Contents lists available at ScienceDirect

Computers and Operations Research

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

A cooperative team orienteering optimisation model and a customised resolution metaheuristic

Andrea Bendazzoli, Michele Urbani [*], Matteo Brunelli, Francesco Pilati

Department of Industrial Engineering, University of Trento, via Sommarive 9, Trento, 38123, Italy

ARTICLE INFO

Keywords:
Workforce routing
Orienteering problem
Operator synchronisation
Metaheuristics
Service industry

ABSTRACT

Workforce routing optimisation is an essential management task to achieve customer satisfaction and minimise costs in service-providing companies. A typical problem for an energy-saving company (ESCo) is to optimise the provisioning of maintenance services for the contracted buildings through a fleet of heterogeneous maintenance staff. This problem is modelled as a cooperative orienteering problem with time windows, operator qualification, and synchronisation constraints. A novel insertion heuristic is proposed and embedded in an adaptive large neighbourhood search algorithm and it is tested against a state-of-the-art algorithm using real-world data. The comparative study demonstrates the potential of the heuristic and a sensitivity analysis shows its robustness, focusing on time window length variation, and the number of synchronisation requirements. We consider managerial insights supported by results and concerns, e.g., the employment of further technicians to increase the number of served facilities. The proposed model and algorithm apply to similar problems as well.

1. Introduction

According to the World Bank, the service industry has steadily grown during the last century and currently accounts for up to 70% of the GDP in developed countries (Bank, 2021). Service providers rely heavily on modern management tools, are information technology-intensive (Wu et al., 2016), and the efficiency of operations and the quality of service are of the utmost importance to achieve customer loyalty (Yee et al., 2010) and to start long-lasting relationships that generate revenues. In the field of energy management, *Energy Saving Companies* (ESCOs) are service contractors that provide energy savings and other benefits to their customers (Larsen et al., 2012). In the ESCO industry, performance contracting generates about 70% of revenues (Stuart et al., 2016) and operations management is a core part of ESCO's business model. The activities of a typical ESCO include developing, installing, and financing performance-based projects aimed at improving energy efficiency and reducing the load on facilities owned by third parties (Vine, 2005). Being energy optimisation at the top of the agenda of several governments and private companies, the estimated business potential for ESCOs is in the order of billions of revenues every year worldwide (Larsen et al., 2012; Vine, 2005; Bertoldi et al., 2006).

To generate revenues, ESCOs have to show excellent abilities in operations management. A contract with an ESCO usually lasts from 5 to 10 years (Vine, 2005): after an initial phase during which the buildings under contract may undergo renovation, an ESCO aims at generating revenues by efficiently managing a facility. Maintenance of contracted buildings is of primary importance because it keeps efficiency high and enhances the perceived quality of service (Bertoldi et al., 2003). In addition to reacting to maintenance requests and complying with regulations, a good operational model should not neglect the workforce needs: well-organised tours of customers' visits may increase the perceived efficiency of the work schedule and, consequently, increase worker satisfaction. No less important is the energy efficiency of the ESCO itself: an operational model that enables shorter trips contributes to reducing the cost of operations due to the misuse of vehicles, thus also reducing the CO_2 footprint.

Problem statement

In the described scenario, maintenance interventions performed by ESCOs present a twofold nature: they can be corrective, i.e., triggered by a customer call, or preventive, scheduled in advance. Corrective maintenance interventions are managed through a ticketing system that establishes the priority of a contingency call and assigns the task directly to an operator, who intervenes quickly. Preventive maintenance requirements are defined by the local regulations and plant manufacturer indications; missing an intervention prescribed by the regulation may trigger sanctions. Preventive maintenance is required

* Corresponding author.
E-mail address: michele.urbani@unitn.it (M. Urbani).

https://doi.org/10.1016/j.cor.2023.106488
Received 3 April 2023; Received in revised form 19 October 2023; Accepted 22 November 2023
Available online 28 November 2023

cyclically with different frequencies depending on the task type: the importance of a task is usually inversely proportional to the distance from the due date and is traditionally lower than the priority assigned to a corrective task. A *rolling horizon* approach is usually adopted to schedule the work: the work schedule is drafted daily based on the set of activities to be carried out over the following weeks.

Concerning the execution of maintenance activities, there are qualification requirements to carry out maintenance: a maintenance task can either be executed by any worker or may require specific skills to be carried out. For instance, there may be tasks that require knowledge of thermal plants, whereas others cannot be carried out without electrical skills. The set of required skills may be large and a single operator may have been trained to acquire only a subset of these. Moreover, more than one operator may be required simultaneously with other colleagues to execute a maintenance intervention and any number of skills may be necessary for a specific task. Both operators and buildings are available within specific time windows: e.g., buildings may be surgery rooms, which are not accessible during specific time slots, and operators may be unavailable due to illness or holiday. Finally, operators usually travel in vans equipped with the tools and parts required for the majority of interventions and they are allowed to depart from and return to their homes.

The contribution of this paper resides in the formalisation of the problem using a mathematical model to optimise the work schedule of a team of maintenance operators considering all the features mentioned above. Operators' skills, the departure from and return to different depots within operators' work time, and synchronisation at specific buildings to carry out maintenance interventions within given time windows are considered. Since operations are scheduled by adopting a rolling horizon approach, we propose the *orienteering problem* formulation to provide flexibility in managing activities with different priorities. An adaptive large neighbourhood search (ALNS) algorithm is supplemented with a novel *best insertion* procedure to efficiently find solutions with synchronisations for the targeted problem. The proposed algorithm is distinguished by adaptability to different problem instances, which may vary in size, i.e., in the number of nodes in the graph or complexity, e.g., due to the number of synchronisations required. The algorithm is tested using real-world activities, buildings, and operator data, and its efficiency is shown through extensive numerical experiments against a state-of-the-art algorithm. The proposed numerical study aims to show that there is a gap between our algorithm and the other metaheuristics.

The remainder of the paper is organised as follows: Section 2 guides the reader through the meaningful contributions already present in the literature about the target topic. Section 3 formalises the model, and Section 4 describes the proposed novel best insertion heuristic and the algorithm workflow. Section 5 presents an overview of a real-world case study, followed by numerical experiments, results and discussion. Section 6 ends the paper with conclusions.

2. Literature review

The orienteering problem (OP) is an $\mathcal{NP}$-hard combinatorial problem (Golden et al., 1987) that was proposed for the first time by Tsiligirides (1984). It can be thought of as a combination of a Knapsack Problem and a Travelling Salesmen Problem and for this reason, it is also known as the Selective Travelling Salesman Problem (Laporte and Martello, 1990). The orienteering problem finds a real-world application for, e.g., the tourist trip design, whereby tourists want to visit the most interesting attractions within an available time (Borràs et al., 2014; Vansteenwegen and Van Oudheusden, 2007). An extension of the OP is the *team orienteering problem* (TOP) (Chao et al., 1996), where the goal is to design multiple trips of maximum length T_{max} that maximise the total collected reward. A prototypical application of the TOP concerns the routing of technicians to service customers (Tang and Miller-Hooks, 2005): all technicians have a limited number of hours during a workday and only a subset of customers can be served. The TOP has been successfully used also to optimise the tours of field sales forces (Meyer et al., 2021). The importance of visiting a customer is reflected by the score assigned to the visit and the objective is to select the subset of nodes that maximises the total score. In several fields of application, e.g., in technician routing, a further requirement for the OP and TOP is the addition of time windows for visiting customers. The extensions of the OP and TOP to time windows are known as the orienteering problem with time windows (OPTW) (Kantor and Rosenwein, 1992) and the team orienteering problem with time windows (TOPTW) (Tricoire et al., 2010; Vansteenwegen et al., 2009), respectively. In the OPTW and TOPTW, a customer can be visited within a time window: usually, a resource can arrive earlier than the customer is available and then wait but it cannot start serving the customer after the time window ends.

A comprehensive review of applications, resolution methods, and benchmark instances for the OP, TOP, and TOPTW up to the year 2010 was provided by Vansteenwegen et al. (2011). Gunawan et al. (2016) updated Vansteenwegen et al.'s review with works up to the year 2016 and extended the review with further variants and applications of the OP, i.e., the Stochastic OP, the Generalised OP, the Arc OP (Souffriau et al., 2011), the Multiagent OP, the Clustered OP, and others.

A recent variant of the OP is the *cooperative orienteering problem* with time windows (COPTW), which was proposed for the first time by Van Der Merwe et al. (2014). In addition to the features listed for the TOPTW, the COPTW models operator qualification requirements and synchronisation constraints, i.e., the presence of more than one operator at the same time and place. According to Soares et al. (2023), there are two types of possible synchronisation, that is operations and movement synchronisation; all the contributions mentioned in the following belong to the class of operations synchronisation.

The asset protection problem (APP) was the first to be modelled as a COPTW by Van Der Merwe et al. (2014) and since then, it has drawn the attention of researchers. The objective of the APP is to secure a set of community assets (i.e., schools, bridges, factories, hospitals, etc.), each associated with a score that maximises the total reward by employing a limited number of firefighter teams. Time window constraints are fundamental to ensure that the securing of assets occurs at a suitable time to escape the wildfire front, whereas assets' scores are assigned to steer securing the overall best combination of assets. The cooperation requirement may be necessary to secure, e.g., assets extended over a large area, or where resources with different skills are required simultaneously. In Van Der Merwe et al. (2014), the authors solved a case study using a commercial solver, which optimally solved small instances (20 nodes) in a few seconds. The resolution time and quality for instances with 100 nodes varied widely depending on instance parameters, thus limiting the use of solvers for operational purposes.

To tackle the computational challenges posed by the cooperative aspect of the COPTW, Roozbeh et al. (2016) proposed a resolution heuristic named *modified Clarke and Wright* heuristic. Roozbeh et al.'s proposal improved the results previously obtained in the literature, still, it presented some limitations to the general algorithmic framework within which it was used. The computational limitations highlighted in Van Der Merwe et al. (2014) were overcome by Roozbeh et al. (2018), who developed an ALNS algorithm with problem-specific adaptations. Roozbeh et al.'s algorithm was shown to be faster than commercial solvers with small problem instances and produced suitable solutions to large instances in an appropriate time for operational purposes. Furthermore, Roozbeh et al. (2018) proposed a set of modified benchmark instances, which they used for numerical experiments. Nuraiman et al. (2020) also addressed the APP and proposed further improvements both to the COPTW model and to the algorithm developed in Roozbeh et al. (2018). Due to the need to escape from the wildfire front, Nuraiman et al. modelled the problem by introducing a single departure location and multiple arrival locations far from the departure. By exploiting the sequentiality of time windows, the authors developed a two-step heuristic that produced better results than those in Roozbeh et al. (2018). First, spatial decomposition is applied to partition the set of interest points, and, secondly, the problem is solved for each subset of nodes sequentially; the stages close to the wildfire front are optimised first, and then the last nodes serviced in the previous stage become the starting points for the next stage.

The APP problem was further studied by Yahiaoui et al. (2023), who advocated for the need for synchronised visits to an asset, i.e., that multiple operators are present at the same place and time to start an activity. Yahiaoui et al. called the problem Synchronised Team Orienteering Problem with Time Windows (STOPTW). To solve the STOPTW, Yahiaoui et al. (2023) proposed a Greedy Randomised Adaptive Search Procedure coupled with an Iterated Local Search and a post-optimisation procedure. The algorithm improved all the medium and large (100 nodes) instances proposed by Roozbeh et al. (2018) and was tested on large instances with up to 200 nodes.

Garcia (2022) addressed the limitation of having a single departure point by adding multiple departure points to the model by Yahiaoui et al. (2023). The author renamed the problem as the Synchronised Multi-Assignment Orienteering problem. Garcia developed his version of the ALNS algorithm to solve the problem and found 24 new best solutions to the COPTW instances proposed by Roozbeh et al. (2018). The paper addresses no real-world application. A generic solution approach to the COPTW is developed by Roozbeh et al. (2020), who improved the merit-based heuristic that they proposed in Roozbeh et al. (2018). The authors presented a hypothetical application of the COPTW to the maintenance service problem, where a set of independent workers should maximise the score collected by servicing a subset of maintenance sites among those with a close due date. Through an extensive numerical study, the proposed algorithm showed to exceed, on average, the performance of the previous ones (Roozbeh et al., 2018).

Metaheuristics embedding low-level heuristics based on local and neighbourhood search are the standard resolution approach for the OP class. The design of metaheuristics for the OP is difficult because “the score of a vertex and the time to reach the vertex are independent and often contradictory” (Gendreau et al., 1998). Unfortunately, the possibility to use heuristics from the VRP field, such as those in Drexl (2012), is limited and OP-specific ones should be preferred (Vansteenwegen et al., 2011). Some examples of heuristics for the TOPTW were provided by Gambardella et al. (2012), Ke et al. (2016), Lin and Yu (2012) and Vansteenwegen et al. (2009), just to mention a few. Concerning the COPTW, the design of specific low-level heuristics to tackle synchronisation seems promising. Afifi et al. (2016) presented the best insertion algorithm, which helps to speed up the search by producing produce feasible solutions to the VRPMS. By the same token, Parragh and Doerner (2018) proposed an ALNS algorithm that incorporates synchronisation-specific heuristics to solve the VRP with synchronisations and achieved better results than in Afifi et al. (2016). In the papers analysed above, the best insertion procedure of Afifi et al. (2016) is often used. However, the efforts to develop OP-specific synchronisation heuristics seem limited. Although the computational tools for the COPTW have been studied and improved to a certain extent, the range of applications of the COPTW (or STOPTW) remains rather limited. Compared to the COPTW (or STOPTW), the VRPMS boasts a wide range of documented applications (Drexl, 2012).

Table 1 characterises the contributions of previously published works with respect to the following concepts: the modelling of multiple depots (MD), which indicates that multiple departure and arrival nodes, ideally, one per operator, are present in the paper; the presence of hard time windows (TW) within which a building can be visited; synchronisation (S) of multiple operators at the same place and time; and the presence of constraints to manage operator qualification (OQ). The column “features” characterises a literature contribution to a specific aspect, which may be the optimisation model or the resolution algorithm; the meaning of a feature is reported in the legend of Table 1. The column “case study” in Table 1 lists the proposed field of application that the authors envisioned for their models and algorithms.

3. Mathematical model

The problem tackled in this research is a multi-depot team maintenance orienteering problem with time windows, synchronisation of operators, and operator qualification constraints. A mathematical model of the problem is formulated to assign a set of maintenance operators K to a subset of the tasks V to be carried out, where each task is associated with a building. Maintenance operators are trained to acquire a specific skill $f \in F = \{1,\dots,Q\}$, which makes them eligible to carry out specialised tasks. The work day of an operator $k \in K$ is modelled by a time window $[o_k, c_k]$ and, by the same token, a site $v \in V$ can be visited within an interval $[o_v, c_v]$. A time window $[o_i, c_i]$ can be used to model the unavailability of a specific building, e.g., a meeting room or a surgery room, or to model the unavailability of an operator, who may be on sick leave or on holiday. Operators depart from and return to their houses, D_d and D_a respectively; this modelling choice enables using departure points different from arrival points. Depots $D = D_d \cup D_a$ and tasks V are the nodes $N = D \cup V$ of an oriented graph $G = (N, A)$ that models the network of connections among sites and depots, where $A = \{(i,j) | i,j \in N, i \neq j, i \notin D_d, j \notin D_a\}$. The travel time between each couple of nodes (i,j) is t_{ij}, whereas the service time required by the ith task is t_i^{ser}. To model synchronised activities, a partition VS of the set of tasks V defines which tasks must be carried out simultaneously by several operators $|v|, v \in VS$. An element of VS identifies multiple nodes in the graph, each corresponding to a task that requires a specific skill. In turn, a subset $VS^C \subseteq VS$ identifies the so-called *synchronisations*, which require the simultaneous presence of multiple operators at the same site.

The execution of a task is modelled employing a binary decision variable y_i, whose value is 1 if the required number of operators carry out the task according to synchronisation needs, and 0 otherwise. Variables y_i are stored in the array Y. The passage of an operator k through an arc (i,j) is modelled by the variables x_{ij}^k, which is equal to 1 if the arc is visited, and 0 otherwise. Modelling the passage through an arc using the three-index variable x_{ij}^k allows us to recognise straightforwardly the tours of operators, i.e., we used a commodity-flow formulation of our transportation problem. The variables x_{ij}^k are collected in the array X. The start time of the ith task is modelled by the continuous variable s_i and all start times are collected in the array S.

The model of the problem is the following.

$$\text{maximise } Z = \sum_{v \in VS} \left(y_v \sum_{i \in v} p_i \right) \tag{1}$$

$$\text{s.t.} \quad \sum_{k \in K} \sum_{j|(i,j) \in A} x_{ij}^k \leq 1 \qquad \forall i \in V; \tag{2}$$

$$\sum_{j|(i,j) \in A, j \in V} x_{ij}^k - \sum_{j|(i,j) \in A, j \in V} x_{ji}^k = 0 \qquad \forall k \in K, \forall i \in V; \tag{3}$$

$$\sum_{j \in V} x_{d_k,j}^k = 1 \qquad \forall k \in K, d_k \in D_d; \tag{4}$$

$$\sum_{j \in V} x_{j,d_k}^k = 1 \qquad \forall k \in K, d_k \in D_a; \tag{5}$$

$$\sum_{i \in v} \sum_{j \in V \setminus v} \sum_{k \in K} x_{ij}^k = y_v |v| \qquad \forall v \in VS; \tag{6}$$

$$s_i + t_{ij} + t_i^{ser} - M(1 - x_{ij}^k) \leq s_j \qquad \forall k \in K, \forall (i,j) \in A; \tag{7}$$

$$o_i \leq s_i \leq c_i \qquad \forall i \in N; \tag{8}$$

$$s_i = s_j \qquad \forall \{i,j\} \subseteq v \in VS^C; \tag{9}$$

$$q_{if} \sum_{j|(i,j) \in A} x_{ij}^k \leq w_f^k \qquad \forall i \in V, k \in K, f \in F; \tag{10}$$

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

$$y_v \in \{0,1\} \qquad \forall v \in VS; \tag{12}$$

$$s_i \geq 0 \qquad \forall i \in N. \tag{13}$$

Table 1
Overview of features addressed in the literature.

	Formulation	Features	MD	TW	S	OQ	Case study
Van Der Merwe et al. (2014)	COPTW	A		✓	✓	✓	APP
Roozbeh et al. (2016)	COPTW	B		✓	✓		APP
Roozbeh et al. (2018)	COPTW	C		✓	✓	✓	APP
Roozbeh et al. (2020)	COPTW			✓	✓	✓	Maintenance
Nuraiman et al. (2020)	COPTW	D		✓	✓	✓	APP
Yahiaoui et al. (2023)	STOPTW			✓	✓	✓	APP
Garcia (2022)	SMOP (COPTW)		✓	✓	✓	✓	
Our paper	COPTW		✓	✓	✓	✓	Maintenance

Legend: MD: multi-depot, TW: time-windows, S: synchronisation, OQ: operator qualification, APP: asset protection problem, A: synchronisation is not hard, B: modified Clarke–Wright heuristic, C: Merit-based heuristic, D: spatial-decomposition heuristic.

The objective function in Eq. (1) concerns the maximisation of the sum of scores obtained by carrying out a maintenance task. A score p_i is assigned to each task and it reflects the urgency of the task. The family of constraints in Eq. (2) ensures that an arc is visited only once during the schedule. The constraint in Eq. (3) imposes that when operators arrive at node i they also depart from the same node. Departure and arrival of operators at/to their depot are ensured by the constraints in Eqs. (4) and (5). To complete a synchronised activity, the operators must visit the building simultaneously; the constraint in Eq. (6) ensures that such a requirement is satisfied. Enough time between two tasks is ensured by the constraint in Eq. (7), where t_{ij} is the travel time between building i and j, t_i^{ser} is the service time required by task i, and M is a large number usually greater than c_i. The work time of operators and the opening time of the buildings to be visited is respected by imposing the constraint in Eq. (8), whereas the arrival of two operators at the same site to carry out a synchronised activity is imposed by Eq. (9). The constraint in Eq. (9) is necessary to impose the synchronisation of all pairs of operators in the same set $v \in VS^C$, see, e.g., Parragh and Doerner (2018); the number of constraints defined by Eq. (9) grows exponentially with the cardinality of a set $v \in VS^C$, and it might not be efficient when a high number of resources must be synchronised. The requirement for a specific skill to carry out an activity is modelled by Eq. (10), where the parameter q_{if} is 1 if the skill f is required by task i else it is 0, and w_f^k is equal to 1 if operator k owns the skill f, otherwise 0. The domains of variables x_{ij}^k, y_i, and s_i are declared in Eqs. (11), (12), and (13), respectively.

4. Metaheuristic algorithm

This section presents a customised version of the adaptive large neighbourhood search (ALNS) algorithm to solve the COPTW, together with a novel procedure for the insertion of synchronised visits in the maintenance plan. The algorithmic procedure is a modified version of the ALNS proposed in Roozbeh et al. (2020), which is used as a benchmark to test the performance of the proposed algorithm. In an ALNS algorithm, a set of heuristics can be used to modify the solution and the algorithm can learn which are the most effective to select and apply them more often. The heuristics used in the proposed algorithm belong to three groups, i.e., insertion, removal, and local search heuristics.

4.1. Insertion algorithms

Insertion algorithms, or heuristics, modify incomplete solutions to explore new regions of the objective function. A few principles from the VRP with synchronisation literature (Drexl, 2012) can be exploited to steer the insertion process: the distance or the travel time between nodes, the number of resources required at a specific site, the width of the time window, and the relative geographical position of two subsequent visits with respect to the operator's depot.

Routing problems with synchronisations are complex to solve because vehicles collect the reward cooperatively, namely, it is not possible to calculate the contribution of a vehicle to the total objective value (Van Der Merwe et al., 2014). A constant time algorithm was proposed by Afifi et al. (2016) to check the insertion feasibility of a synchronised visit; for the sake of brevity, the whole procedure is reported in Appendix A.

4.1.1. Best insertion procedure

The *BestInsertion* is a novel algorithm specifically designed for the insertion of visits in the COPTWs as formulated in Section 3. The algorithm requires as input a solution to the problem, which might be empty or partially filled with visits, and returns a (possibly) improved solution.

Solutions showing the highest "score over insertion time" ratio (Kantor and Rosenwein, 1992) are prioritised for insertion. In Kantor and Rosenwein (1992), the definition of insertion cost is $t_{k(r-1),k(r)} + t_{k(r),k(r+1)} - t_{k(r-1),k(r+1)}$, where r is the position of a visit along a route, and $k(r)$ is the node id corresponding to the visit in position r. Differently from previous insertion heuristics, the *BestInsertion* considers not only the insertion time but also the time required to complete a maintenance operation t_i^{ser}. The insertion quality ψ of a node i in position r along route k is measured by the function

$$\psi(i,r,k) = p_i / \left(t_{k(r-1),k(r)} + t_{k(r),k(r+1)} - t_{k(r-1),k(r+1)} + t_i^{ser}\right), \tag{14}$$

where p_i is the priority of the ith activity.

Let the set of visits that have not been added to a solution be V_u. If all routes are empty, the procedure loops over routes $k \in K$ to insert a visit (lines 3–7 in Alg. 1). For each route $k \in K$, the non-synchronised activity showing the highest score $i^* = \arg\max_{i\in V_u, i\notin v\in VS^C}\{\psi(i,1,k)\}$ is added to route k, and the visit i^* is removed from V_u.

The body of the *BestInsertion* procedure loops over until there is no possibility of adding activities to the plan. At the beginning of each iteration, a route $k \in K$ is selected according to a discrete uniform distribution over routes (line 8 in Alg. 1). The insertion scores $\psi(i,r,k)$ of all possible insertion positions r of visits $i \in V_u$ along route k are evaluated. At this point, checking the feasibility of all visits that require synchronisation is computationally expensive and unnecessary. Therefore, to limit the computational effort, the visits in a set $v \in VS^C$ are momentaneously considered non-synchronised, and their feasibility is checked accordingly. At lines 9 and 10 in Alg. 1, the best score among non-synchronised visits

$$\psi^* = \max_{i\in V_u, i\notin v\in VS^C}\left\{\max_r \psi(i,r,k)\right\},$$

and the best score among visits that require synchronisation

$$\psi^*_{synch} = \max_{i\in V_u, i\in v\in VS^C}\left\{\max_r \psi(i,r,k)\right\}$$

are calculated. The partitioning of visits based on the synchronisation requirement is fundamental to balance the selection of synchronised visits and the speed reduction due to checking their feasibility.

If $\psi^*_{synch} < \psi^*$, the non-synchronised visit

$$i^* = \underset{i\in V_u, i\notin v\in VS^C}{\arg\max}\left\{\max_r \psi(i,r,k)\right\}$$

Algorithm 1: The *BestInsertion* procedure.

```
Data: The set of visits $V_u$, the set of routes $K$, a solution $Q = \langle X, Y, S \rangle$
1  $flag \leftarrow 0$;
2  while $flag = 0$ do
3      for $k \in K$ do
4          if route $k$ is empty then
5              $i^* \leftarrow \arg\max_{i \in V_u} \{\psi(i, 1, k)\}$;
6              Update $X, Y, S$;
7              $V_u \leftarrow V_u \setminus \{i^*\}$;
8      $k \sim$ Unif$(K)$ ;
9      $\psi^* \leftarrow \max_{i \in V_u, i \notin v \in VS^C} \{\max_r \psi(i, r, k)\}$;
10     $\psi^*_{synch} \leftarrow \max_{i \in V_u, i \in v \in VS^C} \{\max_r \psi(i, r, k)\}$;
11     if $\psi^*_{synch} < \psi^*$ then
12         $i^* \leftarrow \arg\max_{i \in V_u, i \notin v \in VS^C} \{\max_r \psi(i, r, k)\}$;
13         Update $X, Y, S$;
14         $V_u \leftarrow V_u \setminus \{i^*\}$;
15     else if $\psi^* < \psi^*_{synch}$ then
16         $i^*_{synch} \leftarrow \arg\max_{i \in V_u, i \in v \in VS^C} \{\max_r \psi(i, r, k)\}$;
17         $r^* \leftarrow \arg\max_r \psi(i^*_{synch}, r, k)$;
18         for $j | \{i^*_{synch}, j\} \subseteq v \in VS^C$ do
19             $k' \leftarrow \arg\min_{k' \in K \setminus k} \{\min_r wait^j_{k'(r)}\}$;
20             if insertion of $j$ is infeasible then
21                 go to line 12;
22             $V_u \leftarrow V_u \setminus \{j\}$;
23         $V_u \leftarrow V_u \setminus \{i^*\}$;
24         Update $X, Y, S$;
25     if $\nexists$ feasible insertion of $i \in V_u$ then
26         $flag \leftarrow 1$;
```

is added to route k at the best available position (lines 11–15 in Alg. 1). Contrarily, if $\psi^* < \psi^*_{synch}$, the highest-score visit that require synchronisation

$$i^*_{synch} = \arg\max_{i \in V_u, i \in v \in VS^C} \left\{ \max_r \psi(i, r, k) \right\}$$

is tentatively added at position $r^* = \arg\max_r \psi(i^*_{synch}, r, k)$ along route k. The insertion feasibility of visits $j | \{i^*_{synch}, j\} \subseteq v$ is checked according to the procedure in Appendix A. Each of the $j | \{i^*_{synch}, j\} \subseteq v$ synchronised activities is tentatively inserted in the route that enables the synchronisation to occur and that minimises the waiting time $wait^j_{k'(r)}$ of the k' operator. If the insertion with the minimum waiting time is feasible for all activities $j | \{i^*_{synch}, j\} \subseteq v$, the solution is updated and the activities just added are removed from V_u. If the insertion of the synchronised activity i^*_{synch} and all its counterparts is not feasible, the *BestInsertion* restarts from line 12 in Alg. 1 and inserts the non-synchronised visit i^*.

4.2. Removal and local-search algorithms

The low-level heuristics concerning the removal of visits from a schedule have been reproduced from previous works with a few adaptations to fit the needs of the proposed algorithm. Local search heuristics, also known as perturbation heuristics, are specific combinations of removal and insertion operations, which cannot be obtained by combining two of the heuristics mentioned before. For brevity, removal and local search heuristics are summarised in Appendices B and C, respectively.

4.3. A resolution metaheuristic

The proposed resolution metaheuristic is a modified version of the adaptive large neighbourhood search (ALNS) algorithm proposed in Roozbeh et al. (2020).

Algorithm 2: The modified ALNS algorithm.

```
Data: The start $T_{start}$ and end temperature $T_{end}$; the cooling coefficient
      $\alpha$; the sets of removal $H_d$ and local search $H_{ls}$ heuristics; the
      parameters $D_{min}$ and $D_{max}$, $it_{new}$, $it_{upd}$, and $it_{res}$.
1  $Q_0 \leftarrow BestInsertion()$;
2  $Q_{best} \leftarrow Q_0$;
3  $\rho^d \leftarrow \mathbf{1}$, $\rho^{ls} \leftarrow \mathbf{1}$;
4  $T \leftarrow T_{start}$;
5  $iter, i \leftarrow 0, 0$;
6  while $T \geq T_{end}$ do
7      $i \leftarrow i + 1$;
8      $n \leftarrow \sum_{y \in Y} y$;
9      $h_d, h_{ls} \sim \mathbf{p}(H_d), \mathbf{p}(H_{ls})$;
10     if $iter \leq it_{new}$ then
11         $d_{min} \leftarrow n \cdot D_{min}$, $d_{max} \leftarrow n \cdot D_{max}$;
12     else
13         $d_{min} \leftarrow 2n \cdot D_{min}$, $d_{max} \leftarrow 2n \cdot D_{max}$;
14     $Q_0 \leftarrow h_d(Q_1, d_{min}, d_{max})$;
15     $Q_1 \leftarrow h_{ls}(Q_1)$;
16     $Q_1 \leftarrow BestInsertion(Q_1)$;
17     $r \leftarrow Unif(0, 1)$;
18     if $r \leq \exp\left(\frac{Z(Q_0) - Z(Q_1)}{T}\right)$ then
19         $Q_0 \leftarrow Q_1$;
20         if $Z(Q_1) \geq Z(Q_{best})$ then
21             $flag, iter \leftarrow True, 0$;
22             while $flag$ do
23                 $Q_2 \leftarrow h_{ls}(Q_1)$;
24                 if $Z(Q_2) \geq Z(Q_{best})$ then
25                     $Q_0 \leftarrow Q_2$;
26                 else
27                     $Q_{best}, flag \leftarrow Q_0, False$;
28             $Q_{best} \leftarrow Q_0$;
29     $T \leftarrow \alpha T$, $iter \leftarrow iter + 1$;
30     if $i$ % $it_{upd} = 0$ then
31         update $\rho^d$ and $\rho^{ls}$;
32     if $i$ % $it_{res} = 0$ then
33         $\rho^d \leftarrow \mathbf{1}$, $\rho^{ls} \leftarrow \mathbf{1}$;
34 return $Q_{best}$;
```

The initialisation phase is run once when the algorithm starts. During this phase, the $BestInsertion$ heuristic is applied to generate a new solution Q_0 (line 1 in Alg. 2), which is also the current best solution Q_{best} (line 2 in Alg. 2). The set of low-level heuristics H is partitioned into two subsets, that is the set of removal heuristics H_d (see Appendix B), and the set of local search heuristics H_{ls} (see Appendix C). Two arrays of score vectors for removal, i.e., ρ_d and π_d, and local search heuristics, i.e., ρ_{ls} and π_{ls}, are initialised to 1 (line 3 in Alg. 2). The temperature parameter T is set to T_{start} (line 4 in Alg. 2), whose value is calculated according to the procedure in Pisinger and Ropke (2007). An iteration of the algorithm consists of a *selection* step and a *move acceptance* step.

During the selection step, a removal heuristic h_d and a local search heuristic h_{ls} are sampled from the sets H_d and H_{ls}, respectively. Sampling of a generic heuristic h from a set H is carried out according to a probability distribution $\mathbf{p}_H$ (lines 7–8 in Alg. 2), and a selection probability value p_h is

$$\mathrm{p}_h = \frac{\rho_h}{\sum_{h \in H} \rho_h}. \tag{15}$$

Differently from Roozbeh et al. (2020), a solution Q_0 is modified by a removal heuristic, a local search heuristic, and the *BestInsertion* heuristic to produce a new solution Q_1(line 16–18 of Alg. 2).

After the selection step, the move acceptance step is carried out to define whether the solution generated during the current iteration

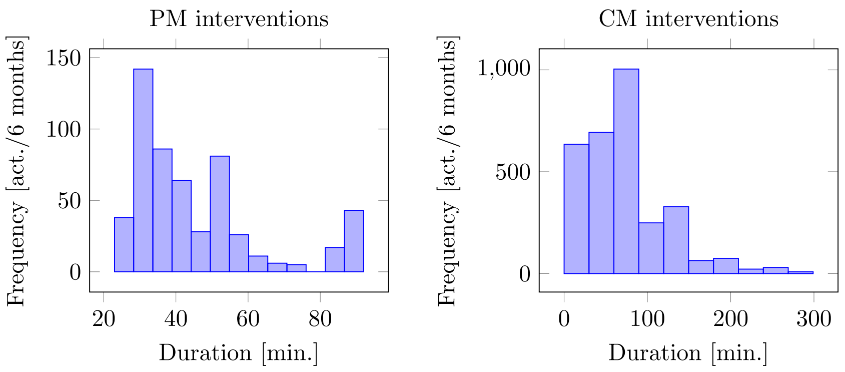


Fig. 1. Distributions of durations for preventive (PM) and corrective (CM) maintenance tasks.

is used in the next iteration or not; the technique applied here is the so-called *simulated annealing* (SA). Depending on the temperature parameter T and the value of the objective function $Z(Q_1)$, a solution Q_1 may be accepted for the next iteration (lines 19–20 in Alg. 2). When T is close to T_{start}, the probability of accepting a solution Q_1 worse than the current solution Q_0 is high, thus allowing the exploration of new regions of the objective function. During later iterations, a new solution Q_1 worse than the current one Q_0 is less likely to be accepted, thus favouring the exploitation of the result found so far. Depending on $Z(Q_1)$, there are three possibilities to update a heuristic's score π_h: first, if the new solution Q_1 is preferred to Q_0 and it is the best found so far, π_h is increased by an amount σ_3. Second, if Q_1 has not been found before but is not the current best solution, π_h increases by σ_2. Third, if the new solution Q_1 is accepted by the SA but is worse than Q_0, a score σ_1 is assigned to the heuristic.

At the end of each iteration, the temperature parameter is progressively reduced to αT, $\alpha \in [0,1]$ (line 33 in Alg. 2), the scores and selection probabilities are updated with a frequency it_{upd}, and reset to their initial value every it_{res} iteration. The rule to update a score ρ'_h is

$$\rho'_h = \begin{cases} (1-\gamma)\cdot\rho_h + \gamma^{\frac{\pi_h}{u_h}} & \text{if } u_h \geq 0, \\ (1-\gamma)\cdot\rho_h & \text{if } u_h = 0, \end{cases} \tag{16}$$

where π_h and u_h represent the score of a heuristic and the number of times the roulette-wheel mechanism has selected that heuristic. The learning rate γ weights the importance of the last score assigned to a heuristic compared to the actual one. The algorithm iterates until T is lower than a value T_{end} (line 6 in Alg. 2); alternatively, a time limit can be set to stop the algorithm.

5. Numerical experiments

The mathematical model and the metaheuristic algorithm proposed in this paper were developed to tackle a real-world problem in the field of maintenance management. This section provides numerical experiments to test the performances of the proposed heuristic using real-world datasets. The computer used to run the tests was equipped with a 64-bit Windows 10 Pro, 32 GB of RAM, 11th. gen. Intel® Core™ i7-11700 @ 2.50 GHz.

5.1. Case study overview

The case study under analysis concerns a workforce routing problem in a European energy-saving company (ESCo). The company reports an annual turnover of more than €50 million and a portfolio of more than 100 customers, which include public infrastructures at all levels, from provinces to ministries, schools, hospitals, and private companies. The company operates nationwide with several active and heterogeneous contracts. The served facilities are hospitals, nursing homes, and local hospital wards, in a region extended over 13 500 km^2. Buildings show heterogeneous accessibility times: some are accessible during the daytime with no limits, whereas others are open only when the concierge staff is working. Exceptions may exist, e.g., while a surgical operation is performed, access to the operating room is prohibited and accessibility is limited to specific time windows. Most maintenance interventions are preventive and periodic; that is, they recur with different frequencies, e.g., weekly, monthly, bi-monthly, etc. On the one hand, preventive tasks include cleaning plants, seasonal inspections before the summer and winter, and statutory security checks. On the other hand, corrective tasks may be triggered by users through a ticketing system and they require the maintenance staff to be on-site within 24 h from the customer's call. The empirical distribution of the duration of the activities considered in our case study over 6 months is shown in Fig. 1. Preventive tasks show an expected duration of at most 90 min and 75% of preventive activities are terminated in less than 50 min. Corrective tasks require on average about 60 min to be carried out and 75% of them are concluded in less than 90 min.

Preventive and corrective tasks are distinguished due to their priority p_i: corrective tasks are assigned the highest value of p_i so that they are inserted in the maintenance schedule with a high probability. The priority value p_i of preventive tasks is a function of their frequency f_i and distance from the due date d_i, and it is calculated as follows:

$$p_i = a\, f_i\, e^{-b\, d_i}, \tag{17}$$

where $a, b > 0$ are set by the system manager. The maintenance staff is employed by the company and receives general-purpose training to address the majority of corrective and preventive maintenance activities. Nevertheless, some tasks require an operator with a specific qualification, i.e., thermal hydraulics or electrical. For example, the thermal-hydraulics skill enables an operator to carry out maintenance of large heating and cooling plants, whereas, e.g., the electrical skill qualifies an operator to install and maintain electrical switchboards. Finally, some activities require the simultaneous presence of multiple operators. An example from the case study is the need to access parts higher than 2.5 m, where it is required the use of a ladder and the assistance, imposed by the regulation, of a second technician, who may intervene in case of emergency.

Current maintenance scheduling process

Maintenance planning in the current practice is carried out manually and on an as-needed basis. A list of activities is assigned daily to the available technicians, who can autonomously decide the execution order of activities. To ease technicians' work, activities are tentatively assigned according to the expertise and the geographic competence area of technicians, that is the area where they usually operate. To coordinate synchronised maintenance activities, staff members interact with each other via mobile phone, and they decide autonomously how to proceed. Exceptions to a tentative schedule may occur at any time,

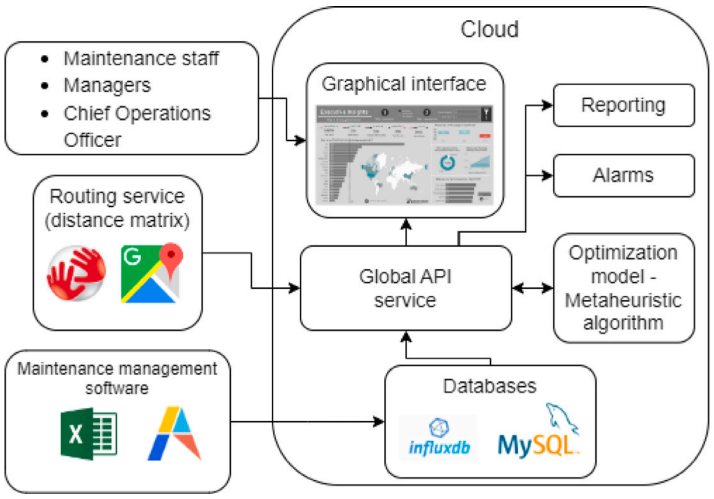


Fig. 2. A schematic representation of the IT infrastructure.

Table 2
The values of the parameters used for the algorithm.

Symbol	Value	Description
T_{start}	$-\frac{0.1\cdot Z}{\ln(0.5)}$	Start temperature, Z is the objective function value of the first solution found
T_{end}	$-\frac{0.005\cdot Z}{\ln(0.5)}$	End temperature, Z is the objective function value of the first solution found
α	0.999	Cooling rate
D_{min}, D_{max}	0.1, 0.2	Degrees of removal
γ	0.1	Learning rate
it_{new}	3	Number of iterations to define recent best solution
$\sigma_1, \sigma_2, \sigma_3$	35, 12, 5	Scores for the heuristics
it_{upd}	20	Iterations over each segment
it_{res}	100	Iterations before the reset
$\theta_1, \theta_2, \theta_3$	3, 13, 7	Parameters of the Shaw heuristic

and, in such a case, the maintenance manager is in charge of coordinating technicians on the field to address the emergency. Drafting a maintenance schedule is time-consuming for the human decision-maker, and respect for all the problem constraints cannot always be ensured.

Algorithm implementation

In addition to the case study under analysis, the ESCo is in charge of servicing several other customers using the designed algorithm. The proposed optimisation solution is embedded in an IT infrastructure, which allows information management and human–system interaction. The users of such a system include, but are not limited to, maintenance staff, operations managers, the chief operating officer, and ticket management companies, which can use third-party maintenance management software. An information technology company realised the IT infrastructure, which was organised into modules. A global API (application programming interface) bridges data collection modules with the algorithm, and with reporting and alarming systems utilising a graphical interface. Routing data are retrieved from external services available under a subscription and are used to feed the optimisation model. Ticketing information is fetched instead from the databases of ticketing companies. Every module of the infrastructure is designed according to the micro-service principle to ease software maintenance and system resilience. Fig. 2 depicts the IT infrastructure and data flow diagram. The optimisation algorithm is called at the end of the workday and returns the maintenance schedule for the day after, which can be submitted to a manager for approval.

5.2. Parameter tuning

The parameters of the proposed metaheuristic were tuned using a trial-and-error approach. An initial guess for each parameter was set to the value found in the literature, if any, or to a reasonable value based on a few standalone problem resolutions otherwise. A batch of 48 test problems was carried out with each parameter value, keeping all the parameters set to their initial guess. The search iterated by increasing/decreasing a parameter until the average objective function value was found to be the highest. Table 2 reports the parameters used in all the experiments carried out later.

5.3. Data sets

Numerical experiments were carried out to test the performance of the proposed algorithm using real-world data. Each data set was obtained by sampling maintenance tasks from a set of 650 activities, which represented the track record of six months for the case-study company. The sampling data set included 4% of activities with a corrective nature – i.e., activities that required taking action within 24 h from the customer call –, whereas the remaining activities were preventive and, on average, 6% of these required the synchronisation of multiple operators. All the data sets used in the numerical experiments in this paper were generated with 4% of corrective maintenance activities and 6% of preventive activities that required synchronisation of operators unless otherwise specified. In the tested instances of the considered case study, there are no corrective tasks that require the synchronisation of operators. The number of technicians assigned to the problem generally ranges from 4 to 8, depending on the size of the contract with the final customer. The skill requirements, the length of the time windows, and the position of the buildings are sampled from real-world data.

5.4. Comparing between ALNS algorithms

This section compares the proposed metaheuristic algorithm with the one proposed by Roozbeh et al. (2018), which has already been

Table 3
The results of the comparison with a termination time of 10 min.

N_a	N_c	N_k	Average objective function value		Standard deviation		Average % of nodes visited		Average number synchronised visits		Difference of nodes visited [%]
			P	R	P	R	P	R	P	R	
100	4	4	446.03	438.73	1.19	1.66	48.10	41.09	1.67	0.63	17.08
		6	465.85	457.00	1.64	2.46	67.16	58.65	2.7	1.87	14.51
		8	479.48	472.74	1.92	3.24	80.27	73.79	2.78	2.61	8.78
		10	489.89	485.26	2.49	3.51	90.28	85.83	3.00	4.13	5.19
200	8	4	846.04	835.65	1.47	2.04	25.98	20.99	1.13	0.21	23.80
		6	870.15	856.77	2.03	2.82	37.57	31.14	3.71	0.75	20.66
		8	892.38	875.67	2.76	3.03	48.26	40.23	4.17	1.46	19.97
		10	914.44	891.33	2.79	3.79	58.85	48.04	5.07	2.4	22.53
300	12	4	1243.68	1232.17	1.27	2.34	17.85	14.14	0.5	0.04	26.06
		6	1269.04	1254.21	1.74	2.52	25.97	21.22	3.13	0.21	22.40
		8	1293.81	1273.96	2.38	2.73	33.91	27.55	3.96	1.08	23.09
		10	1319.56	1292.38	2.51	3.73	42.17	33.46	6.63	2.04	26.04

Legend: the proposed metaheuristic (P); Roozbeh et al. algorithm (R).

Table 4
The results of the comparison with a termination time of 30 min.

N_a	N_c	N_k	Average objective function value		Standard deviation		Average % of nodes visited		Average number synchronised visits		Difference of nodes visited [%]
			P	R	P	R	P	R	P	R	
100	4	4	446.56	439.65	1.09	1.52	48.62	41.97	1.58	0.79	15.83
		6	466.37	458.00	1.58	2.19	67.66	59.61	2.74	1.52	13.50
		8	480.13	474.76	1.94	2.49	80.89	75.73	2.78	2.35	6.82
		10	491.57	487.37	2.7	3.16	91.88	87.86	3.48	4.43	4.60
200	8	4	846.89	837.37	1.19	1.75	26.39	21.81	1.17	0.17	20.98
		6	871.73	858.13	1.51	2.48	38.33	31.79	3.38	1.13	20.57
		8	894.14	877.21	2.16	2.45	49.11	40.96	4	1.54	19.88
		10	915.64	894.62	2.91	3.19	59.44	49.33	4.98	2.81	20.48
300	12	4	1244.50	1233,60	1.09	2.15	18.11	14.62	0.46	0.04	23.90
		6	1270.77	1255.98	1.63	2.4	26.53	21.79	3	0.17	21.76
		8	1294.47	1275.31	2.59	2.57	34.77	27.98	3.83	0.67	21.94
		10	1320.29	1294.13	2.36	2.97	42.40	34.01	6.63	1.92	24.65

Legend: the proposed metaheuristic (P); Roozbeh et al. algorithm (R).

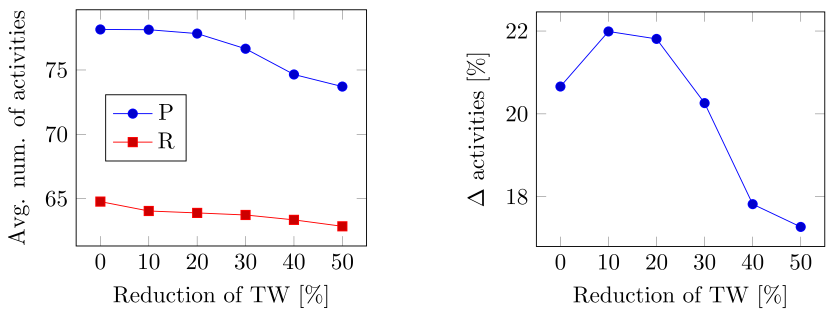
used in the literature as a benchmark for the OP with synchronisations (Garcia, 2022; Yahiaoui et al., 2023; Roozbeh et al., 2020). The tests were carried out using 576 real-world data sets, that is 48 data sets for each combination of $N_a \in \{100, 200, 300\}$ maintenance activities, and several operators $N_k \in \{4, 6, 8, 10\}$. Each experiment resulting from a pair of values (N_p, N_k) was repeated with different termination times, i.e., 10 and 30 min; the termination time values were chosen to reflect the operational needs of the case study ESCO. The proportion of preventive and corrective maintenance activities was kept equal over the data sets: as previously stated, the number N_c of corrective tasks is 4% of N_a, whereas the remainder were preventive. On average, 6% of preventive maintenance activities could require the synchronisation of operators. Corrective and preventive maintenance activities were assigned a score $p_{CM} = 100$ and $p_{PM} = 1$, respectively; such a difference is justified by the practical importance of selecting corrective activities with a high probability. The required skills were evenly distributed between electrical and thermal-hydraulic. Time windows and the location of buildings were set by sampling from real-world data. The objective was the sum of scores given by the selected activities.

The results of the computational experiments are reported in Tables 3 and 4 for the 10 and 30-min resolution time respectively. Due to the high score, corrective maintenance activities were added to the solution by both algorithms. The ALNS heuristic with the *BestInsertion* procedure outperformed on average Roozbeh et al.'s algorithm under all conditions independently of the termination time. The differences in the average objective value between the two algorithms were nevertheless smaller for the 30 min. test than for the 10 min. one. The proposed heuristic could find solutions that were, on average, from 5% to 26% better than those found by the competitor algorithm.

The better performance of the ALNS heuristic is due to the presence of the t_i^{ser} term in Eq. (14): the *BestInsertion* procedure with and without t_i^{ser} was compared to Roozbeh et al.'s algorithm and only with t_i^{ser} a better performance could be observed. Eq. (14) indeed favours the selection of short activities, thus making it possible to add to the maintenance plan more activities on average than the competitor algorithm. Furthermore, a desirable aspect of the proposed heuristic is the standard deviation observed for the objective function value: in all cases, it was lower than the competitor. This result indicates that for the limited set of tests carried out in this study, the proposed heuristic showed a lower variability of results, suggesting that it is more reliable and finds efficient solutions in a wide range of operating conditions.

The number of synchronisations varies widely depending on the instance considered, but in almost all cases, the proposed ALNS algorithm inserts more synchronised visits in the maintenance plan than the competitor as it is reported in the second-last column of Tables 3 and 4. When a rolling-horizon approach was adopted, postponing synchronised maintenance tasks could lead to a scenario where several of these activities are due early. Therefore, scheduling several synchronised activities may be desirable in practice.

Analysing the data present in the last column of Tables 3 and 4, the difference in the average value of the objective function is proportional to the number of visits present in the instance, that is the higher the number of activities among which to chose, the larger the difference between the algorithms. The reason for such a result may be that, when there is a high number of activities among which to choose, the behaviour yielded by Eq. (14) steers the choice of the algorithm towards more compact routes, thus increasing the performance difference.



(a) The average number of maintenance activities selected by the proposed heuristic (P) and the competitor algorithm (R) as a function of time window length reduction.

(b) Percentage difference Δ in the number of activities between Roozbeh et al.'s algorithm and the proposed heuristic.

Fig. 3. Sensitivity analysis of time windows length.

The resolution time does not seem to significantly change the relative performance of the two algorithms: the per cent difference between the two objective functions is smaller in the test with a resolution time equal to 30 min, but this happens because a solution that is already well-optimised is harder to improve than one which is not.

5.5. Sensitivity analysis of time windows length

Five different tests were carried out to evaluate the performance of the proposed algorithm concerning the variation in time window length. The number of activities considered in this test is $N_a = 200$, out of which $N_c = 8$ are corrective maintenance tasks and $N_k = 6$ operators. For the same activities, the time windows within which buildings could be visited were shortened by 10%, 20%, 30%, 40% and 50%, respectively. Since corrective maintenance activities were always selected by both algorithms, the objective function for this analysis was the number of maintenance activities in the plan, which was regarded as more explanatory than the objective of Eq. (1).

The decrease in the average number of activities is almost negligible for the proposed heuristic when the length of time windows is reduced by 10% or 20%, as shown by curve (P) in Fig. 3(a). With a time window reduction equal to or greater than 30%, the performance decrease is evident, meaning that the proposed heuristic struggles to keep the same number of activities in the maintenance plan. Indeed, the number of activities in the maintenance plan decreases by 5.7% on average when the width of the time windows is reduced by 50% compared to the full-length case. Although Roozbeh et al.'s algorithm seems more robust to reductions of the time window length, see curve (R) in Fig. 3(a), this may be due to the quality of the initial solution. Fig. 3(b) depicts the relative difference between Roozbeh et al.'s algorithm and the proposed heuristic: even though it remains greater than 17% the per cent difference drops sharply with time windows reduced by at least 20%.

Although a 50% reduction in time window length is a strong change to the problem's boundary conditions, the proposed heuristic delivered slightly (5.7%) worse performance than with the original time windows and succeeded in planning at least 10 more activities than the competitor under all conditions.

5.6. Sensitivity analysis of synchronisation requirements

Synchronisations impact heavily on the performance of the algorithm: they are hard to insert in an optimal solution, and in several solutions, they yield a long waiting time for at least one operator. To study the performance of the algorithm with several synchronisation requirements, four different tests with $N_s = \{40, 80, 120, 160\}$ synchronisations were carried out on instances with $N_p = 200$ activities, of which $N_c = 8$ are corrective, and $N_k = 6$ operators. Every test was repeated 48 times, each time sampling a different set of activities.

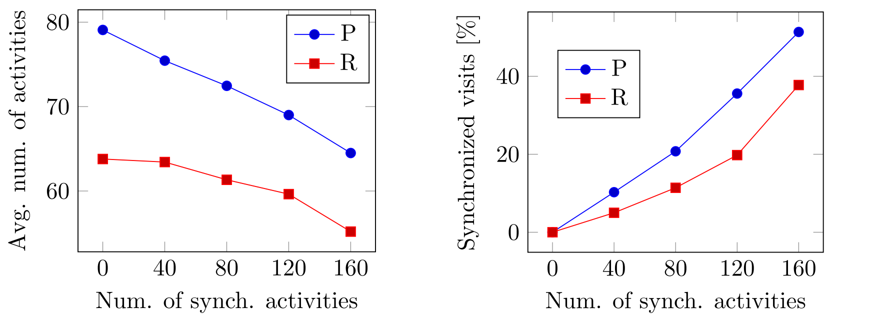
Fig. 4(a) shows the average number of activities planned by the algorithm presented in this paper (P) and Roozbeh et al.'s algorithm (R) as a function of the number of synchronisation requirements. The higher the number of synchronisation requirements, the lower the average number of activities that are added to the solution for both algorithms. Interestingly, the proposed heuristic struggles to accommodate a high number of activities that require synchronisation without reducing the quality of a solution, whereas the competitor algorithm showed a slightly decreasing average number of activities up to 120 synchronisation requirements. The behaviour of the competitor algorithm may be due to the relative quality of the solution, which includes less synchronised activities than that found by the heuristic proposed in this paper. Fig. 4(b) reports the percentage of synchronised activities visited over the total number of selected tasks. Both algorithms seem robust to the increase of synchronisation requirements: as synchronisation requirements increase, the number of synchronised activities inserted in the maintenance plan increases.

A practitioner may appreciate that even when 80% of the activities require synchronisation, 50% of these are included in the solution. This is an extreme condition for the requirements of the case study, yet the mechanics of the proposed heuristic enables to schedule several maintenance activities higher than that delivered by the competitor algorithm.

5.7. Sensitivity analysis of the number of operators

This section studies the sensitivity of the proposed heuristic to the number of available operators $N_k \in \{4, 6, 8, 10\}$. The data sets that were used are the same as the analysis in Section 5.4. The goal of the analysis is to observe the change in the average number of activities per route when the number of available maintenance staff increases.

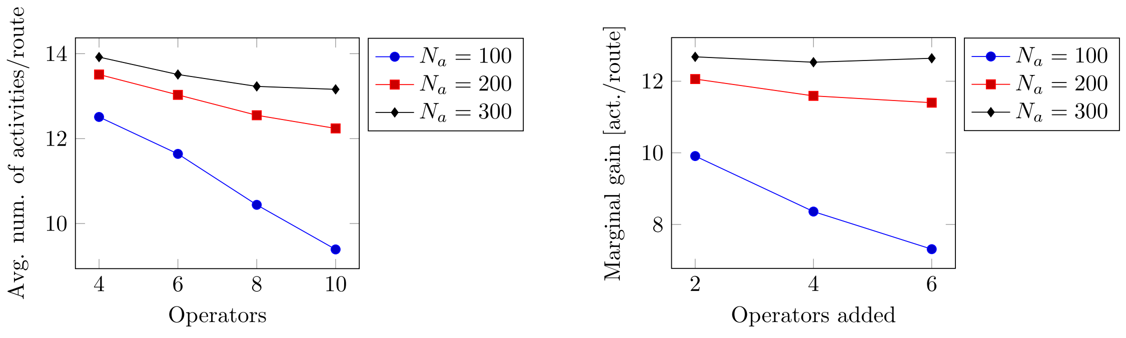
Fig. 5(a) depicts the average number of visits per route as a function of the number of operators for different scenarios, i.e., when the set of activities has size $N_a \in \{100, 200, 300\}$. In general, the relationship between the average number of activities per route and the number of available operators is inversely proportional. However, the trend is clearer in the case of $N_a = 100$ than with $N_a = 200, 300$: the number of activities per route decreases from more than 12 to about nine per route when the number of operators increases from 4 to 10. This behaviour may be due to the geographical distribution of buildings and depots combined with the score of the activities in V_u during the execution of the algorithm. In other words, the routes of the operators overlap with each other, resulting in a higher objective function value but a lower score per operator. With $N_a = 200, 300$, the difference between the average number of activities per route with $N_k = 4$ and $N_k = 10$



(a) The number of activities scheduled by the two algorithms.

(b) Number of synchronised activities in the instance, per cent.

Fig. 4. Sensitivity analysis of synchronisation requirements.



(a) The average number of activities per route as a function of the number of available operators when N_a activities are available.

(b) The marginal gain as a function of the number of operators that are added to the instance with $N_k = 4$ operators when N_a operators are available.

Fig. 5. Results of the sensitivity analysis of the number of operators.

is approximately one activity per route. This result may be due to the relatively high number of activities among which to choose if compared to the number of available operators, i.e., during the creation of a solution, the activities that still have to be added to the maintenance plan are favourably positioned for the operators.

To highlight the benefits of adding further maintenance staff to a problem instance with $N_k = 4$, the marginal gain yielded by new operators – i.e., the number of activities per route that new operators enable to visit – is plotted in Fig. 5(b). With $N_a = 100$, the marginal gain of adding two staff members is about 10 activities per route; however, the marginal gain decreases to 8 activities per route when 4 operators were added, and to approximately 7 when 6 operators were added. Under the assumption $N_a = 100$, it is less and less convenient to add new operators. When the set of activities is larger instead, i.e., with $N_a = 200, 300$, the marginal gain is steady, meaning that having further available operators yields a similar operators' performance. Interestingly, the marginal gain increases when N_a increases; the size of the search space matters to produce better results. The cost–benefit analysis of having more available maintenance staff should be carried out with care to the size of the search space.

When a cost–benefit analysis of employing new maintenance staff was carried out, the analyst should carefully evaluate the problem from the point of view of the number of activities, which seems to be fundamental for achieving better performance of the algorithm.

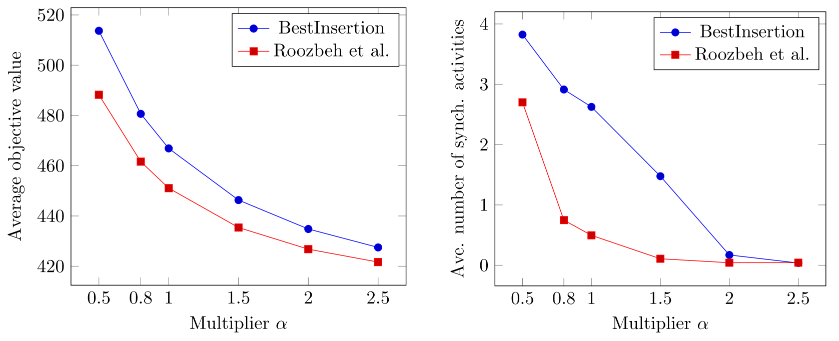
5.8. Sensitivity analysis of the service time duration

In the *BestInsertion* procedure, the service time t_i^{ser} determines the insertion quality ψ. There is an inverse proportionality relationship between t_i^{ser} and ψ in Eq. (14), so visits with a short service time receive a higher score ψ and are prioritised. A sensitivity analysis was carried out to show the effect of t_i^{ser} on the performance of the *BestInsertion* and Roozbeh et al.'s algorithm. Both algorithms were tested on 15 instances with $N_a = 200$ and $N_k = 6$. For each instance, the original value of t_i^{ser} was multiplied by a coefficient $\alpha \in \{0.5, 0.8, 1, 1.5, 2, 2.5\}$, and batches of 15 repetitions were carried out for each α-instance pair.

Fig. 6(a) plots the average objective function value of the algorithms at various values of α. The *BestInsertion* algorithm outperforms the competitor under each scenario. Since all CM activities are always selected, the performance difference lies in the number of PM and, more specifically, in the number of synchronised activities selected on average. Fig. 6(b) shows how the *BestInsertion* assigns a higher number of synchronised activities with most values of α. When t_i^{ser} is twice as large as in the original data, both algorithms insert almost no synchronised activities in a maintenance plan. Such behaviour is probably due to the impossibility of coordinating operators at the same site after they performed high-priority CM activities.

6. Conclusions

The COPTW is a complex combinatorial problem that often requires a metaheuristic to be solved efficiently. In this paper, a specific version of the COPTW is modelled, which includes the possibility to model routes with different departure/arrival depots, time windows to visiting buildings, skill requirements, and multi-operator synchronisation requirements. The presence of synchronisation requirements makes it impossible to separate the contribution of each vehicle to the objective function, thus increasing the computational burden and requiring problem-specific heuristics. A novel procedure for the insertion of synchronised maintenance activities was proposed and tested against a state-of-the-art algorithm for the COPTW. The developed heuristic was embedded in a custom version of the ALNS procedure, and it showed



(a) The average objective function value as a function of parameter α.

(b) The average number of activities requiring synchronisation as a function of the parameter α.

Fig. 6. Results of the service time length sensitivity analysis.

promising results when solving problem instances derived from real-world data. Considering the service time in determining the priority of execution of an activity permitted a significant rise in performance. An in-depth sensitivity analysis of time windows, synchronisation requirements, service time duration, and available maintenance resources was developed to show that the overall procedure proposed in this paper is robust. The analyses of maintenance resources highlighted the importance of providing an adequately large set of maintenance activities among which the algorithm can choose, otherwise, the results may be misleading.

In the real-world case analysed in this study, the proposed heuristic is applied to solve single-day problems using a rolling horizon approach. The preference for short activities may lead to a cyclical delay of long activities, which should be managed by assigning them a high priority. The system manager should be aware of this and should think carefully about setting the parameters in the priority assignment rule. On the other hand, the proposed heuristic showed the ability to plan, on average, a higher number of activities with synchronisation requirements than the competitor algorithm. This may be a desirable feature when the rolling-horizon approach is used: it avoids delaying synchronised activities, which affect the performance of the solution.

The implementation of the proposed COPTW model and resolution algorithm enabled a meaningful improvement in maintenance management for the target company. Due to poor tracking of PM activities with the previous maintenance management system, a quantitative assessment of the benefits yielded by the implemented optimisation solution was not possible. The complete non-existence of data about synchronised maintenance activities was a further hurdle to this valuable comparison between the two management approaches. By modelling the maintenance problem in an ESCO through a COPTW, work schedules that include synchronised activities can be drafted beforehand following the tenet of optimality. Moreover, the scheduling of synchronised activities is carried out with minimum waiting time for maintenance staff. The possibility to use activity duration in a scheduling tool fostered tracking of performance data to refine the accuracy of the schedule. The contract manager was relieved of the assignment task, which often led to changes due to the impossibility of satisfying constraints like time windows, and which was time-consuming. The time required by the algorithm to produce a schedule starting from hundreds of activities could be limited to a few minutes and still obtain good results. Field technicians could benefit from the optimisation solution too: the feeling of disorganisation complained of by maintenance workers was partly resolved by acting according to a rational plan.

Further research should be carried out to test the resilience of the proposed heuristic to different geographies, a large number of operators' qualifications, and constraints to assigning one (or more) technician(s) to a specific task(s). The aspects mentioned above are relevant to the scalability of the optimisation model to new customers, which may show different characteristics compared to the considered case study. When the number of operators became so large that the resolution time was not acceptable in an operational context, it would be efficient to divide the problem into smaller problems; how to perform such a decomposition is a problem on its own that would deserve further study. Finally, unforeseen maintenance interventions must be considered in the practical context. This makes it necessary to change the original (optimised) maintenance plan to address emergencies, which must be served within one hour, or sometimes 30 min, from a customer call. The proposed algorithm is technically suitable for last-minute changes: it could be called to solve a modified problem instance including an emergency activity. However, this use context of the proposed algorithm would be worth further research to consider the practical needs that may occur.

CRediT authorship contribution statement

Andrea Bendazzoli: Methodology, Software, Validation, Investigation, Writing – original draft, Formal analysis. **Michele Urbani:** Conceptualization, Writing – original draft, Visualization, Validation, Investigation, Software, Formal analysis. **Matteo Brunelli:** Conceptualization, Writing – review & editing, Supervision, Project administration, Funding acquisition. **Francesco Pilati:** Conceptualization, Methodology, Writing – review & editing, Supervision, Project administration, Funding acquisition.

Data availability

Data will be made available on request.

Acknowledgements

We acknowledge Lorenzo Pardi for his contribution during the initial phase of this project.

Appendix A. Insertion feasibility

The insertion of a synchronised visit in a solution requires evaluating the feasibility of all subsequent visits linked through synchronisation constraints. Afifi et al. (2016) proposed a constant time algorithm to check the insertion feasibility of synchronised visits, which was used in several works, e.g., in Garcia (2022), Nuraiman et al. (2020), Roozbeh et al. (2018), Parragh and Doerner (2018) and Roozbeh et al. (2020). The procedure to check the insertion feasibility of synchronised visits with more than two operators is reported here to ease the replicability of the proposed algorithm. The arrival time of an operator at a node i is

$$arrival_i = departure_{i-1} + t_{i-1,i}, \tag{A.1}$$

where $departure_{i-1}$ is the end time of the activity at the previous node and $t_{i-1,i}$ is the travel time between nodes $i-1$ and i. The start time of a synchronised activity $start_i$ is the latest arrival time of a group of eligible operators U and the opening time o_i of the building at node i.

$$start_i = \max\left\{\max_{j\in U}\left\{arrival_j\right\}, o_i\right\} \tag{A.2}$$

Due to the different arrival times of the operators involved in a synchronised visit, or for reasons related to time window constraints, some of them may have to wait. The waiting time $wait_i$ spent by an operator waiting for the start of the ith activity is

$$wait_i = start_i - arrival_i. \tag{A.3}$$

Several positions along a route may be considered to evaluate the insertion feasibility of a visit. The time that a visit p along route r, shortly indicated using the function $r(p)$, can be delayed while the solution is still feasible is

$$maxShift_{r(p)} = \min\left\{c_{r(p)} - start_{r(p)}, maxShift_{r(p+1)} + wait_{r(p+1)}\right\}, \tag{A.4}$$

where $c_{r(p)}$ is the close time of the site at activity i. Eq. (A.4) requires recursively evaluating all positions from the last to $r(p)$ every time insertion is considered. When considering a synchronised visit $r(p)$, the $maxShift$ of the activities $j \in V|\{r(p), j\} \subseteq v, v \in VS^C$ is the minimum $maxShift$ among the synchronised tasks, i.e.,

$$maxShift_{r(p)} = \min\left\{maxShift_{r(p)}, \min_{j\in v\setminus r(p)}\left\{maxShift_j\right\}\right\}. \tag{A.5}$$

As stated in Afifi et al. (2016), the insertion of a visit i in a route r between the position p and $p+1$ is feasible if the generated shift

$$shift_i^{(p,p+1)} = t_{r(p),i} + wait_k + t_i^{ser} + t_{i,r(p+1)} - t_{r(p),r(p+1)} \tag{A.6}$$

is less than or equal to the sum of $wait_{r(p+1)}$ and $maxShift_{r(p+1)}$. The least cost insertion is usually applied; in our case, the cost of an insertion is $t_{r(p),i} + t_{i,r(p+1)} - t_{r(p),r(p+1)} + t_i^{ser}$.

Finally, the insertion of synchronised activities must be checked against cross-synchronisations; Afifi et al. (2016) expanded the issue and reported that visits causing cross-synchronisations can be filtered out from the set of possible insertions by using transitive closures (Aho et al., 1972).

Appendix B. Removal heuristics

All algorithms take as input the current solution Q and two quantities $d_{min}, d_{max} \in \mathbb{N}, d_{min} < d_{max}$, which represent the minimum and the maximum number of visits to be removed. These parameters are calculated every time a removal heuristic is applied: they are $d_{min} = \lfloor nD_{min}\rfloor$ and $d_{max} = \lfloor nD_{max}\rfloor$, where n is the number of nodes in the solution, and $D_{min}, D_{max} \in (0,1], D_{min} < D_{max}$. In the following removal heuristics, if a candidate visit is synchronised, also the synchronised visits in different routes are removed.

Random removal The *random removal* heuristic selects a random number n between d_{min} and d_{max} and removes n randomly chosen visits from the solution.

Worst removal The *worst removal* heuristic removes the n visit with the lowest priority, where n is a random number sampled from the uniform probability distribution $U(\{d_{min},\dots,d_{max}\})$.

Sequence removal The *sequence removal* heuristic removes sequences of nodes of length $l = 2,3,4$. A node is randomly selected from the solution, and the following $l-1$ nodes are also removed. To obtain a similar degree of removal for the other removal heuristics, the number of sequences n_{seq} to be removed is $n_{seq} = \lfloor n/l\rfloor$, where n is a random number sampled from the uniform probability distribution $U(\{d_{min},\dots,d_{max}\})$.

Shaw removal The Shaw heuristic (Shaw, 1998) removes from the current solution the n nodes with the highest relatedness. The relatedness Γ_{ij} of two nodes is

$$\Gamma_{ij} = \theta_1 d_{ij} + \theta_2\left|o_i - o_j\right| + \theta_3\Omega_{ij}, \tag{B.1}$$

where the parameters θ_1, θ_2, and θ_3 are called "Shaw parameters", and Ω_{ij} is a parameter that has value -1 if the nodes i and j belong to the same synchronised activity, and 1 otherwise. A random number n is sampled from a uniform probability distribution $U(\{d_{min},\dots,d_{max}\})$ and the $n/2$ nodes showing the highest relatedness are removed from the solution.

Waiting-time oriented removal The waiting-time heuristic removes the node with the highest waiting time among the nodes in the solution.

Appendix C. Local search heuristics

Exchange The exchange heuristic selects all pairs of visits from different routes and, according to qualification and time window constraints, exchanges them to obtain a new solution. For both visits, the position that produces the minimum travel time is chosen.

Or-opt The or-opt heuristic exchanges two visits along the same route. All feasible exchanges along all routes are considered – i.e., those that avoid violations of duration, resources, or time window constraints – and the exchange that minimises the total travel time is applied. The effect of the or-opt heuristic is a reorganisation of visits that decreases the total operation time, thus possibly enabling the insertion of new visits.

Replacement The replacement heuristic tries to insert an unrouted visit by exchanging it with a routed visit. For all unrouted visits $i \in V_u$, all possible substitutions with routed visits are considered and the feasibility of replacement is evaluated. If there are replacements that produce the same objective function value, the one that produces the minimum travel time is chosen.

Two-opt* The two-opt* heuristic exchanges the tails of two arcs in different routes. All feasible exchanges are evaluated and the exchange that produces the lowest total travel time is carried out.

Swap The swap heuristic removes a visit from a route and adds it to another. Only feasible positions are considered, i.e., those that respect time windows, operator qualification, and synchronisation constraints. The visit to be swapped is chosen randomly and the swap that produces the lowest total travel time is carried out.

References

Afifi, S., Dang, D.-C., Moukrim, A., 2016. Heuristic solutions for the vehicle routing problem with time windows and synchronized visits. Optim. Lett. 10 (3), 511–525.
Aho, A.V., Garey, M.R., Ullman, J.D., 1972. The transitive reduction of a directed graph. SIAM J. Comput. 1 (2), 131–137.
Bank, W., 2021. Indicators by the world bank. https://data.worldbank.org/indicator.
Bertoldi, P., Berrutto, V., de Renzio, M., Adnot, J., Vine, E., 2003. How are EU ESCOs behaving and how to create a real ESCO market. In: Proceedings of the 2003 ECEEE Summer Study, European Council for an Energy-Efficient Economy, Paris, France. pp. 909–916.
Bertoldi, P., Rezessy, S., Vine, E., 2006. Energy service companies in European countries: Current status and a strategy to foster their development. Energy Policy 34 (14), 1818–1832.
Borràs, J., Moreno, A., Valls, A., 2014. Intelligent tourism recommender systems: A survey. Expert Syst. Appl. 41 (16), 7370–7389.
Chao, I.-M., Golden, B.L., Wasil, E.A., 1996. The team orienteering problem. European J. Oper. Res. 88 (3), 464–474.

Drexl, M., 2012. Synchronization in vehicle routing—a survey of VRPs with multiple synchronization constraints. Transp. Sci. 46 (3), 297–316.

Gambardella, L., Montemanni, R., Weyland, D., 2012. Coupling ant colony systems with strong local searches. European J. Oper. Res. 220 (3), 831–843.

Garcia, C., 2022. The synchronized multi-assignment orienteering problem. J. Ind. Manag. Optim. 13 (3), 1790–1812.

Gendreau, M., Laporte, G., Semet, F., 1998. A tabu search heuristic for the undirected selective travelling salesman problem. European J. Oper. Res. 106 (2), 539–545.

Golden, B.L., Levy, L., Vohra, R., 1987. The orienteering problem. Nav. Res. Logist. 34 (3), 307–318.

Gunawan, A., Lau, H.C., Vansteenwegen, P., 2016. Orienteering Problem: A survey of recent variants, solution approaches and applications. European J. Oper. Res. 255 (2), 315–332.

Kantor, M.G., Rosenwein, M.B., 1992. The orienteering problem with time windows. J. Oper. Res. Soc. 43 (6), 629–635.

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

Laporte, G., Martello, S., 1990. The selective travelling salesman problem. Discrete Appl. Math. 26 (2), 193–207.

Larsen, P.H., Goldman, C.A., Satchwell, A., 2012. Evolution of the U.S. energy service company industry: Market size and project performance from 1990–2008. Energy Policy 50, 802–820.

Lin, S.-W., Yu, V.F., 2012. A simulated annealing heuristic for the team orienteering problem with time windows. European J. Oper. Res. 217 (1), 94–107.

Meyer, A., Glock, K., Radaschewski, F., 2021. Planning profitable tours for field sales forces: A unified view on sales analytics and mathematical optimization. Omega 105, 102518.

Nuraiman, D., Ozlen, M., Hearne, J., 2020. A spatial decomposition based math-heuristic approach to the asset protection problem. Oper. Res. Perspect. 7, 100141.

Parragh, S.N., Doerner, K.F., 2018. Solving routing problems with pairwise synchronization constraints. CEJOR Cent. Eur. J. Oper. Res. 26 (2), 443–464.

Pisinger, D., Ropke, S., 2007. A general heuristic for vehicle routing problems. Comput. Oper. Res. 34 (8), 2403–2435.

Roozbeh, I., Hearne, J.W., Pahlevani, D., 2020. A solution approach to the orienteering problem with time windows and synchronisation constraints. Heliyon 6 (6), e04202.

Roozbeh, I., Ozlen, M., Hearne, J.W., 2016. A heuristic scheme for the cooperative team orienteering problem with time windows. CoRR abs/1608.05485. URL: http://arxiv.org/abs/1608.05485. arXiv:1608.05485.

Roozbeh, I., Ozlen, M., Hearne, J.W., 2018. An adaptive large neighbourhood search for asset protection during escaped wildfires. Comput. Oper. Res. 97, 125–134.

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

Soares, R., Marques, A., Amorim, P., Parragh, S.N., 2023. Synchronisation in vehicle routing: classification schema, modelling framework and literature review. European J. Oper. Res..

Souffriau, W., Vansteenwegen, P., Vanden Berghe, G., Van Oudheusden, D., 2011. The planning of cycle trips in the province of East Flanders. Omega 39 (2), 209–213.

Stuart, E., Larsen, P.H., Carvallo, J.P., Goldman, C.A., Gilligan, D., 2016. US Energy Service Company (ESCO) Industry: Recent Market Trends. Technical Report, Lawrence Berkeley National Laboratory.

Tang, H., Miller-Hooks, E., 2005. A TABU search heuristic for the team orienteering problem. Comput. Oper. Res. 32 (6), 1379–1407.

Tricoire, F., Romauch, M., Doerner, K.F., Hartl, R.F., 2010. Heuristics for the multi-period orienteering problem with multiple time windows. Comput. Oper. Res. 37 (2), 351–367.

Tsiligirides, T., 1984. Heuristic methods applied to orienteering. J. Oper. Res. Soc. 35 (9), 797–809.

Van Der Merwe, M., Minas, J., Ozlen, M., Hearne, J., 2014. The cooperative orienteering problem with time windows. Optim. Online 7 (11).

Vansteenwegen, P., Souffriau, W., Oudheusden, D.V., 2011. The orienteering problem: A survey. European J. Oper. Res. 209 (1), 1–10.

Vansteenwegen, P., Souffriau, W., Vanden Berghe, G., Van Oudheusden, D., 2009. Iterated local search for the team orienteering problem with time windows. Comput. Oper. Res. 36 (12), 3281–3290.

Vansteenwegen, P., Van Oudheusden, D., 2007. The mobile tourist guide: an OR opportunity. OR Insight 20 (3), 21–27.

Vine, E., 2005. An international survey of the energy service company (ESCO) industry. Energy Policy 33 (5), 691–704.

Wu, Z., Yin, J., Deng, S., Wu, J., Li, Y., Chen, L., 2016. Modern service industry and crossover services: Development and trends in China. IEEE Trans. Serv. Comput. 9 (5), 664–671.

Yahiaoui, A.-E., Moukrim, A., Serairi, M., 2023. GRASP-ILS and set cover hybrid heuristic for the synchronized team orienteering problem with time windows. Int. Trans. Oper. Res. 30 (2), 946–969, arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/itor.13111.

Yee, R.W., Yeung, A.C., Edwin Cheng, T., 2010. An empirical study of employee loyalty, service quality and firm performance in the service industry. Int. J. Prod. Econ. 124 (1), 109–120.