification parameter	1
	β_{max}	Maximum sparsification parameter	15
	β_+	Value of increment of β	0.5
	η	Minimum number of neighbors per customer	10
	γ	Maximum number of neighbors per customer	25
	κ	Number of non-improving iteration before increasing the size of neighborhood lists	1000
Local search	φ	Number of customers considered in local search (%)	5
	w_p	Weight applied to the profit value used to evaluate a customer	1
	w_c	Weight applied to the cost value used to evaluate a customer	1
	w_w	Weight applied to the width value used to evaluate a customer	1
	w_d	Weight applied to the depth value used to evaluate a customer	1

Table 2: Parameters description and values after the calibration phase.

Type of operator	Operator	State	Score
Destroy Operator	Worst Cost Destroy	✓	1
	Worst Cost Sequence Destroy	✓	1
	Random Destroy	×	–
	Random Sequence Destroy	✓	1
	Shaw Destroy	✓	3
	Rectangle Destroy	✓	1
	Worst Profit Destroy	✓	2
	Worst Profit Sequence Destroy	×	–
	Worst Cost-efficiency Destroy	✓	2
	Worst Cost-efficiency Sequence Destroy	✓	1
Repair Operator	Best Cost Insertion	×	–
	Random Insertion	×	–
	Order Insertion	✓	4
	Reverse Order Insertion	✓	4
	Best Profit Insertion	✓	4
	Best Cost-efficiency Insertion	✓	1
	Regret Cost-efficiency Insertion	✓	4

Table 3: Results for the operator selection.

est possible. Overall, it shows that each component add a control over the associated criterion, thus confirming the relevance of the latter one in a routing context. The impact of the weights on the final solution is evaluated in table 5 based on the relative percentage error (RPE), the average relative percentage error (ARPE) and the computational time. RPE and ARPE are computed as:

$$\text{RPE} = \frac{z(S_{BKS}) - z(S_{Best})}{z(S_{BKS})} \tag{15}$$

$$\text{ARPE} = \frac{z(S_{BKS}) - z(S_{mean})}{z(S_{BKS})} \tag{16}$$

with $z(S_{BKS})$ the objective function of the best known solution, $z(S_{Best})$ the profit of the best found solution, and $z(S_{mean})$ the average profit obtained along the whole set of executions. Note

that, for the very large scale instances of the subset of calibration, $z(S_{BKS})$ values are computed using the best value obtained while modifying the parameter's value.

Results show how each term may change the quality of the average solution found and the associated computational time. In particular, considering all three terms of the function gives slightly better solutions in average while also being faster to compute. Final weights used in this paper can be found in table 2.

Weights	C	B	DtC
(1,0,0)	713.16	3.22	514.98
(0,1,0)	1007.38	0.47	568.60
(0,0,1)	1063.48	4.88	236.63
(0.3,0.7,0)	807.85	0.75	662.38
(0.2,0.7,0.1)	870.84	0.55	656.62

Table 4: Summary results of the clustering function when applied on the large scale set of instances with different weights ($w_{medoids}, w_{homogeneity}, w_{depot}$). Compactness (**C**) is associated to the classical k-medoids function, balance (**B**) to the added part of Bernábe-Loranca et al. (2014), and depot-to-cluster (**DtC**) to the cost for a vehicle to reach its associated cluster. Values are computed for each clusters independently and then averaged. Values reported correspond to the average across the whole set of instances. For each criterion, we aim for the smallest value possible.

Parameters	RPE (%)	ARPE (%)	CPU (s)
$w_{Cl} = (1, 0, 0)$	1.09	2.98	344.99
$w_{Cl} = (0, 1, 0)$	0.88	1.88	306.86
$w_{Cl} = (0, 0, 1)$	0.88	1.92	296.55
$w_{Cl} = (0.3, 0.7, 0)$	0.50	2.00	302.36
$w_{Cl} = (0.2, 0.7, 0.1)$	**0.17**	**1.78**	**293.62**

Table 5: Impact of the weight used in the clustering function on the calibration subset, with $w_{Cl} = (w_{medoids}, w_{homogeneity}, w_{depot})$.

6.2.2. Adaptive layer

LNS can be extended to the ALNS by adding an Adaptive layer. In the ALNS framework, operators are attributed a score based on their past performance in the algorithm. Although this might be interesting to prioritize specific operators which might be different based on the instance tackled, a recent study (Turkeš et al., 2021) showed that the added value was quite small. Nevertheless, during the calibration phase, we also tried using the adaptive layer. Results showed that the adaptive layer is not useful for our algorithm as better results were obtained without it (table 6). As LNS can have better results by fixing operators'score (Dumez et al., 2021), several values were tested for the best operators according to both their performance and their computational time. Final score for each operator can be found in table 3.

Parameters	RPE (%)	ARPE (%)	CPU (s)
LNS	**0.25**	**1.87**	**293.62**
$\sigma = (33, 9, 13)$	0.98	2.87	525.09
$\sigma = (33, 20, 13)$	0.49	2.85	511.75
$\sigma = (1, 1, 1)$	0.98	2.58	491.66

Table 6: Impact of the adaptive layer on the calibration subset.

6.2.3. Blink

The blink process is evaluated. In our algorithm, it consists in skipping a new best candidate for both deletion and insertion in our destroy & repair process. Results on the calibration subset are summarized in table 7. Note how this process is, in average, beneficial as it improves the quality of the algorithm (+0.12%) by adding more diversification while also decreasing the computational time needed to compute the solution (-20s).

Parameters	RPE (%)	ARPE (%)	CPU (s)
Blink = 0.0	0.68	1.96	370.77
Blink = 0.1	**0.22**	**1.84**	**349.90**
Blink = 0.3	0.65	1.98	385.18

Table 7: Impact of the blink process on the calibration subset.

6.2.4. Restricted candidate list

The RCL is also evaluated in table 8 where its use and different cost function are considered. In particular, we compare the classical function used to evaluate a customer and only based on its cost, the function introduced by Arnold et al. (2019) which adds relevant metrics, and ours which adapts the latest to the TOP context. Results show that the RCL is beneficial for the algorithm, and that our adapted function also bring a better gain to the process. RCL reduces the computational time of more than 600 seconds while

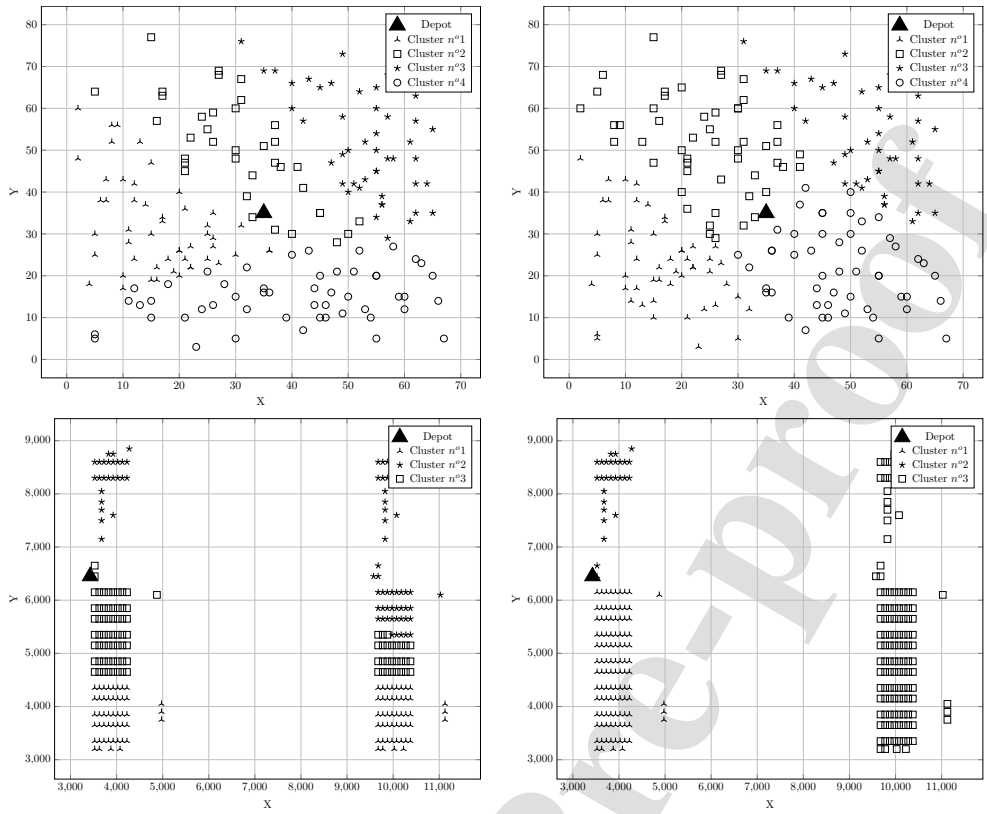


Figure 4: Examples of differences between Loranca's clustering (right) and the adapted function used in this paper (left).

improving the quality of obtained solutions of 0.31% in average. Note that for the biggest instances (3795 customers), when the RCL is not used, the stopping criterion is the maximum computational time (3600s). Results also confirm that Arnold et al. (2019) cost function is better than the classical cost used to evaluate a customer (+0.4%), while the addition of the profit further increases the solution's quality (+0.25%).

6.3. Benchmark

6.3.1. Small scale instances

We first test our algorithm on the small scale set of instances. Our method, named LNS2, is compared with several algorithms (described in table 1) based on their results: a throughout comparison can be seen in figures 5 and 6. The best solutions obtained for each set of instances are reported in tables 17 to 20. On these instances, our algorithm finds 152 out of the 157 best known solutions (BKS) in a smaller computational time than methods obtaining similar results. Our algorithm is indeed very good on this criterion with respect to the number of best known solutions found, outperforming existing literature. The relative percentage error (RPE) is then computed and compared with existing literature. Results are summarized in tables 9 to 11 and also show that our algorithm is competitive to quickly get very good solutions. Figures 5 and 6 show that our algorithm (LNS2) belongs to the pareto front on both the number of BKS found and the RPE, depending on the computing time. In fact, our algorithm, the SkVNS (Vansteenwegen et al., 2009b) and the HALNS (Hammami et al., 2020) compose the pareto front on both criteria as at least one of these methods dominates the others. Overall, results are particularly good as the algorithm parameters were not trained on this set of instances.

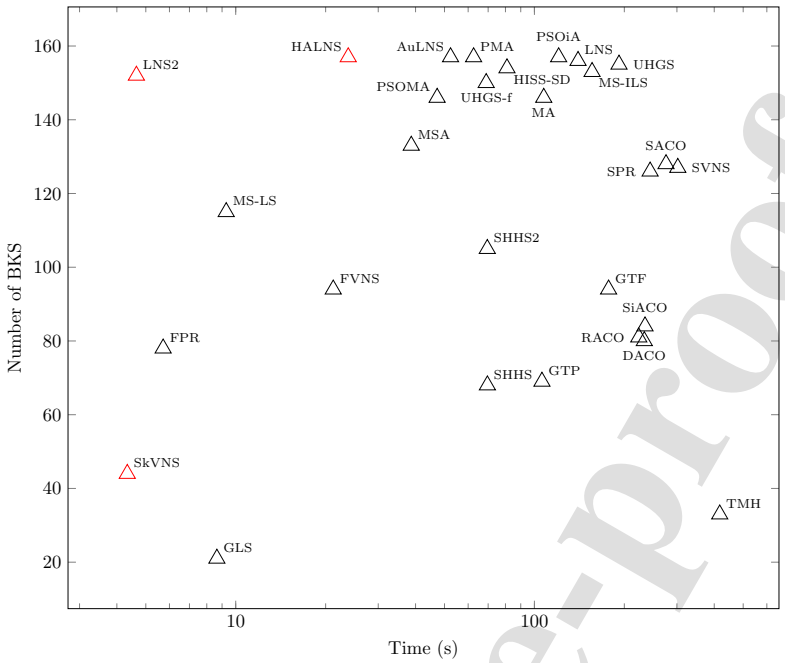


Figure 5: Number of BKS found based on the average computational time and the algorithm used for the small scale set of instances.

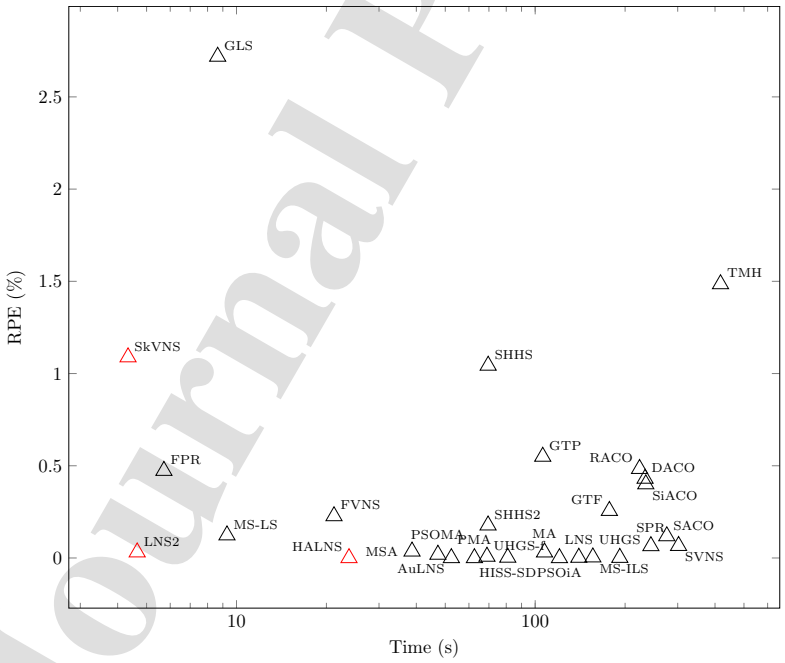


Figure 6: RPE based on the average computational time and the algorithm used for the small scale set of instances.

Parameters	RPE (%)	ARPE (%)	CPU (s)
$\varphi = 100$	0.78	2.15	1045.88
$w_{LS} = (0, 1, 0, 0)$	0.51	2.49	326.57
$w_{LS} = (0, 1, 1, 1)$	0.44	2.09	379.52
$w_{LS} = (1, 1, 1, 1)$	**0.22**	**1.84**	**349.90**

Table 8: Impact of the Restricted Candidate List framework on the calibration subset. We compare the use of restricted candidate list with different values for each weight $w_{LS} = (w_p, w_c, w_w, w_d)$. In particular, we compare the algorithm without RCL ($\varphi = 100$), the use of RCL only considering the cost of a customer in the route ($w_{LS} = (0, 1, 0, 0)$), the RCL with the function of Arnold et al. (2019) ($w_{LS} = (0, 1, 1, 1)$) and with ours ($w_{LS} = (1, 1, 1, 1)$). When considering a RCL, value of φ is 5.

Instances	SkVNS	GLS	FPR	MA	AuLNS	PSOiA	UHGS	MS-ILS	MS-LS	PMA	LNS	HALNS	HISS-SD	**LNS2**
4	7.4	11.4	8.6	182.36	77.3	218.58	236.35	202.68	15.9	109.3	218.02	32.24	80.39	**5.16**
5	**1.5**	3.5	2.9	35.33	22.1	49.5	138.02	89.34	3.36	22.9	66.39	11.63	50.24	3.72
6	**1.9**	4.3	2.1	39.07	12.3	47.08	91.02	56.35	1.97	36.4	42.79	9.67	62.02	3.65
7	**4.3**	12.1	6.3	112.75	66.8	97.47	228.01	201.87	9.76	54.6	152.95	30.89	120.39	5.34
Avgset	**3.78**	7.83	4.98	92.38	44.63	103.16	173.35	137.56	7.75	55.8	120.04	21.1	78.26	4.47
Avg	**4.63**	9.24	6.1	114.77	55.96	128.76	192	156	9.29	66.85	140	23.8	80.95	4.65

Table 9: Average CPU time (s) based on the small scale data set considered.

Instances	**BKS**	SkVNS	GLS	FPR	MA	AuLNS	PSOiA	UHGS	MS-ILS	MS-LS	PMA	LNS	HALNS	HISS-SD	**LNS2**
4	54	1.470	2.967	0.725	0.029	**0.000**	**0.000**	0.004	0.012	0.190	**0.000**	**0.000**	**0.000**	0.001	0.034
5	45	0.622	2.515	0.230	0.062	**0.000**	**0.000**	**0.000**	**0.000**	0.057	**0.000**	0.007	**0.000**	**0.000**	**0.000**
6	15	0.520	1.776	0.112	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**	0.032	**0.000**	**0.000**	**0.000**	**0.000**	**0.000**
7	43	1.309	3.079	0.540	0.013	**0.000**	**0.000**	**0.000**	**0.000**	0.142	**0.000**	**0.000**	**0.000**	0.007	0.071
AVG set	157	0.980	2.584	0.402	0.026	**0.000**	**0.000**	0.001	0.003	0.105	**0.000**	0.002	**0.000**	0.002	0.026
AVG all	157	1.089	2.719	0.473	0.031	**0.000**	**0.000**	0.001	0.004	0.123	**0.000**	0.002	**0.000**	0.002	0.031

Table 10: Average RPE to the best known solution based on the small scale data set considered.

Instances	**BKS**	SkVNS	GLS	FPR	MA	AuLNS	PSOiA	UHGS	MS-ILS	MS-LS	PMA	LNS	HALNS	HISS-SD	**LNS2**
4	54	7	6	17	49	**54**	**54**	52	50	30	**54**	**54**	**54**	53	52
5	45	21	9	33	40	**45**	**45**	**45**	**45**	40	**45**	44	**45**	**45**	**45**
6	15	10	4	12	**15**	**15**	**15**	**15**	**15**	14	**15**	**15**	**15**	**15**	**15**
7	43	6	2	16	42	**43**	**43**	**43**	**43**	31	**43**	**43**	**43**	41	40
All	157	44	21	78	146	157	157	155	153	115	157	156	157	154	152

Table 11: Number of best known solutions found for each set of the small scale instances.

6.3.2. Large scale instances

The algorithm is then tested on the set of large scale instances. Results can be seen in table 21 where they are compared with PSOiA (Dang et al., 2013a), PAM (Ke et al., 2016) and HALNS (Hammami et al., 2020), these algorithms being the only ones applied on this set. Our algorithm finds 54 out of 82 best known solutions among which three new best known solutions for gr229_gen2_m4, gr229_gen3_m4 and rd400_gen1_m2. A detailed description of these solutions, i.e. ordered sets of customers visited by each vehicle, is available at `https://github.com/CharlyChgn/VLS-Team-Orienteering-Problem`. A checker of solution is also joined to verify the integrity of constraints, such as tour duration. In average, it finds solutions which are most of the time very close to the best known solutions while doing so very quickly as shown by its RPE value and its computational time (table 12). When finding the best known solution, our algorithm is clearly faster than any of the other methods. Out of the 4 compared methods, results for the ARPE value are slightly worse but stay very good as, in average, solutions are less than 0.7% worse than the best known solution of the problem. In average, for a computational time 40 times lower, results are only 0.4% worse when compared to the HALNS. Overall, for the large scale set of instances, based on preferences, both HALNS and our LNS are good options as seen

in figure 7, while PSOiA and PMA are both dominated by HALNS. Nevertheless, we believe that our algorithm is more suitable to be used in industrial context as it finds these solutions very quickly at a very low cost of solutions's quality.

6.3.3. Very large scale instances

Finally, we apply the algorithm to the new set of instances ranging from 1001 to 5394 customers, and from 2 to 10 vehicles. As the set is newly introduced to the TOP context, we can only judge the performance of the algorithm based on two values: the average deviation and the time needed to compute the solution. Table 22 reports for each instance the best solution obtained, the CPU time (s) associated, the average solution, the average CPU time (s) and the ARPE value of the algorithm on the instance. Out of the 288 built instances, in three instances the total profit collected is 0, thus are not considered in the results. Overall, the algorithm remains quite fast to compute since for most of the instances it ends in less than one hour, the average computational time being even smaller than the actual processing time of HALNS on the large scale set while having no less than 9.3 times more customers in average. As can be seen in table 13, the ARPE value is higher than in the prior set of instances (1.52%) but is still relevant for such big instances.

6.4. Results analysis based on the number of vehicles

The impact of the number of vehicles is also studied and results are summarized in table 14. Results are interesting as in both the small and large scale sets, the worse results are obtained when the number of vehicles is $m = 3$ while the best results are obtained with the highest number of vehicles $m = 4$. For the newly introduced set of instances, worse results are also obtained when $m = 3$ while results are similar for instances with a higher number of vehicles. We can also note the impact of the number of customers per route on the computational time as the average CPU is higher when considering a smaller number of vehicles. This shows that longer routes take more time to optimize using our algorithm.

6.5. Results analysis based on profit generation

We also study the impact of the profit generation on the results obtained by the algorithm. Results are summarized in table 15. For the large scale set of instances, it shows that the algorithm is in average better on the generation 2 instances corresponding to the random distributed profit on every criterion. When applied to generation 1 instances, where every customer has the same profit of 1, the algorithm performs better than on generation 3, where every customer has a profit depending on its distance from the depot. As both of these types of instances are particular cases of the generation 2, if given more freedom (for example in our neighborhood lists or our restricted candidate list), the algorithm might be able to obtain solutions that are at least as good as in generation 2. On the very large scale set of instances, the algorithm shows similar performances on generation 1 and 2 even if it is slower on generation 1 instances. Worse results are again obtained for generation 3 instances. This behavior might be caused by the clustering that partitions the area and which might not be relevant for this specific kind of instances. Overall, our algorithm remains very competitive on every kind of profit distributed instances.

6.6. Convergence

We then analyse the average convergence of the algorithm based on the set of instances tackled. Results are very interesting as our algorithm quickly converge independently of the set of instances tackled: for the small scale set, solutions reach 99% of the profit of the final solution in 0.075s. For the large scale set, the same value is reached in 0.39s while it is reached in 125s for the very large scale set. Overall, initial solutions worsen with the increase in size of the instance tackled, going from 88.43% for the small scale set to 75.18% for the very large scale set. This shows that the heuristic used to build the initial solution doesn't scale well with the instance size, which seems reasonable as these instances are harder to solve. For every set of instances, most of the time the algorithm does not improve much the solution's quality when getting close to one of the stopping criterion, thus confirming their relevance. It is also of interest to note that, based on its convergence, our algorithm can be stopped earlier in an industrial context where a bigger compromise between the solution's quality and its associated computational time may be

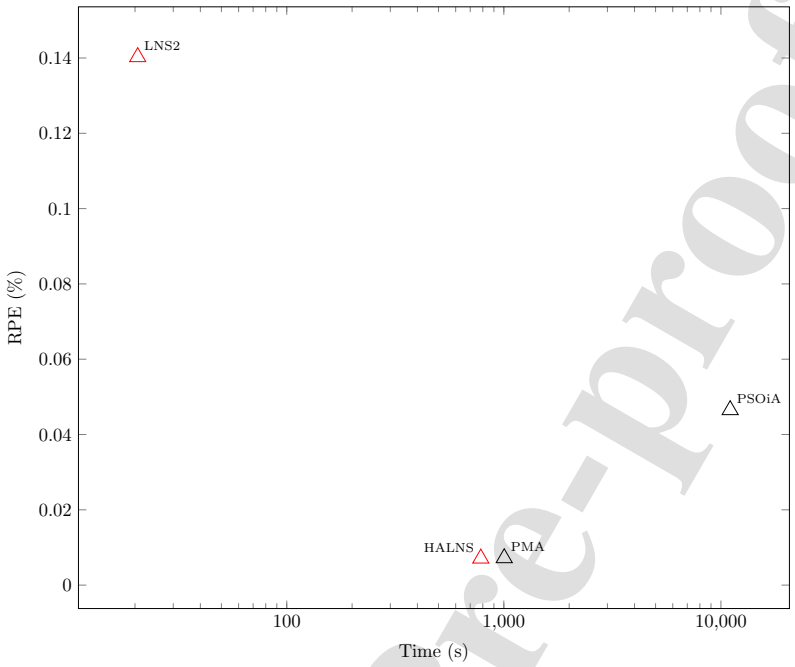


Figure 7: RPE based on the average computational time and the algorithm used for the large scale set of instances.

Criteria	PSOiA	PMA	HALNS	**LNS2**
# BKS	70	78	**79**	54*
RPE (%)	0.04688	0.00722	**0.00712**	0.14027
ARPE (%)	0.50529	0.42899	**0.30205**	0.68628
Average CPU (s)	11,031.04	1,004.15	783.36	**20.58**

*3 new best known solutions

Table 12: Summary results for the set of large scale instances.

Instances	#	ARPE (%)	CPU (s)
1000–1500	117	1.571	164.891
1500–2000	60	1.496	259.342
2000–3000	48	1.248	400.172
3000-6000	60	1.670	1699.364
All	285	1.521	547.449

Table 13: Summary results for the set of very large scale instances.

Instance set	**m**	**#**	**RPE (%)**	**ARPE (%)**	**Average CPU (s)**
Small scale	2	60	0.031	0.252	4.492
	3	52	0.051	0.237	4.583
	4	45	0.009	0.177	4.940
Large scale	2	29	0.151	0.701	19.320
	3	29	0.190	0.799	20.565
	4	24	0.067	0.532	22.138
Very large scale	2	72	–	1.617	705.736
	3	72	–	2.068	588.653
	8	72	–	1.214	442.164
	10	69	–	1.173	449.145

Table 14: Impact of the number of vehicles.

Instance set	**Profit generation**	**#**	**RPE (%)**	**ARPE (%)**	**Average CPU (s)**
Large scale	Gen 1	17	0.120	0.807	21.910
	Gen 2	45	0.096	0.523	18.612
	Gen 3	20	0.207	0.952	22.920
Very large scale	Gen 1	95	–	1.298	566.911
	Gen 2	95	–	1.373	539.418
	Gen 3	95	–	1.893	536.017

Table 15: Impact of profit generation.

needed. Thus, the algorithm meets the requirements linked to the computational time that we were aiming for.

7. Numerical results on industrial instances

Our algorithm is then evaluated on industrial instances. In this context, each vehicle has its own depot and they don't need to finish their route on their depot. The algorithm is adapted to solve MDOTOP instances from TOP ones: each depot is specific to each vehicle, and final depots are dummy points having no cost to reach.

7.1. Industrial instances

Our algorithm is tested on 10 industrial instances ranging from 3133 to 7358 customers and considering 6 to 18 vehicles. We consider a time limit D of two hours with vehicles moving at an average speed of 4 km/h. Industrial solutions considered are computed based on geographical information as in practice, and validated by an industrial expert. As our goal is to propose new solutions every 10 minutes, the associated parameter is set to this value. This won't degrade much the quality of the proposed solution as our algorithm is very fast to converge. Moreover, in our industrial context, computing time is as important as the

solution's quality since hazards regularly change the initial planning. Thus, stopping the algorithm at 10 minutes is reasonable to tackle subsurface imaging instances. The set of parameters introduced for the very large scale set of instances is used for the other parameters as these instances are similar. An example of instance is illustrated figure 8.

7.2. Benchmark

To evaluate the method, we compare 10 executions of our algorithm on all industrial instances considered. The ARPE value, based on the best solution found, is compared for both industrial and algorithmic methods. Results can be seen in table 16 where industrial solutions are compared with solutions obtained using our metaheuristic.

It shows that our algorithm outperforms the traditional way to build solutions used in this industrial context. In particular, the number of points visited is 38% lower using industrial solutions based on LNS2 results. The algorithm is, most of the time, stopped by the maximum computational time of 10mn which is higher than the algorithm based on pre-computed geographical informations used to build industrial solutions, which is immediate. Our algorithm obtains similar results among the executions, as shown by the average ARPE value (0.47%), showing that the budget time of 10 minutes doesn't have much effect on the convergence of the algorithm for an industrial application. Overall, the results obtained using our metaheuristic are way more interesting than the classical way to build solutions in the subsurface imaging field, improving in average by 62% the number of visited points.

8. Conclusion and perspectives

We proposed a LNS to tackle the very large scale MDOTOP. To our knowledge, this is the first time this problem is treated in the literature. Our LNS is paired with mechanisms which aim at reducing the number of evaluations during the algorithm. Results on literature instances for the TOP are promising since it finds a good number of the best known solutions while being much quicker than state-of-the-art algorithms. Three best-known solutions were improved for the large scale set of instances. Our LNS is also applied on a new set of very large scale instances based on OP instances. Results show that our algorithm is fast for most of these instances but is slower the longer the routes are. The algorithm is shown to be very good independently of the type of instance tackled, even if worse results are obtained when customers'profit is based on their distance from the depot.

Our algorithm is then applied on industrial instances from the subsurface imaging field. On these instances, our algorithm greatly improves the performance of vehicles, increasing in average by 62% their productivity. Overall, the simplicity of our method associated to the quality of the obtained results based on its computational time makes our algorithm very interesting to be applied in most industrial contexts with large size problems.

Perspectives includes the application of our algorithm on different industrial contexts. The use of Static Move Descriptors should also be interesting to avoid some evaluations and may bring more diversification in the process. Finally, further work can be done around pre-processing and memory management to avoid some computations and better scale with the instance size. Both these mechanisms might be needed to tackle instances even bigger like it is done in the VRP.

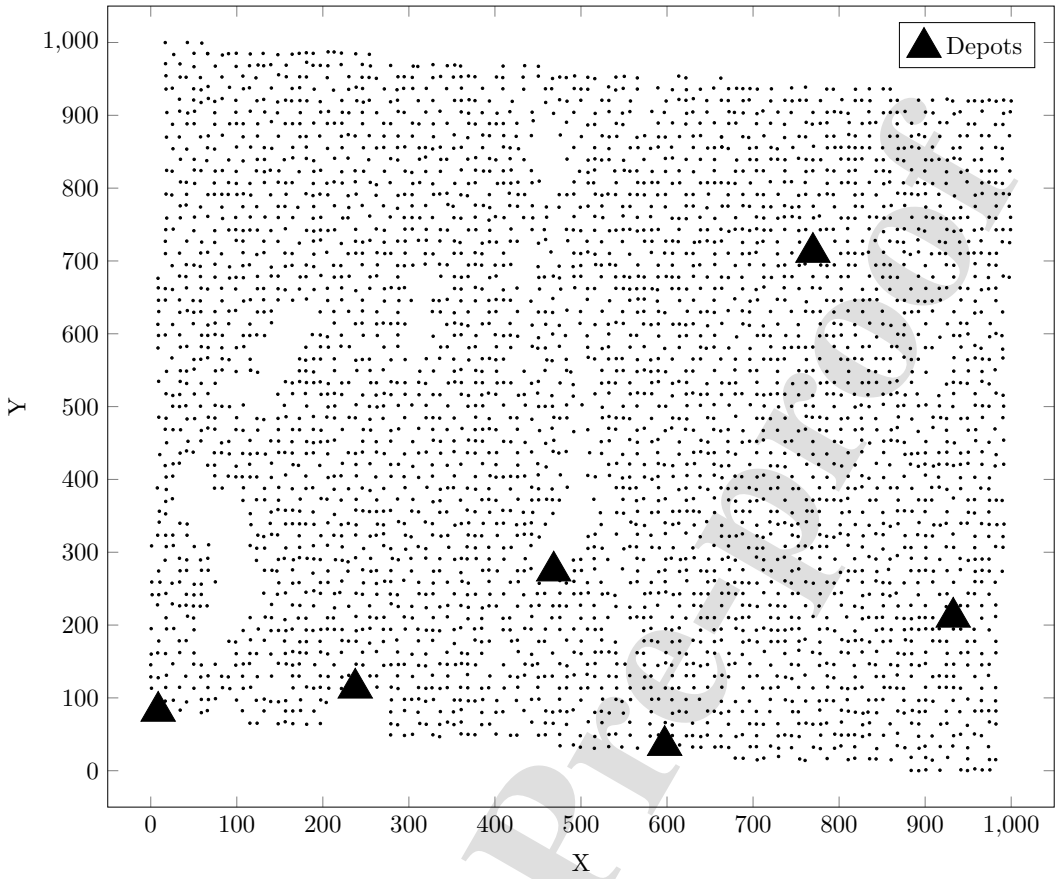


Figure 8: Example of instance considered when solving the subsurface imaging problem.

Instance	N	m	Industrial solution	LNS2			ARPE (%)	
				Best	Average	CPU (s)	Industrial	LNS2
Instance 1	3133	8	1424	**2297**	2290	587.53	38.01	**0.3**
Instance 2	3155	7	1299	**2042**	2033.4	606.91	36.39	**0.42**
Instance 3	3673	6	1052	**1749**	1740.9	614.46	39.85	**0.46**
Instance 4	3745	11	2005	**3102**	3087.9	603.67	35.36	**0.45**
Instance 5	3953	9	1675	**2588**	2572.1	607.84	35.28	**0.61**
Instance 6	4116	10	1897	**2899**	2885.5	606.75	34.56	**0.47**
Instance 7	4144	9	1383	**2535**	2526.2	608.02	45.44	**0.35**
Instance 8	4836	17	2881	**4628**	4617.6	604.22	37.75	**0.22**
Instance 9	6116	18	3365	**5092**	5058.7	608.01	33.92	**0.65**
Instance 10	7358	10	1550	**2864**	2842.3	650.58	45.88	**0.76**
Average	4422.9	10.5	1853.1	2979.6	2965.46	609.8	38.24	**0.47**

Table 16: Results on industrial instances. CPU time higher than 600s are met when the algorithm is stopped from the CPU stopping criterion.

9. Appendix

9.1. Results on the small scale set of instances

Instances	BKS	SkVNS	GLS	FPR	MA	AuLNS	PSOiA	UHGS	MS-ILS	MS-LS	PMA	LNS	HALNS	HISS-SD	LNS2
p4.2.a	206	202	**206**	**206**	**206**	**206**	**206**	**206**	**206**	**206**	**206**	**206**	**206**	**206**	**206**
p4.2.b	341	**341**	303	**341**	**341**	**341**	**341**	**341**	**341**	**341**	**341**	**341**	**341**	**341**	**341**
p4.2.c	452	**452**	447	**452**	**452**	**452**	**452**	**452**	**452**	**452**	**452**	**452**	**452**	**452**	**452**
p4.2.d	531	528	526	**531**	**531**	**531**	**531**	**531**	**531**	**531**	**531**	**531**	**531**	**531**	**531**
p4.2.e	618	593	602	612	**618**	**618**	**618**	**618**	**618**	**618**	**618**	**618**	**618**	**618**	**618**
p4.2.f	687	675	651	**687**	**687**	**687**	**687**	**687**	**687**	**687**	**687**	**687**	**687**	**687**	**687**
p4.2.g	757	750	734	**757**	**757**	**757**	**757**	**757**	**757**	752	**757**	**757**	**757**	**757**	**757**
p4.2.h	835	819	797	**835**	**835**	**835**	**835**	**835**	**835**	825	**835**	**835**	**835**	**835**	827
p4.2.i	918	916	826	**918**	**918**	**918**	**918**	**918**	**918**	**918**	**918**	**918**	**918**	**918**	**918**
p4.2.j	965	962	939	962	**965**	**965**	**965**	**965**	**965**	964	**965**	**965**	**965**	**965**	**965**
p4.2.k	1022	1007	994	1013	**1022**	**1022**	**1022**	**1022**	**1022**	**1022**	**1022**	**1022**	**1022**	**1022**	1013
p4.2.l	1074	1051	1051	1064	1074	**1074**	**1074**	**1074**	**1074**	1073	**1074**	**1074**	**1074**	**1074**	**1074**
p4.2.m	1132	1051	1051	1130	**1132**	**1132**	**1132**	**1132**	**1132**	**1132**	**1132**	**1132**	**1132**	**1132**	**1132**
p4.2.n	1174	1124	1117	1161	**1174**	**1174**	**1174**	**1174**	**1174**	1172	**1174**	**1174**	**1174**	**1174**	**1174**
p4.2.o	1218	1195	1191	1206	**1218**	**1218**	**1218**	**1218**	**1218**	1213	**1218**	**1218**	**1218**	**1218**	**1218**
p4.2.p	1242	1237	1214	1240	**1242**	**1242**	**1242**	**1242**	**1242**	1239	**1242**	**1242**	**1242**	**1242**	**1242**
p4.2.q	1268	1239	1248	1257	1268	**1268**	**1268**	1267	1267	1262	**1268**	**1268**	**1268**	1267	**1268**
p4.2.r	1292	1279	1267	1278	**1292**	**1292**	**1292**	**1292**	1287	1285	**1292**	**1292**	**1292**	**1292**	**1292**
p4.2.s	1304	1295	1286	1293	**1304**	**1304**	**1304**	1302	**1304**	1301	**1304**	**1304**	**1304**	**1304**	**1304**
p4.2.t	1306	1305	1294	1299	**1306**	**1306**	**1306**	**1306**	**1306**	**1306**	**1306**	**1306**	**1306**	**1306**	**1306**
p4.3.c	193	**193**	**193**	**193**	**193**	**193**	**193**	**193**	**193**	**193**	**193**	**193**	**193**	**193**	**193**
p4.3.d	335	331	**335**	333	**335**	**335**	**335**	**335**	**335**	**335**	**335**	**335**	**335**	**335**	**335**
p4.3.e	468	460	444	**468**	**468**	**468**	**468**	**468**	**468**	**468**	**468**	**468**	**468**	**468**	**468**
p4.3.f	579	556	564	**579**	**579**	**579**	**579**	**579**	**579**	**579**	**579**	**579**	**579**	**579**	**579**
p4.3.g	653	651	644	**653**	**653**	**653**	**653**	**653**	**653**	**653**	**653**	**653**	**653**	**653**	**653**
p4.3.h	729	718	706	725	729	**729**	**729**	**729**	**729**	728	**729**	**729**	**729**	**729**	**729**
p4.3.i	809	807	806	797	**809**	**809**	**809**	**809**	**809**	**809**	**809**	**809**	**809**	**809**	**809**
p4.3.j	861	854	826	858	**861**	**861**	**861**	**861**	**861**	**861**	**861**	**861**	**861**	**861**	**861**
p4.3.k	919	902	864	918	**919**	**919**	**919**	**919**	**919**	**919**	**919**	**919**	**919**	**919**	**919**
p4.3.l	979	969	960	968	**979**	**979**	**979**	**979**	**979**	**979**	**979**	**979**	**979**	**979**	**979**
p4.3.m	1063	1047	1030	1043	**1063**	**1063**	**1063**	**1063**	**1063**	1051	**1063**	**1063**	**1063**	**1063**	**1063**
p4.3.n	1121	1106	1113	1108	**1121**	**1121**	**1121**	**1121**	**1121**	**1121**	**1121**	**1121**	**1121**	**1121**	**1121**
p4.3.o	1172	1136	1121	1165	**1172**	**1172**	**1172**	**1172**	**1172**	1170	**1172**	**1172**	**1172**	**1172**	**1172**
p4.3.p	1222	1200	1190	1209	**1222**	**1222**	**1222**	**1222**	**1222**	1208	**1222**	**1222**	**1222**	**1222**	**1222**
p4.3.q	1253	1236	1210	1246	**1253**	**1253**	**1253**	**1253**	**1253**	**1253**	**1253**	**1253**	**1253**	**1253**	**1253**
p4.3.r	1273	1250	1239	1257	**1273**	**1273**	**1273**	**1273**	1272	1271	**1273**	**1273**	**1273**	**1273**	**1273**
p4.3.s	1295	1280	1279	1276	**1295**	**1295**	**1295**	**1295**	**1295**	1290	**1295**	**1295**	**1295**	**1295**	**1295**
p4.3.t	1305	1299	1290	1294	**1305**	**1305**	**1305**	**1305**	1304	1301	**1305**	**1305**	**1305**	**1305**	**1305**
p4.4.e	183	**183**	**183**	**183**	**183**	**183**	**183**	**183**	**183**	**183**	**183**	**183**	**183**	**183**	**183**
p4.4.f	324	319	312	**324**	**324**	**324**	**324**	**324**	**324**	**324**	**324**	**324**	**324**	**324**	**324**
p4.4.g	461	**461**	**461**	**461**	**461**	**461**	**461**	**461**	**461**	**461**	**461**	**461**	**461**	**461**	**461**
p4.4.h	571	553	565	**571**	**571**	**571**	**571**	**571**	**571**	**571**	**571**	**571**	**571**	**571**	**571**
p4.4.i	657	**657**	**657**	653	**657**	**657**	**657**	**657**	**657**	**657**	**657**	**657**	**657**	**657**	**657**
p4.4.j	732	723	691	**732**	**732**	**732**	**732**	**732**	**732**	**732**	**732**	**732**	**732**	**732**	**732**
p4.4.k	821	**821**	815	820	**821**	**821**	**821**	**821**	**821**	**821**	**821**	**821**	**821**	**821**	**821**
p4.4.l	880	876	852	875	**880**	**880**	**880**	**880**	**880**	**880**	**880**	**880**	**880**	**880**	**880**
p4.4.m	919	903	910	914	919	**919**	**919**	**919**	**919**	**919**	**919**	**919**	**919**	**919**	**919**
p4.4.n	976	948	942	953	976	**976**	**976**	**976**	**976**	967	**976**	**976**	**976**	**976**	**976**
p4.4.o	1061	1030	937	1033	**1061**	**1061**	**1061**	**1061**	**1061**	1057	**1061**	**1061**	**1061**	**1061**	**1061**
p4.4.p	1124	1120	1091	1098	**1124**	**1124**	**1124**	**1124**	**1124**	**1124**	**1124**	**1124**	**1124**	**1124**	**1124**
p4.4.q	1161	1149	1106	1139	**1161**	**1161**	**1161**	**1161**	**1161**	1159	**1161**	**1161**	**1161**	**1161**	**1161**
p4.4.r	1216	1193	1148	1196	**1216**	**1216**	**1216**	**1216**	**1216**	1207	**1216**	**1216**	**1216**	**1216**	**1216**
p4.4.s	1260	1213	1242	1231	**1260**	**1260**	**1260**	**1260**	**1260**	1259	**1260**	**1260**	**1260**	**1260**	**1260**
p4.4.t	1285	1281	1250	1256	**1285**	**1285**	**1285**	**1285**	**1285**	1280	**1285**	**1285**	**1285**	**1285**	**1285**

Table 17: Results for the small scale instances set 4.

Instances	BKS	SkVNS	GLS	FPR	MA	AuLNS	PSOiA	UHGS	MS-ILS	MS-LS	PMA	LNS	HALNS	HISS-SD	LNS2
p5.2.h	410	395	385	**410**	**410**	**410**	**410**	**410**	**410**	**410**	**410**	**410**	**410**	**410**	**410**
p5.2.j	580	**580**	**580**	**580**	**580**	**580**	**580**	**580**	**580**	**580**	**580**	**580**	**580**	**580**	**580**
p5.2.k	670	**670**	665	**670**	**670**	**670**	**670**	**670**	**670**	**670**	**670**	**670**	**670**	**670**	**670**
p5.2.l	800	770	760	**800**	**800**	**800**	**800**	**800**	**800**	**800**	**800**	**800**	**800**	**800**	**800**
p5.2.m	860	**860**	830	**860**	**860**	**860**	**860**	**860**	**860**	**860**	**860**	**860**	**860**	**860**	**860**
p5.2.n	925	920	920	**925**	**925**	**925**	**925**	**925**	**925**	**925**	**925**	**925**	**925**	**925**	**925**
p5.2.o	1020	**1020**	1010	**1020**	**1020**	**1020**	**1020**	**1020**	**1020**	**1020**	**1020**	**1020**	**1020**	**1020**	**1020**
p5.2.p	1150	**1150**	1030	**1150**	**1150**	**1150**	**1150**	**1150**	**1150**	**1150**	**1150**	**1150**	**1150**	**1150**	**1150**
p5.2.q	1195	**1195**	1145	**1195**	**1195**	**1195**	**1195**	**1195**	**1195**	**1195**	**1195**	**1195**	**1195**	**1195**	**1195**
p5.2.r	1260	**1260**	1225	**1260**	**1260**	**1260**	**1260**	**1260**	**1260**	**1260**	**1260**	**1260**	**1260**	**1260**	**1260**
p5.2.s	1340	1325	1325	**1340**	1340	**1340**	**1340**	**1340**	**1340**	**1340**	**1340**	**1340**	**1340**	**1340**	**1340**
p5.2.t	1400	1380	1360	**1400**	**1400**	**1400**	**1400**	**1400**	**1400**	**1400**	**1400**	**1400**	**1400**	**1400**	**1400**
p5.2.u	1460	1450	**1460**	**1460**	**1460**	**1460**	**1460**	**1460**	**1460**	**1460**	**1460**	**1460**	**1460**	**1460**	**1460**
p5.2.v	1505	1500	1500	**1505**	**1505**	**1505**	**1505**	**1505**	**1505**	**1505**	**1505**	**1505**	**1505**	**1505**	**1505**
p5.2.w	1565	1560	1560	1560	1565	**1565**	**1565**	**1565**	**1565**	1560	**1565**	1560	**1565**	**1565**	**1565**
p5.2.x	1610	1600	**1610**	**1610**	**1610**	**1610**	**1610**	**1610**	**1610**	**1610**	**1610**	**1610**	**1610**	**1610**	**1610**
p5.2.y	1645	1630	1630	**1645**	**1645**	**1645**	**1645**	**1645**	**1645**	1640	**1645**	**1645**	**1645**	**1645**	**1645**
p5.2.z	1680	1665	**1680**	1670	**1680**	**1680**	**1680**	**1680**	**1680**	**1680**	**1680**	**1680**	**1680**	**1680**	**1680**
p5.3.k	495	**495**	470	**495**	**495**	**495**	**495**	**495**	**495**	**495**	**495**	**495**	**495**	**495**	**495**
p5.3.l	595	**595**	545	**595**	**595**	**595**	**595**	**595**	**595**	**595**	**595**	**595**	**595**	**595**	**595**
p5.3.n	755	**755**	720	**755**	**755**	**755**	**755**	**755**	**755**	**755**	**755**	**755**	**755**	**755**	**755**
p5.3.o	870	**870**	**870**	**870**	**870**	**870**	**870**	**870**	**870**	**870**	**870**	**870**	**870**	**870**	**870**
p5.3.q	1070	1065	1045	**1070**	**1070**	**1070**	**1070**	**1070**	**1070**	**1070**	**1070**	**1070**	**1070**	**1070**	**1070**
p5.3.r	1125	**1125**	1090	**1125**	**1125**	**1125**	**1125**	**1125**	**1125**	**1125**	**1125**	**1125**	**1125**	**1125**	**1125**
p5.3.s	1190	1185	1145	1185	**1190**	**1190**	**1190**	**1190**	**1190**	**1190**	**1190**	**1190**	**1190**	**1190**	**1190**
p5.3.t	1260	**1260**	1240	**1260**	**1260**	**1260**	**1260**	**1260**	**1260**	**1260**	**1260**	**1260**	**1260**	**1260**	**1260**
p5.3.u	1345	**1345**	1305	1335	**1345**	**1345**	**1345**	**1345**	**1345**	**1345**	**1345**	**1345**	**1345**	**1345**	**1345**
p5.3.v	1425	**1425**	**1425**	1420	**1425**	**1425**	**1425**	**1425**	**1425**	**1425**	**1425**	**1425**	**1425**	**1425**	**1425**
p5.3.w	1485	1475	1460	1465	**1485**	**1485**	**1485**	**1485**	**1485**	1480	**1485**	**1485**	**1485**	**1485**	**1485**
p5.3.x	1555	1535	1520	1540	**1555**	**1555**	**1555**	**1555**	**1555**	1535	**1555**	**1555**	**1555**	**1555**	**1555**
p5.3.y	1595	1580	1590	1590	1595	**1595**	**1595**	**1595**	**1595**	1590	**1595**	**1595**	**1595**	**1595**	**1595**
p5.3.z	1635	**1635**	**1635**	**1635**	**1635**	**1635**	**1635**	**1635**	**1635**	**1635**	**1635**	**1635**	**1635**	**1635**	**1635**
p5.4.m	555	550	550	**555**	**555**	**555**	**555**	**555**	**555**	**555**	**555**	**555**	**555**	**555**	**555**
p5.4.o	690	**690**	680	**690**	**690**	**690**	**690**	**690**	**690**	**690**	**690**	**690**	**690**	**690**	**690**
p5.4.p	765	760	760	760	765	**765**	**765**	**765**	**765**	**765**	**765**	**765**	**765**	**765**	**765**
p5.4.q	860	835	830	**860**	**860**	**860**	**860**	**860**	**860**	**860**	**860**	**860**	**860**	**860**	**860**
p5.4.r	960	**960**	890	**960**	**960**	**960**	**960**	**960**	**960**	**960**	**960**	**960**	**960**	**960**	**960**
p5.4.s	1030	1020	1020	1005	**1030**	**1030**	**1030**	**1030**	**1030**	**1030**	**1030**	**1030**	**1030**	**1030**	**1030**
p5.4.t	1160	**1160**	**1160**	**1160**	**1160**	**1160**	**1160**	**1160**	**1160**	**1160**	**1160**	**1160**	**1160**	**1160**	**1160**
p5.4.u	1300	**1300**	**1300**	**1300**	**1300**	**1300**	**1300**	**1300**	**1300**	**1300**	**1300**	**1300**	**1300**	**1300**	**1300**
p5.4.v	1320	**1320**	1245	**1320**	**1320**	**1320**	**1320**	**1320**	**1320**	**1320**	**1320**	**1320**	**1320**	**1320**	**1320**
p5.4.w	1390	1380	1330	1380	1390	**1390**	**1390**	**1390**	**1390**	**1390**	**1390**	**1390**	**1390**	**1390**	**1390**
p5.4.x	1450	1440	1410	1430	**1450**	**1450**	**1450**	**1450**	**1450**	**1450**	**1450**	**1450**	**1450**	**1450**	**1450**
p5.4.y	1520	1500	1485	**1520**	**1520**	**1520**	**1520**	**1520**	**1520**	**1520**	**1520**	**1520**	**1520**	**1520**	**1520**
p5.4.z	1620	1600	1590	**1620**	**1620**	**1620**	**1620**	**1620**	**1620**	**1620**	**1620**	**1620**	**1620**	**1620**	**1620**

Table 18: Results for the small scale instances set 5.

Instances	BKS	SkVNS	GLS	FPR	MA	AuLNS	PSOiA	UHGS	MS-ILS	MS-LS	PMA	LNS	HALNS	HISS-SD	LNS2
p6.2.d	192	**192**	180	**192**	**192**	**192**	**192**	**192**	**192**	**192**	**192**	**192**	**192**	**192**	**192**
p6.2.j	948	**948**	**948**	942	**948**	**948**	**948**	**948**	**948**	**948**	**948**	**948**	**948**	**948**	**948**
p6.2.l	1116	**1116**	1104	1110	**1116**	**1116**	**1116**	**1116**	**1116**	**1116**	**1116**	**1116**	**1116**	**1116**	**1116**
p6.2.m	1188	**1188**	1164	**1188**	**1188**	**1188**	**1188**	**1188**	**1188**	**1188**	**1188**	**1188**	**1188**	**1188**	**1188**
p6.2.n	1260	1248	1254	**1260**	**1260**	**1260**	**1260**	**1260**	**1260**	1254	**1260**	**1260**	**1260**	**1260**	**1260**
p6.3.g	282	276	264	**282**	**282**	**282**	**282**	**282**	**282**	**282**	**282**	**282**	**282**	**282**	**282**
p6.3.h	444	**444**	**444**	**444**	**444**	**444**	**444**	**444**	**444**	**444**	**444**	**444**	**444**	**444**	**444**
p6.3.i	642	**642**	**642**	**642**	**642**	**642**	**642**	**642**	**642**	**642**	**642**	**642**	**642**	**642**	**642**
p6.3.k	894	**894**	882	**894**	**894**	**894**	**894**	**894**	**894**	**894**	**894**	**894**	**894**	**894**	**894**
p6.3.l	1002	996	990	**1002**	**1002**	**1002**	**1002**	**1002**	**1002**	**1002**	**1002**	**1002**	**1002**	**1002**	**1002**
p6.3.m	1080	**1080**	1068	**1080**	**1080**	**1080**	**1080**	**1080**	**1080**	**1080**	**1080**	**1080**	**1080**	**1080**	**1080**
p6.3.n	1170	1152	1140	1164	**1170**	**1170**	**1170**	**1170**	**1170**	**1170**	**1170**	**1170**	**1170**	**1170**	**1170**
p6.4.j	366	**366**	360	**366**	**366**	**366**	**366**	**366**	**366**	**366**	**366**	**366**	**366**	**366**	**366**
p6.4.k	528	**528**	**528**	**528**	**528**	**528**	**528**	**528**	**528**	**528**	**528**	**528**	**528**	**528**	**528**
p6.4.l	696	678	678	**696**	**696**	**696**	**696**	**696**	**696**	**696**	**696**	**696**	**696**	**696**	**696**

Table 19: Results for the small scale instances set 6.

Instances	BKS	SkVNS	GLS	FPR	MA	AuLNS	PSOiA	UHGS	MS-ILS	MS-LS	PMA	LNS	HALNS	HISS-SD	LNS2
p7.2.d	190	182	**190**	**190**	**190**	**190**	**190**	**190**	**190**	**190**	**190**	**190**	**190**	**190**	**190**
p7.2.e	290	289	279	**290**	**290**	**290**	**290**	**290**	**290**	**290**	**290**	**290**	**290**	**290**	**290**
p7.2.f	387	**387**	340	**387**	**387**	**387**	**387**	**387**	**387**	**387**	**387**	**387**	**387**	**387**	**387**
p7.2.g	459	457	440	**459**	**459**	**459**	**459**	**459**	**459**	**459**	**459**	**459**	**459**	**459**	**459**
p7.2.h	521	**521**	517	**521**	**521**	**521**	**521**	**521**	**521**	**521**	**521**	**521**	**521**	**521**	**521**
p7.2.i	580	579	568	578	**580**	**580**	**580**	**580**	**580**	**580**	**580**	**580**	**580**	**580**	**580**
p7.2.j	646	632	633	**646**	**646**	**646**	**646**	**646**	**646**	**646**	**646**	**646**	**646**	**646**	**646**
p7.2.k	705	700	691	702	**705**	**705**	**705**	**705**	**705**	**705**	**705**	**705**	**705**	**705**	**705**
p7.2.l	767	758	748	759	**767**	**767**	**767**	**767**	**767**	**767**	**767**	**767**	**767**	**767**	**767**
p7.2.m	827	**827**	798	816	**827**	**827**	**827**	**827**	**827**	**827**	**827**	**827**	**827**	**827**	**827**
p7.2.n	888	866	861	**888**	**888**	**888**	**888**	**888**	**888**	884	**888**	**888**	**888**	**888**	**888**
p7.2.o	945	928	897	932	**945**	**945**	**945**	**945**	**945**	**945**	**945**	**945**	**945**	**945**	**945**
p7.2.p	1002	955	954	993	**1002**	**1002**	**1002**	**1002**	**1002**	**1002**	**1002**	**1002**	**1002**	**1002**	**1002**
p7.2.q	1044	1029	1031	1043	**1044**	**1044**	**1044**	**1044**	**1044**	**1044**	**1044**	**1044**	**1044**	1043	**1044**
p7.2.r	1094	1069	1075	1076	**1094**	**1094**	**1094**	**1094**	**1094**	1085	**1094**	**1094**	**1094**	**1094**	**1094**
p7.2.s	1136	1118	1102	1125	**1136**	**1136**	**1136**	**1136**	**1136**	1133	**1136**	**1136**	**1136**	**1136**	**1136**
p7.2.t	1179	1154	1142	1168	**1179**	**1179**	**1179**	**1179**	**1179**	1170	**1179**	**1179**	**1179**	**1179**	**1179**
p7.3.h	425	**425**	418	**425**	**425**	**425**	**425**	**425**	**425**	**425**	**425**	**425**	**425**	**425**	**425**
p7.3.i	487	480	480	485	**487**	**487**	**487**	**487**	**487**	**487**	**487**	**487**	**487**	**487**	**487**
p7.3.j	564	543	539	560	**564**	**564**	**564**	**564**	**564**	**564**	**564**	**564**	**564**	**564**	560
p7.3.k	633	**633**	586	**633**	**633**	**633**	**633**	**633**	**633**	**633**	**633**	**633**	**633**	**633**	**633**
p7.3.l	684	681	668	**684**	**684**	**684**	**684**	**684**	**684**	**684**	**684**	**684**	**684**	**684**	**684**
p7.3.m	762	743	735	**762**	**762**	**762**	**762**	**762**	**762**	**762**	**762**	**762**	**762**	**762**	**762**
p7.3.n	820	804	789	813	**820**	**820**	**820**	**820**	**820**	**820**	**820**	**820**	**820**	**820**	804
p7.3.o	874	841	833	859	**874**	**874**	**874**	**874**	**874**	**874**	**874**	**874**	**874**	**874**	**874**
p7.3.p	929	918	912	925	**929**	**929**	**929**	**929**	**929**	925	**929**	**929**	**929**	927	**929**
p7.3.q	987	966	945	970	**987**	**987**	**987**	**987**	**987**	984	**987**	**987**	**987**	**987**	**987**
p7.3.r	1026	1009	1015	1017	**1026**	**1026**	**1026**	**1026**	**1026**	1024	**1026**	**1026**	**1026**	**1026**	**1026**
p7.3.s	1081	1070	1054	1076	**1081**	**1081**	**1081**	**1081**	**1081**	1074	**1081**	**1081**	**1081**	**1081**	**1081**
p7.3.t	1120	1109	1080	1111	**1120**	**1120**	**1120**	**1120**	**1120**	1113	**1120**	**1120**	**1120**	**1120**	**1120**
p7.4.g	217	**217**	209	**217**	**217**	**217**	**217**	**217**	**217**	**217**	**217**	**217**	**217**	**217**	**217**
p7.4.h	285	283	**285**	**285**	**285**	**285**	**285**	**285**	**285**	**285**	**285**	**285**	**285**	**285**	**285**
p7.4.i	366	364	359	**366**	**366**	**366**	**366**	**366**	**366**	**366**	**366**	**366**	**366**	**366**	**366**
p7.4.k	520	518	511	518	**520**	**520**	**520**	**520**	**520**	**520**	**520**	**520**	**520**	**520**	518
p7.4.l	590	575	573	581	**590**	**590**	**590**	**590**	**590**	**590**	**590**	**590**	**590**	**590**	**590**
p7.4.m	646	639	638	**646**	**646**	**646**	**646**	**646**	**646**	**646**	**646**	**646**	**646**	**646**	**646**
p7.4.n	730	723	698	723	730	**730**	**730**	**730**	**730**	726	**730**	**730**	**730**	**730**	**730**
p7.4.o	781	778	761	780	**781**	**781**	**781**	**781**	**781**	777	**781**	**781**	**781**	**781**	**781**
p7.4.p	846	841	803	842	**846**	**846**	**846**	**846**	**846**	**846**	**846**	**846**	**846**	**846**	**846**
p7.4.q	909	896	899	902	**909**	**909**	**909**	**909**	**909**	904	**909**	**909**	**909**	**909**	**909**
p7.4.r	970	964	937	961	**970**	**970**	**970**	**970**	**970**	**970**	**970**	**970**	**970**	**970**	**970**
p7.4.s	1022	1019	1005	**1022**	**1022**	**1022**	**1022**	**1022**	**1022**	**1022**	**1022**	**1022**	**1022**	**1022**	**1022**
p7.4.t	1077	1073	1020	1066	**1077**	**1077**	**1077**	**1077**	**1077**	**1077**	**1077**	**1077**	**1077**	**1077**	**1077**

Table 20: Results for the small scale instances set 7.

9.2. Results on the large scale set of instances

Instance	Best known		PSOiA			PAM			HALNS			LNS2		
	Solution	CPU (s)	Best	Mean	CPU (s)	Best	Mean	CPU (s)	Best	Mean	CPU (s)	Best	Mean	CPU (s)
bier127_gen1_m2	106	7.233	**106**	104.8	1153.87	**106**	105	673.5	**106**	105.5	207.2	**106**	105.35	**7.233**
bier127_gen1_m3	103	10.52	**103**	102.4	591.89	**103**	102.5	470.1	**103**	102.6	209.55	**103**	102.15	**10.52**
bier127_gen2_m2	5464	7.572	**5464**	5446.8	1132.57	**5464**	5454.9	398.8	**5464**	5460.2	301.33	**5464**	5442.45	**7.572**
bier127_gen2_m3	5393	5.457	**5393**	5376.2	648.08	**5393**	5386.9	359.3	**5393**	5390.1	227.51	**5393**	5392.6	**5.457**
bier127_gen2_m4	5123	355.99	5122	5119.2	657.57	**5123**	5120.2	383.8	**5123**	5121	**355.99**	5122	5121.15	7.081
bier127_gen3_m2	2885	6.168	**2885**	2884.3	1301.27	**2885**	2884.7	296.7	**2885**	2884.5	184.99	**2885**	2872.85	**6.168**
bier127_gen3_m3	2706	6.164	**2706**	2703.8	711.74	**2706**	2705.2	509.6	**2706**	2705.6	69.36	**2706**	2703.7	**6.164**
bier127_gen3_m4	2402	5.746	**2402**	2384.6	680.79	**2402**	2398.6	227.7	**2402**	2399.4	222.4	**2402**	2402	**5.746**
cmt101c_m3	1300	4.816	**1300**	1299	111.11	**1300**	1299	42	**1300**	1299.4	23.1	**1300**	1288	**4.816**
cmt151b_m3	1385	10.015	**1385**	1373.8	754.01	**1385**	1374.5	169.9	**1385**	1384	164.65	**1385**	1382.45	**10.015**
cmt151c_m2	1964	13.971	1963	1962	1799.64	**1964**	1962	368.5	**1964**	1962.2	131.96	**1964**	1960.6	**13.971**
cmt151c_m3	1916	355.61	**1916**	1909.1	1376.24	**1916**	1909.2	441.5	**1916**	1914.4	**355.61**	1911	1908.95	12.14
cmt151c_m4	1880	9.371	**1880**	1875.6	881.11	**1880**	1877.6	826.5	**1880**	1878.2	107.89	**1880**	1875.5	**9.371**
cmt200b_m2	2096	17.186	**2096**	2088.2	4180.99	**2096**	2086.8	669.4	**2096**	2090.2	183.28	**2096**	2094.2	**17.186**
cmt200b_m3	2019	21.687	**2019**	2005	2711.66	**2019**	2009.4	1351.7	**2019**	2004.4	386.33	**2019**	2011.5	**21.687**
cmt200b_m4	1894	12.826	**1894**	1889.7	1515.19	**1894**	1891.6	974.5	**1894**	1891	324.31	**1894**	1892.75	**12.826**
cmt200c_m2	2818	368.424	**2818**	2810.1	7320.26	**2818**	2810.6	1048.3	**2818**	2810	368.42	2809	2801.3	22.181
cmt200c_m3	2766	16.684	**2766**	2751.2	4217.29	**2766**	2751.8	1200	**2766**	2751.8	273.01	**2766**	2756.3	**16.684**
cmt200c_m4	2712	395.91	**2712**	2700.6	3004.1	**2712**	2703.3	1411.4	**2712**	2704.4	395.91	2711	2703.3	23.224
eil101b_m3	916	5.893	**916**	913.8	134.39	**916**	914.6	160.6	**916**	915.8	69.44	**916**	910.5	**5.893**
eil101c_m2	1305	4.705	**1305**	1304.8	452.79	**1305**	1304.8	250.5	**1305**	1304.7	153.85	**1305**	1305	**4.705**
eil101c_m3	1251	10.159	**1251**	1244.1	227.61	**1251**	1244.2	95	**1251**	1248.6	116.15	**1251**	1247.4	**10.159**
gil262_gen1_m3	101	13.271	**101**	100.9	1769.31	**101**	100.2	482.6	**101**	100.8	709.66	**101**	100.9	**13.271**
gil262_gen1_m4	78	11.605	**78**	77.1	155.76	**78**	77	123.5	**78**	77.95	76.98	**78**	77.9	**11.605**
gil262_gen2_m2	7498	16.072	**7498**	7457.8	7356.65	**7498**	7458.4	742.1	**7498**	7466.2	387	**7498**	7444.5	**16.072**
gil262_gen2_m3	5615	352.386	**5615**	5608.2	3304.55	**5615**	5604.9	1163.8	**5615**	5609.7	352.39	5614	5590.1	22.556
gil262_gen3_m2	7183	14.639	**7183**	7182.8	9129.3	**7183**	7180	284.2	**7183**	7182.5	532.11	**7183**	7168.4	**14.639**
gil262_gen3_m4	2507	11.092	**2507**	2499.8	276.42	**2507**	2500.1	308.8	**2507**	2501.2	227.15	**2507**	2505.25	**11.092**
gil262a_m2	4078	451.764	**4078**	4056.4	5907.29	**4078**	4066.3	2100	**4078**	4068.8	451.76	4038	4016.15	17.931
gil262a_m4	3175	10.696	**3175**	3174.2	271.83	**3175**	3175	222.6	**3175**	3175	133.79	**3175**	3175	**10.696**
gil262b_m2	8081	545.166	**8081**	8061.1	7473.18	**8081**	8074.1	1267.8	**8081**	8078.5	545.17	8072	8071.75	15.836
gil262b_m3	7585	463.187	**7585**	7574.9	7276.8	**7585**	7566.6	1027.2	**7585**	7569	463.19	7544	7411.95	29.904
gil262b_m4	6781	329.098	**6781**	6742	4878.64	**6781**	6756.7	912.6	**6781**	6761.3	329.1	6780	6760.55	31.56
gil262c_m2	11030	731.25	**11030**	11020	27500.87	**11030**	11016.5	1309	**11030**	11022.4	731.25	11008	10934.85	29.83
gil262c_m3	10757	650.871	**10757**	10714.6	14553.76	**10757**	10715.2	1375.6	**10757**	10713.1	650.87	10755	10683.7	31.52
gil262c_m4	10281	516.497	**10281**	10259.4	8472.01	**10281**	10267.3	1997	**10281**	10262.8	516.5	10280	10264.5	31.707
gr229_gen1_m4	223	46.6	**223**	220.8	11922.02	**223**	220.3	**46.6**	**223**	221	344.28	222	220.7	23.234
gr229_gen2_m3	11566	20.222	**11566**	11551.3	14197.21	**11566**	11557.8	1665.3	**11566**	11559.2	441.8	**11566**	11563.25	**20.222**
gr229_gen2_m4*	11359	20.572	11355	11255.3	18799.5	11355	11328.1	2272	11355	11355	377.28	**11359**	11334.45	**20.572**
gr229_gen3_m3	8056	19.351	**8056**	8051.6	14090.06	**8056**	8052.2	1065.3	**8056**	8055.4	938.75	**8056**	8047.5	**19.351**
gr229_gen3_m4*	7660	19.833	7621	7600	11399.71	7651	7610.5	781.5	7651	7622.9	828.68	**7660**	7610.75	**19.833**
kroA150_gen2_m2	4335	7.697	**4335**	4334.4	892.98	**4335**	4335	495.9	**4335**	4333.2	392.3	**4335**	4335	**7.697**
kroA150_gen3_m3	2726	168.809	**2726**	2719.6	538.01	**2726**	2725.2	597.2	**2726**	2725.8	168.81	2710	2664.15	14.682
kroA200_gen1_m4	81	9.827	**81**	80.4	560.29	**81**	80.6	503.4	**81**	81	114.61	**81**	80.45	**9.827**
kroB200_gen1_m2	111	10.339	**111**	110.4	2344.53	**111**	110.3	663.9	**111**	111	295.3	**111**	111	**10.339**
kroB200_gen2_m2	6185	12.176	**6185**	6182.2	3467.26	**6185**	6183.4	426.7	**6185**	6184.8	438.32	**6185**	6181.5	**12.176**
kroB200_gen2_m4	4944	8.506	**4944**	4942.2	640.66	**4944**	4942.4	582.4	**4944**	4942.5	360.44	**4944**	4944	**8.506**
kroB200_gen3_m2	4765	470.103	**4765**	4757.8	6306.62	**4765**	4762.4	513.9	**4765**	4765	470.1	4703	4669.5	17.803
kroB200_gen3_m3	3028	9.103	**3028**	3016	1713.88	**3028**	3022.4	741.5	**3028**	3022.9	547.22	**3028**	3014.8	**9.103**
lin318_gen1_m2	180	26.906	**180**	170.1	20667.24	**180**	175.3	1168.3	**180**	174.5	2702.83	**180**	173.7	**26.906**
lin318_gen1_m3	149	203.513	**149**	148.6	9014.64	**149**	147.9	721.4	**149**	148.4	203.51	148	148	16.169
lin318_gen2_m2	9544	21.127	**9544**	9533.8	23804.82	**9544**	9537.5	1924.6	**9544**	9539.2	1962.14	**9544**	9493.95	**21.127**
lin318_gen2_m3	7807	28.337	7786	7782.1	9773.63	**7807**	7769.6	1413.5	**7807**	7775.8	1600.58	**7807**	7741	**28.337**
lin318_gen3_m2	7936	19.965	**7936**	7905.6	44029	**7936**	7923.3	1547.3	**7936**	7919.2	581.27	**7936**	7842.45	**19.965**
lin318_gen3_m4	3797	22.165	**3797**	3796.4	1446.26	**3797**	3795.5	970.7	**3797**	3796	128.84	**3797**	3796.05	**22.165**
pr136_gen1_m2	63	8.055	**63**	62.7	451.13	**63**	62.8	107.5	**63**	62.9	70.72	**63**	63	**8.055**
pr136_gen2_m2	3646	6.444	3641	3631.8	601.31	**3646**	3637.4	281.6	**3646**	3642.8	95.62	**3646**	3639.1	**6.444**
pr264_gen1_m4	107	12.052	**107**	106.6	503.07	**107**	106.7	289.8	**107**	106.6	195.83	**107**	106.65	**12.052**
pr264_gen2_m2	6635	642.978	**6635**	6634.2	2048.2	**6635**	6632	719	**6635**	6634.4	642.98	6631	6627.1	15.713
pr264_gen2_m3	6420	11.334	**6420**	6410.7	938.39	**6420**	6417	859	**6420**	6418	527.27	**6420**	6400.5	**11.334**
pr264_gen2_m4	5584	25.988	**5584**	5564.5	590.79	**5584**	5565.1	663.2	**5584**	5566.2	294.7	**5584**	5566	**25.988**
pr264_gen3_m3	2772	16.526	**2772**	2770	1037.51	**2772**	2769.8	922.5	**2772**	2770	1490.82	**2772**	2761.65	**16.526**
pr299_gen1_m2	139	14.544	**139**	138.5	4775.93	**139**	138.3	573.4	**139**	138.8	355.26	**139**	138.25	**14.544**
pr299_gen1_m3	111	33.519	**111**	110.1	1303.73	**111**	109.2	506	**111**	110.2	413.78	**111**	109.2	**33.519**

(To be continued)

Instance	Best known		PSOiA			PAM			HALNS			LNS2		
	Solution	CPU (s)	Best	Mean	CPU (s)	Best	Mean	CPU (s)	Best	Mean	CPU (s)	Best	Mean	CPU (s)
pr299_gen1_m4	84	34.302	**84**	83.6	383.48	**84**	83.6	340.3	**84**	83.4	191.29	**84**	83.8	**34.302**
pr299_gen2_m3	6018	690.41	**6018**	5966.7	1446.05	**6018**	5979.2	909.7	**6018**	5978.5	**690.41**	6005	5961.6	32.493
pr299_gen2_m4	4457	32.49	**4457**	4453	593.41	**4457**	4455.2	767.7	**4457**	4454.8	257.75	**4457**	4446.2	**32.49**
pr299_gen3_m2	5729	692.124	**5729**	5728.6	11872.55	**5729**	5709.8	1489.5	**5729**	5729	692.12	5728	5715.95	41.758
pr299_gen3_m3	3655	31.122	**3655**	3611	2705.82	**3655**	3611.6	1058.2	**3655**	3648.2	644.38	**3655**	3607.8	**31.122**
pr299_gen3_m4	2268	29.293	**2268**	2258	455.64	**2268**	2261.8	402.4	**2268**	2262.3	271.94	**2268**	2183.65	**29.293**
rat195_gen2_m2	5148	335.533	**5148**	5145.6	2156.98	**5148**	5145.9	886.6	**5148**	5147.6	335.53	5145	5135.75	17.739
rat195_gen3_m3	2574	15.321	**2574**	2571.2	721.82	**2574**	2569.9	369.5	**2574**	2570.2	276.84	**2574**	2574	**15.321**
rd400_gen1_m2*	233	51.437	230	227.8	56767.29	232	228.5	3066.6	232	227.9	3925.58	**233**	230.7	**51.437**
rd400_gen1_m3	224	2844	222	221.7	62476.08	**224**	221.3	**2844**	**224**	221.4	4178	223	221.65	48.846
rd400_gen1_m4	213	1985.4	**213**	210.6	34744.8	**213**	209.9	**1985.4**	**213**	210.8	3000.9	212	210	40.603
rd400_gen2_m2	13045	3220.6	12993	12787.5	77049.22	**13045**	12873	**3220.6**	**13045**	12872.4	4035.08	13018	12873.15	49.308
rd400_gen2_m3	12646	2814.581	12645	12372.1	53707.14	12645	12543.9	2852.7	**12646**	12555.8	2814.58	12362	12299.4	51.986
rd400_gen2_m4	12032	3299.3	**12032**	11953.5	42001.58	**12032**	11969.7	**3299.3**	**12032**	11981.2	4070.44	12031	11895.65	50.686
rd400_gen3_m2	12431	2418.4	12428	12274.1	96178.7	**12431**	12312.2	**2418.4**	**12431**	12308	2814.66	12291	12240.45	47.492
rd400_gen3_m3	11639	3500	**11639**	11629.5	68074.77	**11639**	11549.8	**3500**	**11639**	11609.8	3811.24	11579	11291.65	46.578
rd400_gen3_m4	10436	3500	10417	10383.1	48462.77	**10436**	10345.4	**3500**	**10436**	10392.6	3615.46	10374	10282.95	46.846
ts225_gen2_m2	5859	18.454	**5859**	5858.5	2998.43	**5859**	5859	759.6	**5859**	5858.6	686.54	**5859**	5728.1	**18.454**

Table 21: Summary results for the set of large scale instances.

9.3. Results on very large scale set of instances

Instance	Best	Mean	CPU (s)	Instance	Best	Mean	CPU (s)	Instance	Best	Mean	CPU (s)
pr1002_gen1_m10	215	210.45	104.118	fl1400_gen1_m3	538	529.85	211.396	u2152_gen1_m2	1079	1067.55	448.405
pr1002_gen1_m2	550	529.9	117.181	fl1400_gen1_m8	288	281	148.402	u2152_gen1_m3	1069	1056.15	381.973
pr1002_gen1_m3	496	490.05	109.458	fl1400_gen2_m10	5331	5331	117.348	u2152_gen1_m8	844	836.4	331.438
pr1002_gen1_m8	265	260.3	115.722	fl1400_gen2_m2	45519	45459.9	307.618	u2152_gen2_m10	39930	39612.05	329.565
pr1002_gen2_m10	11928	11711.05	102.929	fl1400_gen2_m3	29911	29110.05	187.739	u2152_gen2_m2	60946	60131.45	415.845
pr1002_gen2_m2	30461	28949.05	125.856	fl1400_gen2_m8	14594	14579.6	162.858	u2152_gen2_m3	58881	58499.5	391.487
pr1002_gen2_m3	27175	26791.75	108.35	fl1400_gen3_m10	1450	1434.7	123.051	u2152_gen2_m8	45701	45311.3	316.243
pr1002_gen2_m8	14183	13868.45	111.789	fl1400_gen3_m2	38339	36457.65	295.203	u2152_gen3_m10	20347	20075.9	328.789
pr1002_gen3_m10	5565	5548.15	103.095	fl1400_gen3_m3	20796	16407.65	217.344	u2152_gen3_m2	67950	67022	405.896
pr1002_gen3_m2	34637	34092.3	115.28	fl1400_gen3_m8	5112	5094	154.688	u2152_gen3_m3	61520	60683.1	345.428
pr1002_gen3_m3	27333	26701.6	101.344	u1432_gen1_m10	641	633.55	189.197	u2152_gen3_m8	26383	26053.85	320.334
pr1002_gen3_m8	7739	7518.8	104.374	u1432_gen1_m2	740	735.1	237.625	u2319_gen1_m10	1067	1062	451.098
u1060_gen1_m10	292	288.5	127.988	u1432_gen1_m3	728	722.1	217.797	u2319_gen1_m2	1166	1164.3	615.41
u1060_gen1_m2	605	595.25	142.994	u1432_gen1_m8	667	659.4	201.458	u2319_gen1_m3	1163	1161.2	576.607
u1060_gen1_m3	563	553.85	138.828	u1432_gen2_m10	34889	34591.5	202.634	u2319_gen1_m8	1109	1102.75	472.08
u1060_gen1_m8	391	384.6	117.293	u1432_gen2_m2	45074	44655.75	224.796	u2319_gen2_m10	65270	64825.6	433.703
u1060_gen2_m10	15174	14963.6	132.918	u1432_gen2_m3	44406	43812.15	214.047	u2319_gen2_m2	77763	77315.75	576.063
u1060_gen2_m2	33655	33051.8	138.517	u1432_gen2_m8	37724	37344.5	191.346	u2319_gen2_m3	77407	76794.7	539.57
u1060_gen2_m3	31283	30470.35	128.437	u1432_gen3_m10	18759	18559.7	172.361	u2319_gen2_m8	70603	69863.05	405.996
u1060_gen2_m8	20607	20502.15	123.422	u1432_gen3_m2	44906	44695.95	248.799	u2319_gen3_m10	45807	45428.4	455.556
u1060_gen3_m10	5718	5633	120.408	u1432_gen3_m3	42393	42118.8	191.488	u2319_gen3_m2	76452	76013.15	757.645
u1060_gen3_m2	33931	32963.1	122.56	u1432_gen3_m8	23179	22914.95	175.678	u2319_gen3_m3	73609	73246.9	567.175
u1060_gen3_m3	27044	26607.25	103.197	fl1577_gen1_m10	294	290.3	198.441	u2319_gen3_m8	54524	54219.25	414.717
u1060_gen3_m8	9040	8991.55	111.938	fl1577_gen1_m2	821	810.7	238.721	pr2392_gen1_m10	780	775.4	366.129
vm1084_gen1_m10	422	420.25	161.439	fl1577_gen1_m3	775	745.4	217.173	pr2392_gen1_m2	1267	1252.5	567.095
vm1084_gen1_m2	739	733.2	190.626	fl1577_gen1_m8	397	387.6	201.181	pr2392_gen1_m3	1228	1216.55	491.994
vm1084_gen1_m3	690	685.7	161.704	fl1577_gen2_m10	15676	15586	206	pr2392_gen1_m8	916	905.75	383.213
vm1084_gen1_m8	465	460.3	159.598	fl1577_gen2_m2	43958	42670.1	224.016	pr2392_gen2_m10	41579	41123.2	396.523
vm1084_gen2_m10	21134	20976.4	183.81	fl1577_gen2_m3	40608	39831.9	218.282	pr2392_gen2_m2	71250	69946.4	556.238
vm1084_gen2_m2	40141	38879.15	193.154	fl1577_gen2_m8	22309	22222.7	198.312	pr2392_gen2_m3	68643	67748.05	478.978
vm1084_gen2_m3	37038	36381.15	170.173	fl1577_gen3_m10	4748	4732.55	194.489	pr2392_gen2_m8	49361	48963.25	418.866
vm1084_gen2_m8	23703	23550.35	159.189	fl1577_gen3_m2	33172	32952.15	267.03	pr2392_gen3_m10	24484	24220.55	380.322
vm1084_gen3_m10	7892	7859.9	163.761	fl1577_gen3_m3	26374	25044.8	193.18	pr2392_gen3_m2	79487	78705.55	570.504
vm1084_gen3_m2	34872	34566.3	164.207	fl1577_gen3_m8	8462	8376.25	221.178	pr2392_gen3_m3	74225	73293.25	439.969
vm1084_gen3_m3	32582	32157.6	136.939	d1655_gen1_m10	16	16	104.819	pr2392_gen3_m8	33824	33360.25	370.884
vm1084_gen3_m8	10047	9944.2	152.159	d1655_gen1_m2	755	742	225.14	pcb3038_gen1_m10	1231	1221.4	615.156
pcb1173_gen1_m10	319	318.15	130.556	d1655_gen1_m3	592	580	205.651	pcb3038_gen1_m2	1615	1599.95	1054.969
pcb1173_gen1_m2	633	629	165.288	d1655_gen1_m8	107	105.8	125.122	pcb3038_gen1_m3	1617	1594.9	828.263
pcb1173_gen1_m3	612	605.6	163.247	d1655_gen2_m10	698	698	109.471	pcb3038_gen1_m8	1350	1338.05	616.676
pcb1173_gen1_m8	417	415.45	135.601	d1655_gen2_m2	41425	40941.65	222.491	pcb3038_gen2_m10	67239	66686.65	577.361

(To be continued)

Instance	Best	Mean	CPU (s)	Instance	Best	Mean	CPU (s)	Instance	Best	Mean	CPU (s)
pcb1173_gen2_m10	16665	16556.4	134.288	d1655_gen2_m3	34701	33707.4	192.853	pcb3038_gen2_m2	91808	91013.2	886.666
pcb1173_gen2_m2	34723	34391.65	160.813	d1655_gen2_m8	5490	5447.05	137.195	pcb3038_gen2_m3	90632	90150.4	781.123
pcb1173_gen2_m3	33706	33230.3	152.4	d1655_gen3_m10	511	511	105.376	pcb3038_gen2_m8	75638	74826.6	617.63
pcb1173_gen2_m8	21921	21714.8	142.331	d1655_gen3_m2	48683	45426.3	199.828	pcb3038_gen3_m10	44547	44179.85	665.562
pcb1173_gen3_m10	8953	8894.25	129.79	d1655_gen3_m3	35146	33708.15	169.953	pcb3038_gen3_m2	101395	100463.4	1063.779
pcb1173_gen3_m2	37660	37225.5	156.218	d1655_gen3_m8	3827	3793.05	127.774	pcb3038_gen3_m3	96597	95961.4	777.048
pcb1173_gen3_m3	33994	33435.05	130.341	vm1748_gen1_m10	603	599.65	272.704	pcb3038_gen3_m8	56937	56345.05	562.118
pcb1173_gen3_m8	13463	13405.85	136.932	vm1748_gen1_m2	1230	1221.35	426.532	fl3795_gen1_m10	410	410	793.287
d1291_gen1_m2	598	593.85	136.857	vm1748_gen1_m3	1230	1224	358.198	fl3795_gen1_m2	1989	1917.25	1300.893
d1291_gen1_m3	486	479.4	123.49	vm1748_gen1_m8	793	785.2	268.893	fl3795_gen1_m3	1859	1762.45	1041.273
d1291_gen1_m8	39	39	82.937	vm1748_gen2_m10	32540	31870.35	283.007	fl3795_gen1_m8	565	560.35	724.526
d1291_gen2_m2	31656	31121.05	140.718	vm1748_gen2_m2	65707	65370.4	424.636	fl3795_gen2_m10	24311	23464.4	787.527
d1291_gen2_m3	25419	24897	129.694	vm1748_gen2_m3	65731	65053.8	352.004	fl3795_gen2_m2	106336	105453.9	1042.231
d1291_gen2_m8	2013	2013	82.029	vm1748_gen2_m8	42184	41231	285.625	fl3795_gen2_m3	100818	99137.2	826.789
d1291_gen3_m2	31965	31045.5	119.418	vm1748_gen3_m10	18111	17768.1	259.586	fl3795_gen2_m8	34262	34031.9	619.511
d1291_gen3_m3	23942	22953.6	107.009	vm1748_gen3_m2	69111	68818.05	529.487	fl3795_gen3_m10	7518	7274.7	943.546
d1291_gen3_m8	1166	1166	80.31	vm1748_gen3_m3	66755	66326.15	391.11	fl3795_gen3_m2	77042	75552.25	648.336
rl1304_gen1_m10	463	458.4	196.221	vm1748_gen3_m8	28302	27693.05	298.117	fl3795_gen3_m3	57135	54359.05	783.672
rl1304_gen1_m2	774	759.4	234.356	u1817_gen1_m10	542	536.75	257.277	fl3795_gen3_m8	12286	12083.6	795.271
rl1304_gen1_m3	760	738	186.646	u1817_gen1_m2	919	906.05	303.575	fnl4461_gen1_m10	2138	2119.75	1424.848
rl1304_gen1_m8	537	534.35	172.057	u1817_gen1_m3	890	880.35	262.182	fnl4461_gen1_m2	2414	2391.2	2445.002
rl1304_gen2_m10	24416	24085.4	180.04	u1817_gen1_m8	637	630.35	229.776	fnl4461_gen1_m3	2400	2377.95	2076.616
rl1304_gen2_m2	41120	39766.15	217.164	u1817_gen2_m10	28636	28413.9	253.325	fnl4461_gen1_m8	2232	2214.65	1458.891
rl1304_gen2_m3	39157	38405.6	194.536	u1817_gen2_m2	50796	50100.4	292.257	fnl4461_gen2_m10	123087	121837.7	1240.387
rl1304_gen2_m8	28708	28278.7	175.155	u1817_gen2_m3	49486	48839.3	275.61	fnl4461_gen2_m2	141676	140707.25	1935.007
rl1304_gen3_m10	9502	9373.15	187.925	u1817_gen2_m8	34042	33731.85	247.567	fnl4461_gen2_m3	140947	139495.15	1684.085
rl1304_gen3_m2	39532	38334.65	160.461	u1817_gen3_m10	14054	13650.7	258.418	fnl4461_gen2_m8	129519	128014.85	1303.958
rl1304_gen3_m3	36891	34841.5	185.188	u1817_gen3_m2	57808	57417.85	262.001	fnl4461_gen3_m10	96728	94751.3	1129.311
rl1304_gen3_m8	13141	12831.1	154.157	u1817_gen3_m3	49806	49253.15	224.523	fnl4461_gen3_m2	139885	139169.1	2573.647
rl1323_gen1_m10	352	348	174.686	u1817_gen3_m8	18583	17939.55	238.898	fnl4461_gen3_m3	136181	135203.15	1757.353
rl1323_gen1_m2	769	758.7	196.641	rl1889_gen1_m10	462	452.95	286.04	fnl4461_gen3_m8	110719	109059.65	1046.82
rl1323_gen1_m3	723	707.85	182.443	rl1889_gen1_m2	1138	1114.7	369.783	rl5915_gen1_m10	2733	2690.9	2077.884
rl1323_gen1_m8	457	455.35	182.067	rl1889_gen1_m3	1091	1080.15	355.116	rl5915_gen1_m2	3477	3405.5	3568.756
rl1323_gen2_m10	18296	18128.8	178.89	rl1889_gen1_m8	636	628.55	283.385	rl5915_gen1_m3	3457	3393.9	3050.192
rl1323_gen2_m2	41481	40466.3	202.354	rl1889_gen2_m10	24728	24453.7	290.245	rl5915_gen1_m8	3040	2966.85	2263.107
rl1323_gen2_m3	38791	38536.7	180.256	rl1889_gen2_m2	59936	59136.45	377.037	rl5915_gen2_m10	144675	142173	2110.743
rl1323_gen2_m8	23830	23644	176.487	rl1889_gen2_m3	57521	56929.85	327.585	rl5915_gen2_m2	182690	180305.8	3469.629
rl1323_gen3_m10	7351	7273.65	164.084	rl1889_gen2_m8	33968	33566.95	287.128	rl5915_gen2_m3	182018	178997.85	3010.962
rl1323_gen3_m2	43907	42881.75	197.115	rl1889_gen3_m10	10700	10587.8	271.358	rl5915_gen2_m8	159789	156982.4	2039.985
rl1323_gen3_m3	38072	37123.85	159.29	rl1889_gen3_m2	64900	64271.6	370.993	rl5915_gen3_m10	104482	101300.65	1848.89
rl1323_gen3_m8	11491	11434.65	170.393	rl1889_gen3_m3	58696	58236.1	289.093	rl5915_gen3_m2	198737	195990.35	3600.635
nrw1379_gen1_m10	629	620.35	204.062	rl1889_gen3_m8	18359	17891.75	293.784	rl5915_gen3_m3	195470	192648.05	2873.377
nrw1379_gen1_m2	785	774.05	258.487	d2103_gen1_m10	191	188.6	191.32	rl5915_gen3_m8	140636	135635.05	1861.404
nrw1379_gen1_m3	774	760.05	224.215	d2103_gen1_m2	1017	1001.5	351.286	rl5934_gen1_m10	2370	2314.2	1758.814
nrw1379_gen1_m8	677	671.45	199.835	d2103_gen1_m3	916	889.6	281.483	rl5934_gen1_m2	3363	3276.7	3600.345
nrw1379_gen2_m10	33914	33688.2	217.036	d2103_gen1_m8	323	321.5	224.683	rl5934_gen1_m3	3341	3270.45	2964.668
nrw1379_gen2_m2	45225	45019.25	259.447	d2103_gen2_m10	10548	10303.05	213.89	rl5934_gen1_m8	2642	2580.4	1979.486
nrw1379_gen2_m3	44463	44138.35	220.819	d2103_gen2_m2	56431	53576.65	363.469	rl5934_gen2_m10	122321	120639.2	1739.486
nrw1379_gen2_m8	37552	37259.4	202.48	d2103_gen2_m3	48508	46559.7	295.177	rl5934_gen2_m2	180470	177031.95	3601.828
nrw1379_gen3_m10	15375	15070.9	206.396	d2103_gen2_m8	18498	18355.55	218.114	rl5934_gen2_m3	178631	174893.4	2867.982
nrw1379_gen3_m2	40448	40200.1	223.57	d2103_gen3_m10	5887	5829.6	198.967	rl5934_gen2_m8	138437	135501.75	1913.607
nrw1379_gen3_m3	38276	37916.15	179.829	d2103_gen3_m2	69480	68094.3	369.836	rl5934_gen3_m10	87227	84321.75	1622.863
nrw1379_gen3_m8	19178	18707.95	175.773	d2103_gen3_m3	57293	54366.55	257.5	rl5934_gen3_m2	205202	202564.65	3600.209
fl1400_gen1_m10	103	102.25	117.246	d2103_gen3_m8	10038	9985.7	208.133	rl5934_gen3_m3	197516	195185.2	3062.074
fl1400_gen1_m2	865	864.25	236.562	u2152_gen1_m10	751	746.95	332.637	rl5934_gen3_m8	120936	117290.95	2045.778

Table 22: Results for the set of very large scale instances.

References

L. Accorsi and D. Vigo. A fast and scalable heuristic for the solution of large-scale capacitated vehicle routing problems. Transportation Science, 55:832–856, 2021. doi: 10.1287/trsc.2021.1059.

H. Alkhazaleh, N. Abdallah, A. Ibrahim, M. Habli, and T. Zeki. A scatter search hybrid approach for team orienteering problem. GIS Business, 14:767–782, 12 2019. doi: 10.26643/gis.v14i6.15249.

C. Archetti, A. Hertz, and M. G. Speranza. Metaheuristics for the team orienteering problem. J. Heuristics, 13:49–76, 02 2007. doi: 10.1007/s10732-006-9004-0.

F. Arnold and K. Sörensen. A simple, deterministic, and efficient knowledge-driven heuristic for the vehicle routing problem. Working Papers 2017012, University of Antwerp, Faculty of Business and Economics, December 2017.

F. Arnold and K. Sörensen. What makes a VRP solution good? The generation of problem-specific knowledge for heuristics. Computers & Operations Research, 106:280–288, 2019. ISSN 0305-0548. doi: 10.1016/j.cor.2018.02.007.

F. Arnold, M. Gendreau, and K. Sörensen. Efficiently solving very large-scale routing problems. Computers & Operations Research, 107:32–42, 2019. ISSN 0305-0548. doi: 10.1016/j.cor.2019.03.006.

L. Assunção and G. Mateus. A cutting-plane algorithm for the steiner team orienteering problem. Computers & Industrial Engineering, 135, 06 2019. doi: 10.1016/j.cie.2019.06.051.

B. Bernábe-Loranca, R. Gonzalez-Velázquez, E. Olivares-Benítez, J. Ruiz-Vanoye, and J. Martínez-Flores. Extensions to k-medoids with balance restrictions over the cardinality of the partitions. Journal of Applied Research and Technology, 12 (3):396–408, 2014. ISSN 1665-6423. doi: 10.1016/S1665-6423(14)71621-9.

N. Bianchessi, R. Mansini, and M. G. Speranza. A branch-and-cut algorithm for the team orienteering problem. International Transactions in Operational Research, 25(2):627–635, 2018. doi: 10.1111/itor.12422.

H. Bouly, DC. Dang, and A. Moukrim. A memetic algorithm for the team orienteering problem. In Applications of Evolutionary Computing, pages 649–658, Berlin, Heidelberg, 2008. Springer Berlin Heidelberg. ISBN 978-3-540-78761-7. doi: 10.1007/978-3-540-78761-7_71.

S. Boussier, D. Feillet, and M. Gendreau. An exact algorithm for team orienteering problems. 4OR quarterly journal of the Belgian, French and Italian Operations Research Societies, 5:211–230, 09 2007. doi: 10.1007/s10288-006-0009-1.

O. Bräysy and G. Hasle. Chapter 12: Software Tools and Emerging Technologies for Vehicle Routing and Intermodal Transportation, pages 351–380. 11 2014. ISBN 978-1-61197-358-7. doi: 10.1137/1.9781611973594.ch12.

I. Chao, B. L. Golden, and E. A. Wasil. The team orienteering problem. European Journal of Operational Research, 88(3): 464–474, 1996. ISSN 0377-2217.

J. Christiaens and G. Vanden Berghe. Slack induction by string removals for vehicle routing problems. Transportation Science, 54, 01 2020. doi: 10.1287/trsc.2019.0914.

G. Clarke and J. W. Wright. Scheduling of vehicles from a central depot to a number of delivery points. Operations Research, 12(4):568–581, 1964. doi: 10.1287/opre.12.4.568.

D. Dang, R. Guibadj, and A. Moukrim. An effective pso-inspired algorithm for the team orienteering problem. European Journal of Operational Research, 229:332–344, 09 2013a. doi: 10.1016/j.ejor.2013.02.049.

DC. Dang, R.N. Guibadj, and A. Moukrim. A pso-based memetic algorithm for the team orienteering problem. In Applications of Evolutionary Computation, pages 471–480, Berlin, Heidelberg, 2011. Springer Berlin Heidelberg. ISBN 978-3-642-20520-0. doi: 10.1007/978-3-642-20520-0_48.

DC. Dang, R. El-Hajj, and A. Moukrim. A branch-and-cut algorithm for solving the team orienteering problem. 05 2013b. ISBN 978-3-642-38170-6. doi: 10.1007/978-3-642-38171-3_23.

E. Demir, T. Bektaş, and G. Laporte. An adaptive large neighborhood search heuristic for the pollution-routing problem. European Journal of Operational Research, 223:346–359, 12 2012. doi: 10.1016/j.ejor.2012.06.044.

D. Dumez, F. Lehuédé, and O. Péton. A large neighborhood search approach to the vehicle routing problem with delivery options. Transportation Research Part B: Methodological, 144:103–132, 02 2021. doi: 10.1016/j.trb.2020.11.012.

R. El-Hajj, DC. Dang, and A. Moukrim. Solving the team orienteering problem with cutting planes. Computers & Operations Research, 74, 04 2016. doi: 10.1016/j.cor.2016.04.008.

J. Ferreira, A. Quintas, J. A. Oliveira, G. A. B. Pereira, and L. Dias. Solving the team orienteering problem: Developing a solution tool using a genetic algorithm approach. In Soft Computing in Industrial Applications, pages 365–375, Cham, 2014. Springer International Publishing. ISBN 978-3-319-00930-8. doi: 10.1007/978-3-319-00930-8_32.

M. Fischetti, Salazar G. J. J., and P. Toth. Solving the orienteering problem through branch-and-cut. INFORMS Journal on Computing, 10:133–148, 05 1998. doi: 10.1287/ijoc.10.2.133.

D. Gavalas, C. Konstantopoulos, K. Mastakas, and G. Pantziou. A survey on algorithmic approaches for solving tourist trip design problems. Journal of Heuristics, 20:291–328, 06 2014. doi: 10.1007/s10732-014-9242-5.

B. Golden, L. Levy, and R. Vohra. The orienteering problem. Nav Res Logist, 34:307–318, 06 1987. doi: 10.1002/1520-6750(198706)34:3<307::AID-NAV3220340302>3.0.CO;2-D.

F. Hammami, M. Rekik, and L. C. Coelho. A hybrid adaptive large neighborhood search heuristic for the team orienteering problem. Computers & Operations Research, 123:105034, 2020. ISSN 0305-0548. doi: 10.1016/j.cor.2020.105034.

L. Kaufman and P. J. Rousseeuw. Partitioning Around Medoids (Program PAM), chapter 2, pages 68–125. John Wiley & Sons, Ltd, 1990. ISBN 9780470316801. doi: 10.1002/9780470316801.ch2.

L. Ke, C. Archetti, and Z. Feng. Ants can solve the team orienteering problem. Computers & Industrial Engineering, 54: 648–665, 04 2008. doi: 10.1016/j.cie.2007.10.001.

L. Ke, L. Zhai, J. Li, and F. T.S. Chan. Pareto mimic algorithm: An approach to the team orienteering problem. Omega, 61: 155–166, 2016. ISSN 0305-0483. doi: 10.1016/j.omega.2015.08.003.

M. Keshtkaran, K. Ziarati, A. Bettinelli, and D. Vigo. Enhanced exact solution methods for the team orienteering problem. International Journal of Production Research, ahead-of-print:1–11, 07 2015. doi: 10.1080/00207543.2015.1058982.

BI. Kim, H. Li, and A. Johnson. An augmented large neighborhood search method for solving the team orienteering problem. Expert Systems with Applications, 40:3065–3072, 06 2013. doi: 10.1016/j.eswa.2012.12.022.

S. Kirkpatrick, C. Gelatt, and M. Vecchi. Optimization by simulated annealing. Science (New York, N.Y.), 220:671–80, 06 1983. doi: 10.1126/science.220.4598.671.

G. Kobeaga, M. Merino, and J. Lozano. An efficient evolutionary algorithm for the orienteering problem. Computers & Operations Research, 90, 09 2017. doi: 10.1016/j.cor.2017.09.003.

J. Kytöjoki, T. Nuortio, O. Bräysy, and M. Gendreau. An efficient variable neighborhood search heuristic for very large scale vehicle routing problems. Computers & Operations Research, 34:2743–2757, 09 2007. doi: 10.1016/j.cor.2005.10.010.

G. Laporte and S. Martello. The selective travelling salesman problem. Discrete Applied Mathematics, 26(2):193–207, 1990. ISSN 0166-218X. doi: 10.1016/0166-218X(90)90100-Q.

SW. Lin. Solving the team orienteering problem using effective multi-start simulated annealing. Applied Soft Computing, 13: 1064–1073, 02 2013. doi: 10.1016/j.asoc.2012.09.022.

S.T. Windras Mara, R. Norcahyo, P. Jodiawan, L. Lusiantoro, and A. P. Rifai. A survey of adaptive large neighborhood search algorithms and applications. Computers & Operations Research, 146:105903, 2022. ISSN 0305-0548. doi: 10.1016/j.cor.2022.105903.

R. Mariescu-Istodor. Solving the large-scale tsp problem in 1 h: Santa claus challenge 2020. Frontiers in Robotics and AI, 8, 10 2021. doi: 10.3389/frobt.2021.689908.

C. Orlis, N. Bianchessi, R. Roberti, and W. Dullaert. The team orienteering problem with overlaps: An application in cash logistics. Transportation Science, 54:470–487, 03 2020. doi: 10.1287/trsc.2019.0923.

A. Pessoa, R. Sadykov, E. Uchoa, and F. Vanderbeck. A Generic Exact Solver for Vehicle Routing and Related Problems, pages 354–369. 04 2019. ISBN 978-3-030-17952-6. doi: 10.1007/978-3-030-17953-3_27.

M. Poggi, H. Viana, and E. Uchoa. The team orienteering problem: Formulations and branch-cut and price. volume 14, pages 142–155, 01 2010. doi: 10.4230/OASIcs.ATMOS.2010.142.

S. Ropke and D. Pisinger. An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transportation Science, 40:455–472, 11 2006. doi: 10.1287/trsc.1050.0135.

J. Ruiz-Meza and J. R. Montoya-Torres. A systematic literature review for the tourist trip design problem: Extensions, solution techniques and future research lines. Operations Research Perspectives, 9:100228, 2022. ISSN 2214-7160. doi: 10.1016/j.orp.2022.100228.

P. Shaw. Using constraint programming and local search methods to solve vehicle routing problems. In Michael Maher and Jean-Francois Puget, editors, Principles and Practice of Constraint Programming — CP98, pages 417–431, Berlin, Heidelberg, 1998. Springer Berlin Heidelberg. ISBN 978-3-540-49481-2. doi: 10.1007/3-540-49481-2_30.

W. Souffriau, P. Vansteenwegen, and G. Vanden Berghe. A path relinking approach for the team orienteering problem. Computers & Operations Research, 37:1853–1859, 11 2010. doi: 10.1016/j.cor.2009.05.002.

É. Taillard and K. Helsgaun. Popmusic for the travelling salesman problem. European Journal of Operational Research, 272, 06 2018. doi: 10.1016/j.ejor.2018.06.039.

H. Tang and E. Miller-Hooks. A tabu search heuristic for the team orienteering problem. Computers & Operations Research, 32:1379–1407, 06 2005. doi: 10.1016/j.cor.2003.11.008.

P. Toth. and D. Vigo. The granular tabu search and its application to the vehicle-routing problem. INFORMS Journal on Computing, 15:333–346, 11 2003. doi: 10.1287/ijoc.15.4.333.24890.

E. Tsakirakis, M. Marinaki, Y. Marinakis, and N. Matsatsinis. A similarity hybrid harmony search algorithm for the team orienteering problem. Applied Soft Computing, 80, 05 2019. doi: 10.1016/j.asoc.2019.04.038.

T. Tsiligiridis. Heuristic methods applied to orienteering. Journal of the Operational Research Society, 35:797–809, 09 1984. doi: 10.1057/jors.1984.162.

R. Turkeš, K. Sörensen, and L. M. Hvattum. Meta-analysis of metaheuristics: Quantifying the effect of adaptiveness in adaptive large neighborhood search. European Journal of Operational Research, 292(2):423–442, 2021. ISSN 0377-2217. doi: 10.1016/j.ejor.2020.10.045.

P. Vansteenwegen. The mobile tourist guide: An or opportunity. OR Insight, 20:21–27, 07 2007. doi: 10.1057/ori.2007.17.

P. Vansteenwegen, W. Souffriau, G. Vanden Berghe, and D. Van Oudheusden. A guided local search metaheuristic for the team orienteering problem. European Journal of Operational Research, 196(1):118–127, 2009a. ISSN 0377-2217. doi: 10.1016/j.ejor.2008.02.037.

P. Vansteenwegen, W. Souffriau, and G. Vanden Berghe. Metaheuristics for Tourist Trip Planning, volume 624, pages 15–31. 05 2009b. ISBN 978-3-642-00938-9. doi: 10.1007/978-3-642-00939-6_2.

T. Vidal, T. G. Crainic, M. Gendreau, and C. Prins. A unified solution framework for multi-attribute vehicle routing problems. European Journal of Operational Research, 234:658–673, 05 2014a. doi: 10.1016/j.ejor.2013.09.045.

T. Vidal, N. Maculan, P. Penna, and L. Ochi. Large neighborhoods with implicit customer selection for vehicle routing problems with profits. Transportation Science, 50, 01 2014b. doi: 10.1287/trsc.2015.0584.

E. Zachariadis and C. Kiranoudis. A strategy for reducing the computational complexity of local search-based methods for the vehicle routing problem. Computers & Operations Research, 37:2089–2105, 12 2010. doi: 10.1016/j.cor.2010.02.009.