ORIGINAL ARTICLE

Networks

# Optimization of gas metering maintenance services: a multiobjective vehicle routing problem with a set of predefined overlapping time windows

**Lucia Cassettari**[1] | **Mauro Gaggero**[2] | **Stefano Saccaro**[1]

[1]Department of Mechanical Engineering, University of Genoa, I-16145 Genoa, Italy

[2]Institute of Marine Engineering, National Research Council of Italy, I-16149 Genoa, Italy

**Correspondence**
Mauro Gaggero, Institute of Marine Engineering, National Research Council of Italy, I-16149 Genoa, Italy
Email: mauro.gaggero@cnr.it

**Funding information**
No funding information.

Optimization of maintenance services of a company working in the gas metering sector is investigated. In particular, daily tasks of operators are optimized by exploiting the paradigm of vehicle routing problems with a set of predefined time windows that overlap one with the others and four competing objectives to take into account. First, an exact integer formulation is presented that can be solved only for a reduced number of customer sites to visit. Then, a heuristic approach is proposed to find approximate solutions with huge savings on the computational effort, also for high-dimensional instances. Numerical results on both real and synthetic scenarios showcase the effectiveness of the approach.

**KEYWORDS**
Gas metering service optimization; vehicle routing problem with time windows; multiple objectives; mixed-integer programming; heuristics.

## 1 | INTRODUCTION

Optimization of natural gas distribution networks is an interesting topic for several disciplines, ranging from Operations Research to Engineering, that embraces financial, technical, and environmental aspects [23]. This field of research includes various optimization problems, involving optimization of many different parts of the distribution network, such as physical components, storage, transportation, and layout. Examples include pipeline and compressor station management [35], planning to face uncertainties [6], computation of the best amount of natural gas to transfer [1], specification of the optimal network layout to minimize investment costs [51], as well as design of new networks or expansion of existing ones [34]. However, as pointed out also in previous works [39], the literature lacks

of approaches to plan and optimize last-mile service operations at the end-user level, such as planning of maintenance and technical activities on gas meters to guarantee satisfaction of customers and profitability of natural gas providers. This paper aims at bridging this gap by optimizing the logistics operations related to the activation and deactivation of gas meters of customers. In particular, the daily tasks of a set of operators of a company working in the gas metering sector are optimized by exploiting the paradigm of vehicle routing problems with time windows. Thus, the problem studied in this work is an interesting variant of the classical vehicle routing one coming from a real application. Each operator is assumed to drive a vehicle of the company fleet and has to visit a certain number of sites in order to perform given tasks on gas meters of customers. Such tasks have to be executed within a sequence of predefined time windows that overlap one with the others, and capacity constraints on working time of operators for each window and day exist. Time windows account for constraints that have to be taken into account when planning maintenance tasks on gas meters of customers and cannot be violated. On the one hand, the presence of time windows where tasks are performed allows final customers to reserve a particular, restricted portion of the day to make operators access gas meters located in their own properties. Hence, final customers are satisfied only if the company serves them by respecting the corresponding time windows. On the other hand, time windows allow to organize the working day of operators, and penalties with the natural gas provider of the client are avoided if maintenance services are performed within the prescribed time windows. In more detail, the gas provider usually agrees with final customers the preferred days and time slots for interventions on gas meters. Then, it collects all the requests of customers in a given area by grouping them by date and communicates the list of interventions with the required time windows to the gas metering company of the considered area. The gas metering company cannot modify the list generated by the natural gas provider (it is not possible to insert new operations or change the time windows). In this work, we adopt the point of view of the gas metering company: given the list, the goal is to search for optimal paths of operators. We have no chance of modifying the list of interventions and the related time windows; we have just to organize in an optimal way the working day of operators by satisfying all the constraints given by the time windows. The considered problem has a multiobjective nature since four conflicting goals are usually pursued by gas metering companies: (i) minimize the overall distance traveled by operators, (ii) perform a "balanced" assignment of tasks to operators, i.e., make the working time of operators within a time window as equal as possible, (iii) reduce idle times of operators, and (iv) execute all tasks with a number of operators as small as possible. To the best of the authors' knowledge, no previous attempts pursuing the aforementioned goals (i)-(iv) in the optimization of gas metering services exist in the literature. The direct application of results available in the literature on vehicle routing problems with time windows is prevented by goals and constraints that have to be taken into account in this problem.

First, we present an exact formulation for the considered problem based on integer programming. Unfortunately, finding an optimal solution is possible only for small instances of the problem. Then, we propose a heuristic approach that allows finding approximate solutions also for a large number of customers to visit, with a very low computational effort. The developed heuristic is composed of four steps, and combines the savings algorithm of Clarke and Wright [13], the 2-opt local search [15], and a nearest-neighbor criterion. It belongs to the class of cluster-first-route-second approaches [43], as the various customer sites are first organized into clusters, and then routes are constructed for each cluster. In more detail, the first step is devoted to the computation of upper bounds on the maximum working times of operators for each time window. Such bounds account for overlaps of the various windows and balance working times of operators within each window. In the second step, the overall problem is split into different subproblems, one for each time window, and clusters of customer sites are created by exploiting the output of the savings algorithm. The result is a series of routes for each time window, starting from the depot and returning therein after visiting a certain amount of sites. In the third step, such routes are optimized through a local search algorithm and then merged to create daily paths across the various windows that start from and end to the depot. Lastly, the fourth

step refines the paths computed through the previous steps via a random search. Another advantage of the proposed heuristic is its simplicity: there exists only one parameter to be tuned (the number of random perturbations in the final step), and therefore it can be easily applied by practitioners with no difficulties.

To sum up, the main contributions of this work are the focus on service operations at the end-user level of a gas distribution network and the formalization of an optimization problem for their optimal management as a vehicle routing problem with several competing objectives based on a set of predefined, overlapping time windows, for which an exact and a heuristic approach are proposed. To the best of the authors' knowledge, both aspects have not been sufficiently investigated up to now.

The rest of this paper is organized as follows. Section 2 reviews recent contributions in the field of vehicle routing problems with time windows. The considered gas metering service optimization problem is presented in Section 3, together with its integer programming formulation. The heuristic approach developed to find approximate solutions is discussed in Section 4. Numerical results are reported in Section 5. Conclusions are drawn in Section 6.

## 2 | LITERATURE REVIEW

Vehicle routing is a classic NP-hard, combinatorial optimization problem that has received a lot of attention from the research community working on Operations Research [43, 8]. The problem consists in searching for an optimal route (for instance, the one characterized by the minimum distance or minimum time) for a fleet of vehicles that have to visit a set of customers, under suitable constraints, starting from a depot and returning therein at the end of the route. As previously pointed out, in this paper we consider one of the most studied variants: the vehicle routing problem with time windows. In this case, customers have to be visited within given time window constraints [17, 4]. As said, we focus on the optimization of gas metering maintenance tasks. Such problem, which did not receive significant attention in the past, involves a sequence of predefined, hard time windows that overlap one with the others. We consider the situation in which all customers must be visited within a fixed, limited number of time windows; all the windows have the same duration, and several tasks are assigned to the same window.

In a nutshell, two main families of approaches are available in the literature to solve vehicle routing problems with time windows: exact and heuristic techniques. Among the exact ones, we mention classical methods based on Lagrange relaxation and dynamic programming [29, 19, 28], together with recent developments relying on set-partitioning, column generation, and branch-and-price methods [9, 2, 3, 18, 32, 14, 20]. Concerning heuristic techniques, several approaches are available that are based on genetic or evolutionary algorithms [44, 12, 27], particle swarm optimization [11, 48, 33], simulated annealing [31, 45, 30], and methods based on local search [26, 10, 24, 50]. Other works investigate the selection of the most suitable heuristic approach among several available ones using learning techniques [22]. An important variant of the vehicle routing problem with time windows is the presence of multiple time windows [16, 7], while recent developments are focused on the development of heuristic approaches that are able to account for flexibility in the execution of tasks within time windows [42, 47] and uncertainties in the travel and execution times [37, 21, 32].

The two main characteristics of the problem considered in this work that are not shared by the aforementioned works are the presence of a sequence of predefined overlapping time windows and a number of competing objectives to be taken into account. Concerning the sequence of predefined overlapping time windows, an exception is [46] for delivery and pickup problems, where the authors point out that the overlap of time batches is a source of issues since it entails the problem of assigning and reusing vehicles among different batches, in addition to vehicle routing. Two approaches based on tabu search are developed to solve the problem in an approximate way. Overlapping windows

are mentioned also in [39], but they are taken into account by using a simple scoring policy based on experience of practitioners, with no guarantee on the violation of the corresponding working times of operators or vehicles. Lastly, [25] presents mixed-integer linear programs to take into account possibly overlapping time windows, and points out that, in the absence of overlaps, the number of variables and constraints can be conveniently reduced, thus simplifying the problem to be solved. As regards the presence of multiple competing objectives, some results are available in the literature [41, 12, 38, 5, 49]. In [41], an approach based on evolutionary computation is proposed to minimize traveled distance, driver remuneration, and number of vehicles subject to constraints such as time windows and vehicle capacity. In [12], the goals are the minimization of the number of vehicles and the total traveled distance in the presence of time windows using again an evolutionary algorithm. Reference [38] aims at minimizing operational costs and maximizing service level using memetic and local search approaches. In [5], an artificial bee colony algorithm is developed to minimize total transportation costs and maximize customers' satisfaction. Lastly, [49] pursues reduction of the number of vehicles and minimization of time-wasting during the delivery process caused by early arrival using again an evolutionary approach. However, in none of the aforementioned works the goals of balancing the assignment of tasks and the reduction of idle times are explicitly taken into account. A balance of the traveled distance over vehicles is pursued in [40], together with the minimization of the number of vehicles, by means of a hybrid estimation of distribution algorithm. However, also in this case the goal of balancing the working time of operators is not addressed.

## 3 | PROBLEM FORMULATION

The motivation of the problem considered in this paper is the need for planning the daily duties of a set of operators of a company working in the gas metering sector. The company needs to plan a certain amount of tasks, given by activation and deactivation of gas meters of end users, i.e., the opening and closing of gas meters corresponding to beginning and cancellation of supply contracts with the client company, respectively. All tasks have to be performed within a set of predefined, hard time windows to increase satisfaction of final customers and avoid penalties with the gas provider. In other words, we have to assign a sequence of tasks to a set of operators. Operators drive vehicles of the company fleet and have to execute tasks at given customer sites (i.e., there is a one-to-one correspondence between sites and tasks) within certain time windows. Moreover, all operators start from the company depot at the beginning of the day and return therein at the end of the day.

The problem can be modeled through a graph, whose nodes and arcs represent customer sites and connections among them, respectively. With a little abuse of notation, from now on we will use the terms "node", "task", "operation", and "customer site" interchangeably. As said, four goals have to be pursued, which may conflict one with the others: (i) minimize the total distance traveled by operators, (ii) assign tasks to operators in a balanced way by making the working time of operators, given by the sum of the time needed to perform tasks and the time to travel among customers' locations, as equal as possible within a time window, (iii) avoid idle times of operators, i.e., the case in which an operator has no tasks to execute in a time window between two windows with at least an operation to be performed, and (iv) use the minimum number of operators to perform all tasks.

Goals (i) and (iv) are due to obvious economical savings for the gas metering company: the lowest is the distance traveled by operators, the lower are the travel costs and the higher is the number of tasks that can be executed by a given operator, which in turn implies further savings. The same applies for the number of operators that are needed to perform the various activities. The balance of the working time of operators per time window, i.e., goal (ii), is introduced to account for a common request of gas metering companies. In fact, due to labor agreements, such companies have to divide the overall workload as fairly as possible among the available operators. The balance of

TABLE 1 Notation for the considered problem.

| | |
|---|---|
| $n$ | number of tasks |
| $m$ | number of operators |
| $q$ | number of time windows |
| $i, j$ | indices for tasks |
| $k$ | index for time windows |
| $l$ | index for operators |
| $\tau_i$ | execution time of tasks |
| $W_k$ | duration of time windows |
| $V_k$ | overlapping of time windows |
| $R$ | maximum working time per day of operators |
| $F_{kj}$ | relationship between time windows and tasks |
| $S_{ij}$ | precedence between tasks |
| $D_{ij}$ | distance between nodes |
| $T_{ij}$ | travel time between nodes |
| $x_{lij}$ | arc variable |
| $u_{li}$ | subtour elimination variable |
| $p_{lk}$ | auxiliary decision variable |

working time per window has the consequence of balancing the working time of operators over the entire day, and it concurs to avoid idle times, i.e., goal (iii). In fact, the balance on the whole day does not guarantee a balance within each time window since the average daily working time may be the same for different combinations of working times per window. As said, the reduction of idle times pursued through goal (iii) is strictly related to the balance of working times of operators. In more detail, an idle time preceded and followed by very busy periods, as well as an unbalanced assignment of tasks, results in an inefficient management of the workforce. In fact, it is much more convenient for the gas metering company to let windows with no tasks to be executed at the beginning or at the end of the day in order to use operators to perform other types of activities than activation and deactivation of meters. To avoid burdening the notation, we decided to neglect investigation of such additional activities but we just limit to avoid idle times within two windows with at least one task to be executed.

In the following, we formulate the problem as an integer programming one. Toward this end, below we provide a description of the involved variables and parameters. A summary of all the considered quantities is reported in Table 1.

### Number and type of operations

The number of customer sites to visit in each day, or, equivalently, the number of tasks, is denoted by $n$. As said, each operator has to perform a subset of such tasks starting from the company depot and, after completing the tour, he/she has to return to the depot. To avoid burdening the notation, we assume that tasks 1 and 2 are fictitious and correspond to the start from and return to the depot, respectively. Each task is associated with an execution time, which we denote by $\tau_i$, $i = 1, \ldots, n$. Clearly, we have $\tau_1 = 0$ and $\tau_2 = 0$ since such tasks are fictitious.

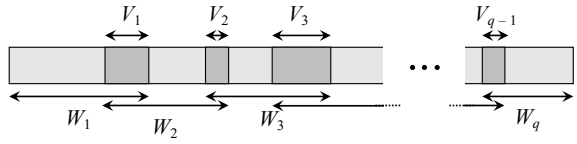



FIGURE 1 Sketch of the sequence of $q$ predefined, overlapping time windows considered in this paper.

### Time windows

The execution of operations is scheduled in a sequence of $q$ predefined, hard time windows (see Figure 1). No flexibility is allowed, in the sense that no deviation from the required window is permitted, and several tasks have to be executed within the same window. The duration of each window is denoted by $W_k$, $k = 1, \ldots, q$. Time windows overlap with each other: we denote by $V_k$, $k = 1, \ldots, q-1$, the amount of time such that windows $k$ and $k+1$ are overlapped.

To avoid burdening the notation, we introduce the fictitious windows 0 and $q+1$ to impose the start from and return to the company depot at the beginning and at the end of each day. Then, we define a $(q+2) \times n$ binary matrix $F$, whose elements $F_{kj}$ are such that

$$F_{kj} := \begin{cases} 1 & \text{if the operation } j \text{ belongs to the time window } k, \\ 0 & \text{otherwise,} \end{cases}$$

where $k = 0, \ldots, q+1$ and $j = 1, \ldots, n$, with the window $k$ preceding the window $k+1$. Since each task has to be performed in one and only one time window, the matrix $F$ is such that $\sum_{k=0}^{q+1} F_{kj} = 1$ for all $j = 1, \ldots, n$. As regards the first two fictitious tasks, we assume they are assigned to the first and last time windows, respectively. Moreover, they are the unique operations scheduled in their time windows with, of course, the first one preceding the second one. Thus, we have $F_{01} = 1$, $F_{0j} = 0$, $j = 2, \ldots, n$, as well as $F_{q+1\,2} = 1$, $F_{q+1\,j} = 0$, $j = 1$ and $j = 3, \ldots, n$.

One of the objectives of the assignment is to avoid idle time of operators (see goal (iii) stated at the beginning of this section) since this corresponds to their inefficient management, with the consequence of an increase of costs for the company. In other words, we have to avoid that an operator has no tasks to execute in a time window between two windows with at least one task to be performed, excluding the fictitious windows 0 and $q+1$. As it will be detailed in the following, a suitable penalization term in the cost function to minimize will be adopted to reduce the occurrence of this situation.

### Number of operators

The number of available operators for the execution of tasks is denoted by $m$. As said, one of the goals of the assignment is to rely on the minimum number of operators that is needed to satisfy all tasks requests (see objective (iv) at the beginning of this section).

The maximum working time that an operator can provide for each time window coincides with the duration of the window: it is equal to $W_k$, $k = 1, \ldots, q$. However, the overlap of time windows may reduce such amount of time in practice. In more detail, since the windows $k$ and $k+1$ overlap for an amount of time equal to $V_k$, the maximum working time of operators for the two windows is equal to $W_k + W_{k+1} - V_k$. Furthermore, operators are characterized by a maximum daily working time denoted by $R$. Clearly, we have $R \leqslant \sum_{k=1}^{q} W_k$ due to the overlap of time windows. Such times include both the times needed to perform the various tasks and the times needed to travel among customer sites.

### Precedence relations, distance, and travel times

We account for the precedence among tasks by means of a $n \times n$ binary matrix $S$, whose elements $S_{ij}$ are such that

$$S_{ij} := \begin{cases} 1 & \text{if operation } i \text{ is in a time window succeeding operation } j, \\ 0 & \text{otherwise,} \end{cases}$$

where $i = 1, \ldots, n$ and $j = 1, \ldots, n$. Clearly, we have $S_{ii} = 0$ for $i = 1, \ldots, n$, $S_{1j} = 0$ for $j = 1, \ldots, n$, and $S_{2j} = 1$ for $j = 1, \ldots, n, j \neq 2$.

The distance between nodes is taken into account via the $n \times n$ symmetric matrix $D$, whose elements $D_{ij}$ represent the distance between node $i$ and node $j$, $i = 1, \ldots, n$, $j = 1, \ldots, n$. Of course, we have $D_{ii} = 0$ for $i = 1, \ldots, n$. We point out the role played by the first two tasks, for which $D_{12} = D_{21} = 0$. In fact, as said such operations are conventionally located at the company depot to ensure that paths of operators start and return therein.

Travel times between customer sites are modeled through the $n \times n$ symmetric matrix $T$, whose elements $T_{ij}$ account for the time needed to travel from node $i$ to node $j$, $i = 1, \ldots, n$, $j = 1, \ldots, n$. Likewise for the matrix $D$, also in this case we have $T_{ii} = 0$ for $i = 1, \ldots, n$, and $T_{12} = T_{21} = 0$.

### Decision variables

The binary decision variables $x_{lij} \in \{0, 1\}$, $l = 1, \ldots, m$, $i = 1, \ldots, n$, $j = 1, \ldots, n$, take into account if operator $l$ is enabled to travel along the arc from node $i$ to node $j$, i.e., he/she is committed to perform tasks $i$ and $j$ consecutively. In more detail, such variables are defined as follows:

$$x_{lij} := \begin{cases} 1 & \text{if operator } l \text{ travels from node } i \text{ to node } j, \\ 0 & \text{otherwise,} \end{cases}$$

where $l = 1, \ldots, m$, $i = 1, \ldots, n$, and $j = 1, \ldots, n$.

We introduce also the integer variables $u_{li} \in \mathbb{N}$, $l = 1, \ldots, m$, $i = 1, \ldots, n$, to avoid subtours in the path [36]. Such variables denote the position of task $i$ performed by operator $l$ in the sequence of tasks executed by the operator.

Then, we consider the auxiliary binary variables $p_{lk} \in \{0, 1\}$, $l = 1, \ldots, m$, $k = 1, \ldots, q$, which are equal to 1 if operator $l$ has to perform at least one task in time window $k$, otherwise they are equal to 0. Such quantities are crucial to avoid idle times of operators, and can be written in terms of the variables $x_{lij}$ (we will connect them through suitable constraints, as detailed in the following). They are introduced to avoid burdening the notation.

## 3.1 | Formulation of the optimization problem

Based on the previous definitions, the considered problem can be formulated as an integer programming problem as follows:

$$\min \; c_1 \sum_{l=1}^{m}\sum_{i=1}^{n}\sum_{j=1}^{n} D_{ij}\, x_{lij} + c_2 \sum_{l=1}^{m}\sum_{k=1}^{q}\left|\sum_{i=1}^{n}\sum_{j=1}^{n}(T_{ij}+\tau_j)F_{kj}\, x_{lij} - \frac{1}{m}\sum_{z=1}^{m}\sum_{i=1}^{n}\sum_{j=1}^{n}(T_{ij}+\tau_j)F_{kj}\, x_{zij}\right| +$$
$$c_3 \sum_{l=1}^{m}\sum_{k=1}^{q-1} 0.5\left(1+\left|-M(p_{lk+1}-p_{lk}+1)+\sum_{z=k+1}^{q} p_{lz}\right| - \left|-M(p_{lk+1}-p_{lk}+1)+\sum_{z=k+1}^{q} p_{lz}-1\right|\right) \tag{1a}$$

$$\text{s.t.} \quad \sum_{j=1}^{n} x_{l1j} = 1, \quad l = 1,\dots,m, \tag{1b}$$

$$\sum_{i=1}^{n} x_{li2} = 1, \quad l = 1,\dots,m, \tag{1c}$$

$$\sum_{l=1}^{m}\sum_{i=1}^{n} x_{lij} = 1, \quad j = 3,\dots,n, \tag{1d}$$

$$u_{li} - u_{lj} + n\, x_{lij} \leqslant n-1, \quad l = 1,\dots,m,\ i = 1,\dots,n,\ j = 3,\dots,n, \tag{1e}$$

$$\sum_{l=1}^{m}\sum_{i=1}^{n}\sum_{j=1}^{n} F_{ki} x_{lij} = \sum_{i=1}^{n} F_{ki}, \quad k = 1,\dots,q, \tag{1f}$$

$$\sum_{j=1}^{n} S_{ij} x_{lij} = 0, \quad l = 1,\dots,m, \quad i = 1,\dots,n, \tag{1g}$$

$$\sum_{i=1}^{n} x_{lij} \leqslant 1, \quad l = 1,\dots,m, \quad j = 1,\dots,n, \tag{1h}$$

$$\sum_{j=1}^{n} x_{lij} \leqslant 1, \quad l = 1,\dots,m, \quad i = 1,\dots,n, \tag{1i}$$

$$x_{lii} = 0, \quad l = 1,\dots,m, \quad i = 1,\dots,n, \tag{1j}$$

$$\sum_{i=1}^{n} x_{lij} = \sum_{h=1}^{n} x_{ljh}, \quad l = 1,\dots,m, \quad j = 3,\dots,n, \tag{1k}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}[T_{ij}+\tau_j]F_{kj} x_{lij} \leqslant W_k, \quad l = 1,\dots,m, \quad k = 1,\dots,q, \tag{1l}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} T_{ij} x_{lij} + \sum_{k=1}^{q}\sum_{i=1}^{n}\sum_{j=1}^{n}\tau_j F_{kj} x_{lij} \leqslant R, \qquad l = 1,\dots,m, \tag{1m}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{z=k}^{k+1}[T_{ij}+\tau_j]F_{zj} x_{lij} \leqslant W_k + W_{k+1} - V_k, \quad l = 1,\dots,m, \quad k = 1,\dots,q-1, \tag{1n}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n}\sum_{z=1}^{k}[T_{ij}+\tau_j]F_{zj} x_{lij} \leqslant \sum_{z=1}^{k} W_z - \sum_{z=1}^{k-1} V_z, \quad l = 1,\dots,m, \quad k = 1,\dots,q, \tag{1o}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} F_{kj} x_{lij} \geqslant p_{lk}, \quad l = 1,\dots,m, \quad k = 1,\dots,q, \tag{1p}$$

$$\sum_{i=1}^{n}\sum_{j=1}^{n} F_{kj} x_{lij} \leqslant M p_{lk}, \quad l = 1,\dots,m, \quad k = 1,\dots,q, \tag{1q}$$

where $x_{lij} \in \{0,1\}$, $u_{li} \in \mathbb{N}$, $p_{lk} \in \{0,1\}$ for $l = 1,\ldots,m$, $i = 1,\ldots,n$, $j = 1,\ldots,n$, $k = 1,\ldots,q$, $M$ is a large positive constant, and $c_1$, $c_2$, and $c_3$ are given positive weight coefficients. For the sake of brevity, we will refer to (1) as vehicle routing problem with overlapping time windows, or VRPOTW for short.

The objective function in (1a) is made up of three terms weighted by the coefficients $c_1$, $c_2$, and $c_3$. The first term is the total distance traveled by operators and accounts for goal (i) listed at the beginning of this section. The second term represents the imbalance in the assignment of tasks to operators over time windows and is given by the difference between the amount of working time of an operator in a time window and the average working time of operators within the same window. The working time includes both the time needed to perform a task and the travel time between customers' locations. Clearly, a balanced allocation of tasks to operators according to goal (ii) requires that such a difference is small. The goal (iii) about the reduction of idle times of operators is taken into account via the third term of the cost function. In more detail, for a given operator, it is a penalization term that is equal to the number of idle times over time windows. Its expression can be easily derived by subtracting two ramp functions with argument $-M(p_{lk+1}-p_{lk}+1)+\sum_{z=k+1}^{q} p_{lz}$. The rationale is to penalize, for a given $l$, sequences of $p_{lk}$, $k = 1,\ldots,q$, with a zero between two ones.

As regards constraints, (1b) and (1c) ensure that each path starts and ends at the company depot. Constraints (1d) impose that each task except the first and second ones is assigned to one and only one operator. Equations (1e) avoid subtours. Constraints (1f) guarantee that all tasks to be performed are assigned within the corresponding time window. The precedence among tasks is taken into account by means of (1g). Constraints (1h) and (1i) ensure that each node is visited only once. A node cannot be simultaneously origin and destination by imposing (1j). Constraints (1k) ensure that, if the operator $l$ visits node $j$, $j = 3,\ldots,n$, a predecessor and a successor of node $j$ exist in the path of the same operator. Equations (1l) and (1m) account for the maximum working time of operators in each window and in the entire day, respectively. Such time is made up by the sum of the travel time and the time needed to perform the various tasks. The overlap of time windows $k$ and $k+1$, $k = 1,\ldots,q-1$, which reduces the amount of working time available for operators, is taken into account via (1n) and (1o). Lastly, constraints (1p) and (1q) connects the variables $x_{lij}$ and $p_{lk}$, in such a way that $p_{lk} = 1$ if the operator $l$ has to perform at least one task in the time window $k$, and $p_{lk} = 0$ otherwise.

Since one of the goals of the considered problem is the minimization of the number of operators, see goal (iv) discussed at the beginning of this section, we may solve a sequence of VRPOTW, each one characterized by a fixed value of $m$ that progressively increases of one unit until a feasible solution is found (the first problem of the sequence corresponds to the choice $m = 1$). The rationale is that the use of a number of operators lower than the minimum one is a cause of infeasibility for the considered problem. A failure in finding a feasible solution for the problem corresponding to a given value of $m$ may be overcome by increasing $m$. Thus, it is important to quickly check the lack of feasibility for the problem under consideration. The minimum number of operators needed to execute all the tasks can be determined by starting with a small value of $m$ and then increasing it until a feasible solution can be found. Clearly, such a choice is sub-optimal since it ignores the possible conflicting nature of goals (i)–(iii) and (iv). A better trade-off could be obtained by considering the number of operators $m$ as an additional decision variable in (1), but this further complicates finding a solution since it entails a structural dependence of the number of constraints on the amount of operators.

In fact, finding an exact solution to VRPOTW is still very difficult, especially in the case of a large number of tasks and operators, and therefore a heuristic approach is needed to find approximate solutions with a reduced computational burden, as detailed in Section 4.

**Remark** The absolute values in the second and third terms of the cost function in (1a) can be easily linearized. In fact, as it is well known, an absolute value in the cost function of a mathematical programming problem can be replaced by

introducing an additional decision variable that is constrained to be greater than the argument of the absolute value and also greater than its opposite. For instance, if we refer to the second term of the cost, we may introduce an additional real decision variable $y_{lk}$, $l = 1, \ldots, m$, $k = 1, \ldots, q$, and consider the minimization of the cost $c_2 \sum_{l=1}^{m} \sum_{k=1}^{q} y_{lk}$ with constraints given by (1b)–(1q) together with

$$y_{lk} \geqslant \sum_{i=1}^{n} \sum_{j=1}^{n} (T_{ij} + \tau_j) F_{kj}\, x_{lij} - \frac{1}{m} \sum_{z=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{n} (T_{ij} + \tau_j) F_{kj}\, x_{zij}$$

and

$$y_{lk} \geqslant -\sum_{i=1}^{n} \sum_{j=1}^{n} (T_{ij} + \tau_j) F_{kj}\, x_{lij} + \frac{1}{m} \sum_{z=1}^{m} \sum_{i=1}^{n} \sum_{j=1}^{n} (T_{ij} + \tau_j) F_{kj}\, x_{zij}$$

for $l = 1, \ldots, m$ and $k = 1, \ldots, q$. In this way, VRPOTW reduces to a mixed-integer linear programming problem.

## 4 | THE PROPOSED FOUR-STEP HEURISTIC APPROACH

In this section, we describe a simple yet effective four-step heuristic approach (FSH, for short) to find suboptimal solutions to VRPOTW. The proposed approach combines the savings algorithm of Clarke and Wright, the 2-opt local search method, and a nearest-neighbor criterion. As pointed out at the beginning of Section 3, the goals are the minimization of the total traveled distance of operators, a balanced assignment of tasks to operators within the same time window (in order to have similar working times of operators within a certain window), the reduction of idle times of operators, and the adoption of a number of operators as small as possible, i.e., the maximization of the number of tasks per operator while satisfying a given maximum daily workload per operator and time window.

The proposed approach is made up of four steps, denoted by "initialization", "clustering", "cluster optimization and merging", and "refinement", respectively. The various steps are summarized in Figure 2 and detailed in the following, while the pseudo-code of the proposed method is reported in Algorithm 1. Their application to a toy example with $n = 20$ tasks and $q = 3$ time windows is sketched in Figure 3.

### 4.1 | Initialization step

The developed heuristic starts with an initialization step that aims at taking into account the overlap of time windows and at giving temporal continuity to the work of operators throughout the entire day without idle times, thus reducing the risk that an operator has no tasks to be performed in a time window between two windows with one or more operations. In other words, this step accounts for the goal (iii) of the considered problem. Moreover, the initialization step also accounts for the goal (ii) related to a balanced assignment of tasks to operators within the same time window, as detailed in the following.

In more detail, upper bounds $\tilde{U}_k$, $k = 1, \ldots, q$, on the maximum working time of operators for each time window are computed on the basis of the corresponding workload, given by the sum of the execution times of all tasks belonging to the window. Such bounds are determined as follows:

$$\tilde{U}_k := \min\left(W_k,\ R\,\frac{\sum_{i=1}^{n} F_{ki}\,\tau_i}{\sum_{i=1}^{n} \tau_i}\right), \quad k = 1, \ldots, q, \tag{2}$$

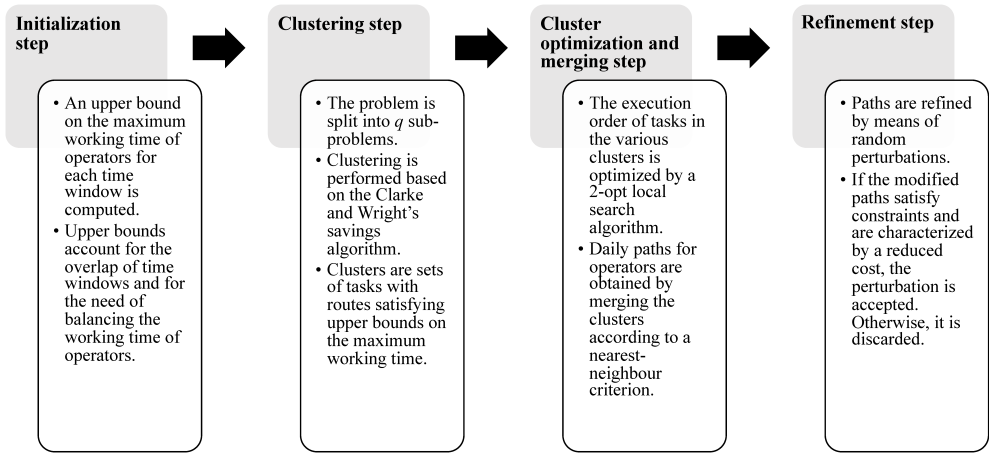



FIGURE 2 The four steps of the proposed heuristic approach.

where $W_k$ and $R$ are the duration of the time window $k$ and the maximum working time per day of operators introduced in Section 3, respectively.

However, upper bounds $\tilde{U}_k$ computed in (2) do not account for the possible overlap of time windows. In other words, it may happen that $\tilde{U}_k + \tilde{U}_{k+1} \geqslant W_k + W_{k+1} - V_k$ for a certain $k$, i.e., constraints (1n) and (1o) of VRPOTW may be violated. This happens especially in the presence of large numbers of tasks to be executed in the windows $k$ and $k+1$. In this case, the quantities $\tilde{U}_k$ are reduced proportionally to account for the overlapping nature of time windows. Otherwise, they are not modified. Thus, we introduce new bounds $U_k$, $k = 1, \ldots, q$, that are computed iteratively as follows:

$$U_k^{(\chi,\xi)} := \begin{cases} \dfrac{U_k^{(\chi-1,\xi)}}{\sum_{z=\xi}^{\chi} U_z^{(\chi-1,\xi)}} \left( \sum_{z=\xi}^{\chi} W_z - \sum_{z=\xi}^{\chi-1} V_z \right) & \text{if} \quad \sum_{z=\xi}^{\chi} U_z^{(\chi-1,\xi)} \geqslant \sum_{z=\xi}^{\chi} W_z - \sum_{z=\xi}^{\chi-1} V_z, \\ U_k^{(\chi-1,\xi)} & \text{otherwise,} \end{cases}$$

$$k = 1, \ldots, \chi, \quad \xi = 1, \ldots, q,$$

$$U_k^{(\chi,\xi)} := U_k^{(\chi-1,\xi)}, \quad k = \chi+1, \ldots, q, \quad \xi = 1, \ldots, q,$$

where $\chi = 2, \ldots, q$. The procedure is initialized with $U_k^{(1,1)} := \tilde{U}_k$ for $k = 1, \ldots, q$ and updated with $U_k^{(1,\xi)} := U_k^{(q,\xi-1)}$ for $k = 1, \ldots, q$ and $\xi = 2, \ldots, q$. Eventually, the assignment $U_k := U_k^{(q,q)}$, $k = 1, \ldots, q$, yields the final upper bounds. The upper bounds $U_k$ allow to distribute the workload of operators on the various time windows proportionally to the number of tasks to be performed. The smaller is the lower bound $U_k$, the lower is the time that is available for an operator to perform tasks belonging to the time window $k$, and therefore a higher number of operators will be required for that window. Hence, the upper bounds guarantee to assign tasks to operators in a balanced way over time windows, thus reducing the risk that an operator has too many tasks to be performed in a given window, while another one has very few operations to be executed within the same timeframe. Therefore, as previously pointed out, the initialization step also concur to pursue goal (ii) of VRPOTW described in Section 3.

**Algorithm 1** Four-step heuristic approach for the solution of VRPOTW

**Inputs:** $n$, $q$, $\tau_i$, $W_k$, $V_k$, $R$, $F_{kj}$, $S_{ij}$, $D_{ij}$, $T_{ij}$, $P$, $c_1$, $c_2$, $c_3$.
*// Initialization step //*
**for** $k = 1, \ldots, q$ **do**
    compute upper bounds $U_k$;
**end for**
*// Clustering step //*
**for** $k = 1, \ldots, q$ **do**
    compute Clarke and Wright's savings for tasks of window $k$;
    sort savings in descending order;
    **for** $z = 1, \ldots,$ number of savings **do**
        create routes starting from and returning to the company depot such that travel and execution times ↳ do not exceed $U_k$;
    **end for**
    group tasks belonging to the various routes in different clusters;
    $C_k \leftarrow$ total number of clusters of time window $k$;
**end for**
*// Cluster optimization and merging step //*
define an empty set of locked tasks;
**for** $k = 1, \ldots, q - 1$ **do**
    **for** $c = 1, \ldots, C_k$ **do**
        apply 2-opt to optimize the order of tasks of window $k$ and cluster $c$, with the exception of locked tasks;
    **end for**
    **for** $c$=1, ..., min($C_k$ +non-merged clusters of previous windows, $C_{k+1}$) **do**
        merge clusters by connecting the final node of a non-merged cluster of window $k$ with the task ↳ of window $k + 1$ with minimum distance belonging to a non-merged cluster;
        insert this task into the set of locked tasks;
    **end for**
**end for**
*// Refinement step //*
compute the cost of the current assignment of tasks to operators and label it as best cost;
**for** $p = 1, \ldots, P$ **do**
    randomly extract two paths and two operations within the selected paths;
    randomly select and apply a perturbation among swap, insertion, and crossing;
    compute the cost of the new assignment and check constraints satisfaction of the perturbed paths;
    **if** new cost is less than best cost **and** constraints are satisfied **then**
        accept perturbation and update paths and best cost;
    **end if**
**end for**
$m \leftarrow$ number of daily paths;
**Outputs:** $m$, daily paths for all the operators.

## 4.2 | Clustering step

The goal of the second step is to group tasks belonging to a certain time window in one or more sets, called clusters, which will be then merged in the third step with clusters of the next time windows in order to create daily paths for operators.

In more detail, the problem is split into $q$ sub-problems, one for each time window, and the Clarke and Wright's savings algorithm is applied by considering each window separately from the others. The algorithm consists in computing savings that may occur if two tasks are executed in sequence rather than starting from/returning to the company depot before/after the execution of each operation. For each time window, a certain number of routes are created by joining tasks sequentially based on a decreasing saving order. The creation of a route among tasks belonging to the window $k$ is feasible only if the total working time of the corresponding route, given by the sum of execution times of tasks and travel times, does not exceed the upper bound $U_k$ computed in the initialization step. The process terminates when no further feasible routes can be created. Each route starts from the depot and returns therein after visiting a certain number of customers' sites. The tasks of the various routes are then grouped into ordered sets of nodes called clusters.

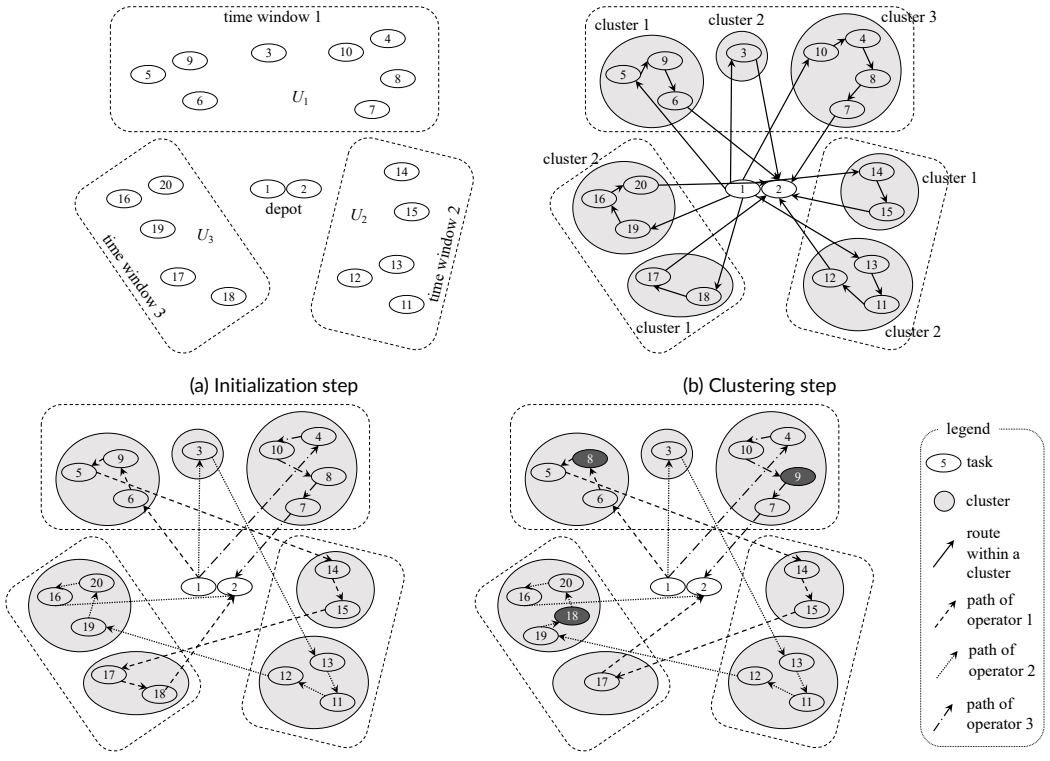
The amount of clusters of a time window defines the number of operators requested to perform all tasks belonging to it. In general, the number of clusters of a window may be different from the amount of clusters of another one, even if the upper bounds computed in the initialization step should lead to rather homogeneous numbers for all the windows. Hence, the initialization and clustering steps play a crucial role to determine the minimum required number of operators stated in goal (iv) of VRPOTW. This number is equal to the maximum number of clusters belonging to the various time windows.

## 4.3 | Cluster optimization and merging step

The goal of this step is to optimize the order of execution of tasks belonging to the various clusters identified in the previous step by means of a 2-opt local search algorithm, and then merge the clusters of a time window with those of another one to create daily paths for operators with the goal of minimizing the overall traveled distance, i.e., pursuing goal (i) of the considered problem. In particular, the "final" tasks of each cluster (preceding the return to the depot) are joined with the "best" tasks belonging to clusters of the next time windows, as detailed in the following.

Starting with the first time window, the order of execution of tasks belonging to each cluster is optimized via the 2-opt method. Hence, the 2-opt algorithm is used to improve part of the route of an operator. It is based on an edge-exchange criterion: starting from a given initial route within a cluster, the procedure considers each pair of tasks and reorders them to check whether the resulting route is shorter. The acceptance criterion is of the first-improvement type: the first obtained improvement is immediately carried out, i.e., tasks are reordered and the current route is modified. The algorithm stops when no further improvement can be obtained. The main advantage of the 2-opt algorithm lies in its very small computational effort, even if a local optimal solution may be found that highly depends on the initial one. After the application of the 2-opt algorithm to the first time window, the final nodes in the execution order of each optimized cluster are considered, and a merge with a task of the next time window is performed based on a nearest-neighbor criterion. In particular, for each final task, all the nodes of the clusters not yet merged belonging to the next time window are analyzed, and the best merge is selected as the one minimizing the distance between tasks.

It may happen that a cluster of the time window $k$ is not merged with a cluster of the window $k+1$, but with a cluster of the following ones (e.g., $k+2$, $k+3$, and so on). This happens if the number of clusters of the window $k$ is




(a) Initialization step

(b) Clustering step

(c) Cluster optimization and merging step

(d) Refinement step

FIGURE 3 The four steps of the proposed heuristic approach applied to a toy example with $n = 20$ tasks and $q = 3$ time windows. The initialization step (a) sketches all the tasks with the corresponding time windows and upper bounds $U_1$, $U_2$, and $U_3$ on the maximum working time of operators. The clustering step (b) results into a series of routes for each time window starting from the depot and returning therein. The clusters of different time windows are then merged in the third step (c). Lastly, the fourth step (d) implements a refinement of the solution via swaps, insertions, and crossings aimed at further minimizing the cost: in particular, the swap of tasks 8 and 9 in the first time window and the insertion of task 18 in the third one are performed (both moves are highlighted in dark gray).

greater than the number of clusters of the window $k + 1$. In this case, the final node of a non-merged cluster will be taken into account in the next merging phase, i.e., when considering the merging of final nodes of the window $k + 1$ with the tasks of the window $k + 2$.

The tasks of the time window $k + 1$ that are chosen to be merged with the final nodes of clusters of the window $k$ become the first nodes of the corresponding routes within clusters of the window $k + 1$ and are then "locked", i.e., they will remain the first ones in the execution order and will not be taken into consideration by the 2-opt search algorithm applied to the clusters belonging to the time window $k + 1$.

The procedure is iterated for all time windows: the first window is optimized by 2-opt and the second window is merged, the second window is optimized and the third one is merged, and so on. As a result, $m$ daily paths starting from the company depot and returning therein are obtained that are assigned to $m$ different operators.

### 4.4 | Refinement step

The aim of the refinement step is to improve the paths computed through the previous steps in order to further minimize the cost function (1a). In more detail, a random search starting from the current best solution is performed via the generation of a given number $P$ of perturbations. Perturbations may be of three different kinds: swaps, insertions, and crossings. In particular, for $p = 1, \ldots, P$, first two paths and two operations within the selected paths are randomly extracted. Then, a perturbation within swap, insertion, and crossing is randomly chosen and applied. The perturbation is accepted if the resulting assignment of tasks to operators reduces the cost as compared to the current best solution and satisfies all the constraints. In this case, the current best solution and best cost are updated accordingly. Otherwise, if the perturbed paths violate constraints or correspond to an assignment with a higher cost, they are discarded. Then, a new perturbation of the best solution is analyzed until the considered maximum number of moves is reached.

In more detail, the swap perturbation acts as follows. Given two paths $a$ and $b$ and two operations $i$ and $j$ belonging to $a$ and $b$, respectively, they are simply swapped, i.e., task $i$ of path $a$ is assigned as task $j$ of path $b$ and viceversa. Concerning insertion, given two paths $a$ and $b$ and two operations $i$ and $j$ belonging to $a$ and $b$, respectively, task $i$ is removed from path $a$ and inserted between tasks $j$ and $j + 1$ of path $b$. Lastly, let us focus on the crossing perturbation. Given two paths $a$ and $b$ and two operations $i$ and $j$ belonging to $a$ and $b$, respectively, tasks from $i + 1$ to the end of path $a$ are assigned to path $b$ after task $j$, while tasks from $j + 1$ to the end of path $b$ are assigned to path $a$ after task $i$.

## 5 | NUMERICAL RESULTS

The effectiveness of the proposed approach is evaluated by means of a case study coming from the real world, and real data are used for the various tests. In particular, we consider a company providing gas metering services for third parties in the urban area of Genoa, Italy. As previously pointed out, operations to be executed are of two types, i.e., activation and deactivation of gas meters, and at each customer site operators can do either the first or the second one, but not both together. Each task has a known execution time that depends on the type of operation. In particular, the execution times are equal to 30 and 15 min for the activation and deactivation tasks, respectively. We have a total number of $q = 6$ fixed time windows throughout the day, each one with duration equal to 2 h. Thus, we have $W_k = 2$ h for all $k = 1, \ldots, q$. There exists a lunch break around midday of 1 h. As a consequence, we have three windows in the morning and three windows in the afternoon, which overlap 1 h with each other. Hence, we have $V_1 = V_2 = 1$ h, $V_3 = 0$, and $V_4 = V_5 = 1$ h. In addition, we consider the fictitious windows 0 and 7 to model the departure from and return to the company depot at the beginning and at the end of the day, respectively. Each operator has a daily maximum working time $R = 8$ h, from 8 am to 12 pm and from 1 pm to 5 pm (the lunch break is from 12 pm to 1pm). A sketch of time windows where the various tasks are performed is reported in Figure 4.

To evaluate performance and scalability of the proposed FSH approach, we have considered several instances of VRPOTW: the approximate solutions computed by the FSH have been compared with the exact solutions obtained by solving (1) for all the considered instances. For the sake of brevity, in the following we will refer to such approach as "optimization-based method", or OPT for short. Optimization has been performed with the mixed-integer programming solver available in CPLEX and with a time limit of 24 h. Instead, the heuristic technique has been implemented in Matlab. The effectiveness of the proposed approach has been tested also against a simple, ad-hoc developed heuristic technique based on the paradigm of iterated local search (ILS, for short). Starting from a feasible initial solution, this approach iteratively modifies it through perturbations, i.e., relocation of customers within routes, deletion of arcs, and

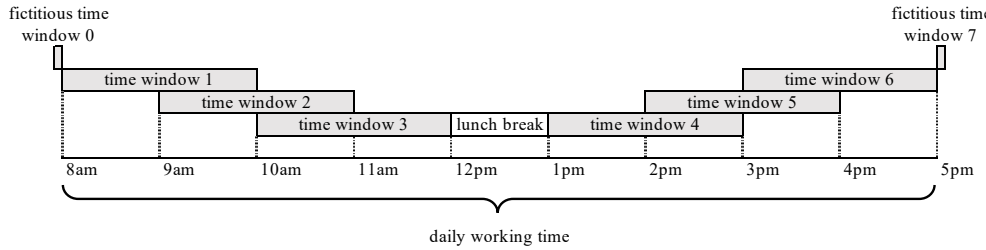



FIGURE 4 Time windows in the considered case study.

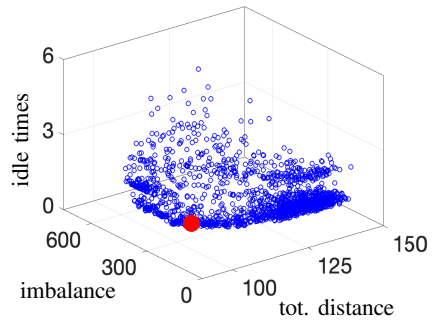



FIGURE 5 Solutions of the instance with $n = 30$ tasks of the VRPOTW corresponding to different values of the weighting coefficients of the cost function (blue points) and solution chosen to fix the coefficients (bold, red point).

replacement with other ones with the goal of minimizing the objective function (1a). In more detail, the initial solution is generated starting from the list of tasks to be performed, sorted by time window and execution time. Tasks are assigned sequentially to operators starting from the sorted list in such a way to satisfy all the constraints (1b)–(1q). Then, the initial solution is improved via a number of perturbations, which consist in iteratively swapping, inserting, and crossing nodes or parts of routes. In more details, the same perturbations detailed in Section 4.4 are considered, i.e., swap, insertion, and crossing. For all the combinations of two paths in the set of all paths, and for all the pairs of tasks belonging to the cross-product of tasks belonging to the selected paths, the three perturbation moves swap, insertion, and crossing are applied. Each move is accepted if the change in the paths corresponds to a descent of the cost (1a) and constraints are satisfied, while moves entailing infeasibility or an increase of the cost are discarded.

The comparison has been performed by evaluating separately the various goals (i)–(iv): the total distance traveled by operators (the first term in the cost function in (1a)), the imbalance in the working time of operators within the same time window (the second term in the cost (1a)), i.e., the difference between working time of operators in a time window and the average working time of operators within the same window, the presence of idle times of operators (the case of time windows with no tasks to be executed between two windows with at least one operation to be performed, as accounted for in the third term in (1a)), and the minimum number of operators required to perform all the activities (the number $m$ in the exact formulation of VRPOTW). Then, we have focused on a condensed performance indicator that allows evaluating the effectiveness of the various approaches "at a glance". Such an indicator, which will be called overall performance index (OPI), is given by the linear combination (with coefficients $c_1$, $c_2$, and $c_3$) of the corresponding terms in the optimal cost of VRPOTW. We have evaluated also the computational requirements of the OPT, FSH, and ILS approaches by comparing the time needed to find a planning for the entire day, referred to as

computation time in the following.

The coefficients $c_1$, $c_2$, and $c_3$ of the cost function (1a) have been chosen equal to 1, 0.1, and 10, respectively. Such values have been fixed via the numerical computation of the three-dimensional Pareto front related to VRPOTW. In fact, the cost (1a) can be viewed as a scalarized version of a multiobjective optimization problem. Figure 5 reports as blue points the solutions of the instance with $n = 30$ tasks of VRPOTW corresponding to about 2000 different values of the coefficients. The chosen values of the coefficients, highlighted with a larger, red point, correspond to a non-dominated solution. Among the various points belonging the Pareto front, the choice of the one related to $c_1 = 1$, $c_2 = 0.1$, and $c_3 = 10$ has been guided by some managerial considerations (see later on for additional details). In a nutshell, the imbalance is the term in the objective function with the lowest impact on the costs of the gas metering company, and for this reason the corresponding coefficient $c_2$ has been fixed to a smaller value. The selection of other points in the Pareto front, although possible, has not been taken into account since it does not correspond to solutions that are significant from the practical viewpoint.

The value of $M$ in (1a) and (1q) must be chosen sufficiently large. After a trial-and-error investigation, we have chosen $M = 1000$ since we have observed that larger values of $M$ may cause numerical issues while lower ones may cause infeasibility. A detailed investigation on the lower bounds of such a quantity ensuring feasibility is outside the scope of this work.

Concerning the FSH approach, the only parameter to be tuned, i.e., the number $P$ of refinements in the last step, has been fixed equal to 1000. This value has been selected after a trial-and-error procedure aimed at balancing effectiveness and computational requirements. In more details, we verified that the selection of larger values was useless as regards performance improvements but entailed only increased computational requirements.

Two rounds of tests have been performed for performance evaluation. In the first one, we have focused on ten real instances of VRPOTW with an increasing number of tasks to be executed: $n = 10, 20, 30, 40, 50, 60, 70, 80, 90$, and 100. The goal is to assess the scalability properties of the proposed approach and to enable a comparison with the optimal solution provided the OPT method, which, as we will see in the following, can be found only for small values of $n$. In practice, the standard number of gas meter maintenance tasks to be executed for each day ranges from 80 to 100 on the average, hence the maximum number of operations considered in this paper. The operations included in the smaller instances of VRPOTW from $n = 10$ to $n = 70$ are subsets of the daily workload of the company in a standard business day. Clearly, only the cases with $n = 80$, $n = 90$, and $n = 100$ have an importance from the practical point of view, while the lower-dimensional instances are much less interesting. In fact, in the case of $n = 100$, the average number of tasks to be executed per each time window is about 16, while in the case of $n = 10$ or $n = 20$ it is equal to about 2 or 3. Clearly, no practical significance can be given to the case of 2 or 3 interventions per time window, but the investigation of the performances is important to check effectiveness in comparison with the optimization-based method, which is unable to find a solution for larger numbers of tasks. Thus, in the first round of tests, matrices $F$, $S$, $D$, and $T$ introduced in Section 3 are filled with real data. The second round of tests is devoted to assess performances on many different synthetic instances of VRPOTW. In more detail, starting from the real data used in the previous round of tests, we have generated 100 additional instances for each value of $n$ obtained through random perturbations of the customers' location where the various tasks have to be executed.

The results obtained in both cases are showcased and discussed in the following sections. All the simulations have been performed on a computer equipped with a 3.7 GHz Intel i7 CPU with 32 GB of RAM.

TABLE 2 Summary of the numerical results obtained in the first round of tests.

| | Num. tasks | Total traveled distance [km] | Imbalance [min] | Num. idle times | Num. operators | Overall perf. index | Gap [%] distance | Gap [%] imbalance | Gap [%] idle times | Gap [%] operators | Gap [%] OPI | Comput. time [s] |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| OPT | 10 | 35.15 | 0.00 | 0 | 1 | 35.15 | - | - | - | - | - | 0.039 |
| | 20 | 69.72 | 127.64 | 0 | 2 | 82.48 | - | - | - | - | - | 2.81 |
| | 30 | 106.80 | 114.45 | 1 | 3 | 128.25 | - | - | - | - | - | 971.89 |
| | 40 | 112.00 | 253.02 | 1 | 3 | 147.30 | - | - | - | - | - | 86400.00 |
| | 50 | - | - | - | - | - | - | - | - | - | - | - |
| | 60 | - | - | - | - | - | - | - | - | - | - | - |
| | 70 | - | - | - | - | - | - | - | - | - | - | - |
| | 80 | - | - | - | - | - | - | - | - | - | - | - |
| | 90 | - | - | - | - | - | - | - | - | - | - | - |
| | 100 | - | - | - | - | - | - | - | - | - | - | - |
| FSH | 10 | 35.15 | 0.00 | 0 | 1 | 35.15 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.031 |
| | 20 | 69.72 | 127.64 | 0 | 2 | 82.48 | 0.00 | 0.00 | 0.00 | 0.00 | 0.00 | 0.25 |
| | 30 | 102.42 | 338.29 | 0 | 3 | 136.25 | -4.10 | 195.58 | -100.00 | 0.00 | 6.24 | 0.31 |
| | 40 | 125.90 | 220.27 | 0 | 3 | 147.93 | 12.41 | -12.94 | -100.00 | 0.00 | 0.42 | 0.66 |
| | 50 | 148.15 | 369.57 | 0 | 4 | 185.11 | - | - | - | - | - | 0.88 |
| | 60 | 172.17 | 232.63 | 0 | 5 | 195.43 | - | - | - | - | - | 2.65 |
| | 70 | 193.27 | 372.71 | 0 | 6 | 230.54 | - | - | - | - | - | 2.82 |
| | 80 | 200.27 | 621.71 | 0 | 7 | 262.44 | - | - | - | - | - | 4.62 |
| | 90 | 247.01 | 467.10 | 1 | 7 | 303.72 | - | - | - | - | - | 4.71 |
| | 100 | 265.19 | 658.07 | 0 | 8 | 331.00 | - | - | - | - | - | 5.12 |
| ILS | 10 | 37.30 | 0.00 | 0 | 1 | 37.30 | 6.12 | 0.00 | 0.00 | 0.00 | 6.12 | 0.063 |
| | 20 | 84.16 | 102.72 | 0 | 2 | 94.43 | 20.71 | -19.52 | 0.00 | 0.00 | 14.49 | 0.28 |
| | 30 | 111.21 | 252.55 | 1 | 3 | 146.47 | 4.13 | 120.66 | 0.00 | 0.00 | 14.21 | 1.09 |
| | 40 | 148.31 | 110.45 | 0 | 3 | 159.36 | 32.42 | -56.35 | -100.00 | 0.00 | 8.18 | 2.50 |
| | 50 | 158.34 | 288.10 | 0 | 4 | 187.15 | - | - | - | - | - | 4.87 |
| | 60 | 185.64 | 213.44 | 0 | 5 | 206.98 | - | - | - | - | - | 8.94 |
| | 70 | 219.27 | 279.80 | 0 | 6 | 247.25 | - | - | - | - | - | 14.38 |
| | 80 | 248.72 | 387.23 | 0 | 7 | 287.44 | - | - | - | - | - | 30.51 |
| | 90 | 278.69 | 825.85 | 4 | 9 | 401.27 | - | - | - | - | - | 57.81 |
| | 100 | 288.77 | 921.56 | 2 | 10 | 400.93 | - | - | - | - | - | 84.04 |

## 5.1 | First round of tests: simulations on real instances of VRPOTW

Table 2 contains a summary of the numerical results obtained in the first round of tests with an increasing number of tasks to be executed. In particular, the total distance traveled by operators, the total imbalance over time windows, the total number of idle times for operators, the required number of operators, and the OPI are reported. Moreover, for the case of the FSH and ILS approaches, the table also contains, for each index, the percentage gap with respect to the corresponding indicator provided by the OPT method. The last column showcases the computation time required to find a solution for all the considered techniques. A sketch of the behaviors of the OPI and of the computation time with respect to the number of tasks is reported in Figure 6.

Looking at the simulation results, we note an increasing trend of the total traveled distance, total imbalance, amount of idle times, required number of operators, and overall performance index with respect to the number of tasks to be performed, as expected. The FSH and ILS approaches are able to find a solution for all the instances of

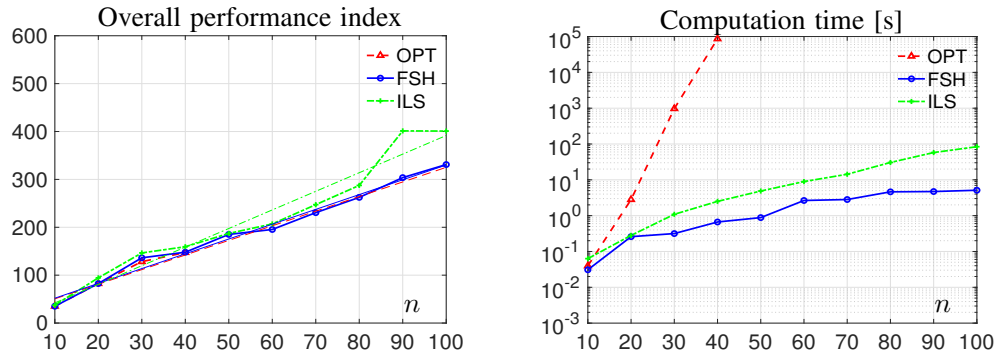



FIGURE 6 Overall performance index (thin lines without markers represent a linear fitting of data) and computation time (in logarithmic scale) for increasing number of tasks obtained in the first round of tests.

VRPOTW from $n = 10$ up to $n = 100$. In the case of FSH, a computation time ranging from few milliseconds to few seconds is needed to obtain an approximate solution, while the ILS technique requires at most about 80 seconds to find a solution in the largest instance of the problem. On the contrary, the true optimal solution can be found by the OPT approach only up to $n = 40$ tasks, but with an exponential increase of the required computational effort. This impedes finding a solution for $n$ greater than 40.

Concerning the comparison between the FSH and the OPT methods, the same solution is found for $n = 10$ and $n = 20$ tasks, while the instance with $n = 30$ operations is characterized by the largest gap between the two methods. In more detail, the solution provided by FSH has a lower value for the total traveled distance, but an increased imbalance. Overall, the gap in term of the OPI with respect to the optimization-based approach is equal to about 6%. As regards the largest instance that can be solved with the OPT method, i.e., the case $n = 40$, the overall gap between FSH and OPT is less than 0.5%, thus confirming the effectiveness of the former to find a suboptimal solution. Such a solution is characterized by an increased value of the total traveled distance, but a saving of the imbalance and a reduction of the number of idle times.

On the average, from $n = 10$ to $n = 40$, an increase of about 1.5% of the OPI is obtained by using the proposed heuristic method for all the values of $n$ such that we can find an optimal solution (see the column "Gap OPI" in Table 2). A saving of 5 orders of magnitude in the computation time is experienced for the largest instance of the problem that can be solved through optimization, i.e., for $n = 40$. As regards the various goals (i)-(iv), we note that both the FSH and OPT approaches require the same number of operators, and therefore the corresponding gaps in Table 2 are equal to zero. The FSH always overcomes the OPT in terms of minimization of the number of idle times, as no idle times are experienced from $n = 10$ to $n = 40$. An average gap of about 2% is experienced for the total distance traveled by operators, while we can observe a higher variability as concerns the total imbalance. In particular, for $n = 40$, the FSH guarantees a reduced imbalance as compared to the OPT approach, which in turn provides quite a reduced traveled distance as compared to FSH. For $n = 30$, the gap on the total imbalance is larger. Such results suggest that there may be possible margins to improve the proposed heuristic as regards the imbalance, for instance by acting on the initialization step.

However, from a managerial point of view, the imbalance is the objective term with the lowest impact on company costs. As previously pointed out, the imbalance has been introduced in the objective function due to labor agreements requiring to divide the overall workload as fairly as possible among the available operators, but it does not have a direct impact on company expenditures. Instead, the gap on the total distance between the FSH and the OPT approaches

is reduced and the difference in terms of company disbursements of the two assignments is practically negligible. To fix ideas, let us consider the case of $n = 40$ tasks: if we consider an all-inclusive cost of fuel and vehicle wear of about 0.6 €/km, the difference in the total traveled distance results into an increase of less than 10 € per day. The most impacting components of the objective function on company savings and efficiency are the number of required operators and their idle times. In the presence of idle times between time windows, operators cannot perform any technical activities on meters (activation and deactivation) and therefore they are unproductive for the gas metering company: the company bears the costs of operators for idle time slots without an economic return in terms of incomes. In this case, the FSH method guarantees no idle times for the problem instances from $n = 10$ to $n = 40$, whereas the OPT solution suffers from the presence on one idle time for both the instances with $n = 30$ and $n = 40$ tasks. Therefore, the performances of the proposed approach are very satisfactory for the gas metering company.

As regards the comparison between FSH and ILS, we observe that performances of the former are much better than the latter for all the considered indicators, with the exception of the imbalance. The OPI of the ILS is larger in general, especially for the largest instances of the problem with $n = 90$ and $n = 100$. The worst indicators are given by the total traveled distance and by the required number of operators, which are greater than the corresponding values of the FSH technique.

Summarizing, satisfactory results are obtained by using the FSH approach, as a reduced decay of performance is obtained as compared to the OPT, but a huge saving of the required effort is guaranteed. The linear extrapolation of the trend of the OPI in Figure 6 (the thin lines without markers represents a linear fitting of the available data) for the OPT method confirms that the proposed heuristic approach can be successfully used to solve high-dimensional instances of VRPOTW with negligible computation time. The comparison with the ILS heuristic is successful, as better results are provided for the performance indicators with the largest impact on the costs of the gas metering company, and a reduced computational effort is required to find an approximate solution.

## 5.2 | Second round of tests: simulations on many synthetic instances of VRPOTW

As previously pointed out, the second round of tests is devoted to assess performances on many different instances of VRPOTW. In more detail, starting from the real data used in the first round of tests, we have generated 100 synthetic instances for each value of $n$ obtained through random perturbations of customers' location. Figure 7 contains the boxplots of the OPI provided by the OPT, FSH, and ILS approaches over the considered 100 instances.

Likewise in the first round of tests, an optimal solution is found by the OPT method only up to $n = 40$ tasks, whereas the FSH and ILS methods are able to find a solution for $n$ up to 100 in all the considered 100 synthetic instances. In more detail, an increasing trend of the OPI with the number of tasks can be observed, but the gap between the OPT and the FSH approach is reduced and similar to the values obtained in the first round of tests when using real data. The OPT approach guarantees the lowest dispersion around the median values, while a higher dispersion is provided by the FSH method for $n = 30$ or $n = 40$. However, for larger values of $n$, the dispersion around the median of FSH remains more or less constant and much lower as compared to the one guaranteed by ILS, which is again the worst method.

Figure 8 depicts the medians of the OPI provided by the OPT, FSH, and ILS approaches (the triangle, circle, and plus markers, respectively) together with their linear fitting. It turns out that the difference between the OPT and FSH values remains almost constant for increasing values of $n$, likewise in the first round of tests, whereas the gap between FSH and ILS grows with $n$.

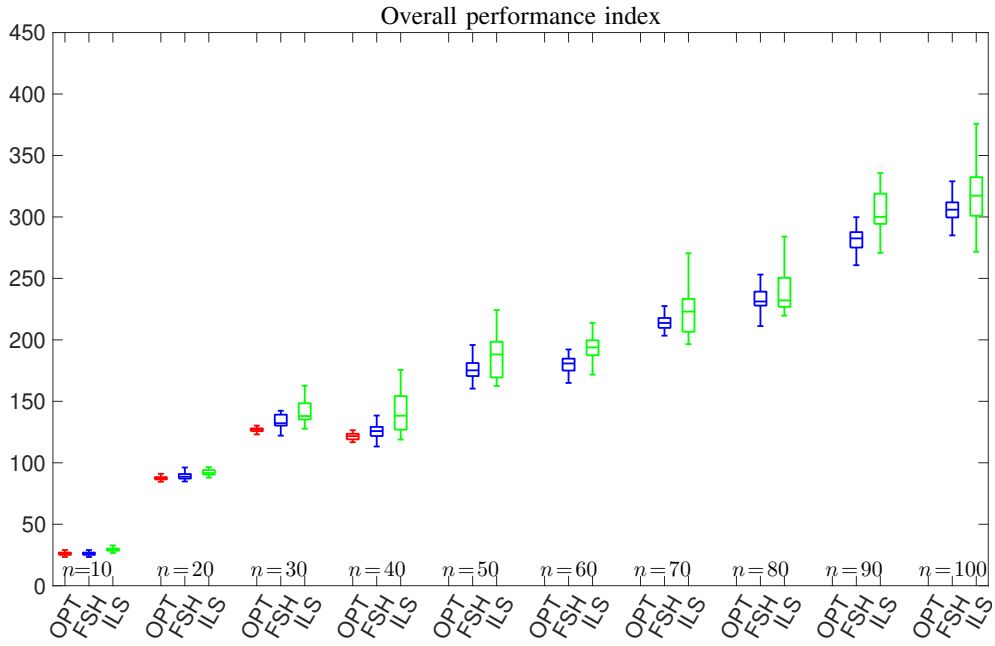



FIGURE 7 Boxplots of the overall performance index over 100 simulations obtained in the second round of tests.

## 6 | CONCLUSIONS

In this paper, we have presented an approach to optimize maintenance services of a company working in the gas metering sector for third parties by exploiting the paradigm of vehicle routing problems with time windows. In more detail, the considered problem involves a set of predefined time windows that overlap one with the others and four competing objectives that have to be taken into account. First, an integer programming formulation of the problem has been devised. However, finding the optimal solution is possible only for small instances. Thus, we have developed a heuristic approach that is able to find approximate solutions with a reduced computational effort, also for a large number of nodes to be visited. The heuristic is made up of four steps, and combines the savings algorithm of Clarke and Wright, the 2-opt local search, and a nearest-neighbor criterion. Successful numerical results using both data coming from the real world and multiple synthetic instances have confirmed the effectiveness of the proposed approach. An important advantage of the developed heuristic, which does not turn out explicitly from numerical results, is its simplicity. In particular, its performances do not depend on many parameters that have to be properly tuned before obtaining satisfactory results, as it often happens for other heuristics commonly used in vehicle routing problems. This makes the proposed approach interesting also for practitioners that are non-experts in complex tuning procedures.

As pointed out in the Introduction, this is the first work dealing with the optimization of gas metering services by pursuing the considered four conflicting goals. Thus, several directions of future investigation can be outlined. The main efforts will be devoted to refining the proposed FSH approach, for instance by using other methods than the Clarke and Wright savings algorithm and the 2-opt in the second and third step, respectively. In more detail, we will investigate the effect of introducing a “working area” for operators to improve the Clarke and Wright approach used in the clustering step and avoid starting/returning from/to the depot for each path. A special attention will be devoted

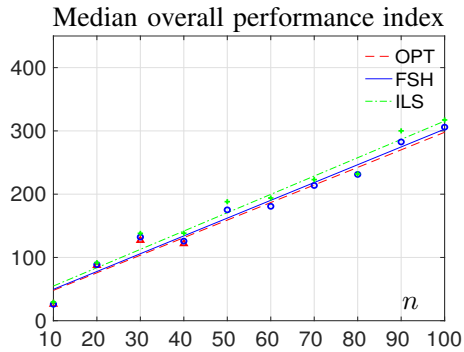



FIGURE 8 Medians of the overall performance index over 100 simulations obtained in the second round of tests. The thin lines represent a linear fitting of the available data.

also to the initialization step, as the obtained numerical results have indicated that the largest gaps with respect to the optimization-based approach are in the imbalance component of the objective function. Further, we will investigate whether the insertion of additional decision variables accounting for the time scheduling of the various operations may help to better make use of all the amount of time within a window, for instance by exploiting idle times to travel among customer locations. Moreover, the effectiveness of the proposed approach in low-dimensional instances of the problem will be evaluated also in comparison with alternative approaches, such as, for instance, dynamic programming, while larger instances will be compared with new heuristic methods that are more sophisticated than the considered one based on the iterated local search paradigm. Lastly, we plan to test the proposed approach in other real case studies, introduce stochasticity, for instance by considering travel and execution times as random variables with suitable probability distributions, as well as account for the case of flexible and multiple time windows.

## References

[1] F.S. Alves, J.M. de Souza, and A.H. Costa, *Multi-objective design optimization of natural gas transmission networks*, Comput. Chemical Eng. **93** (2016), 212–220.

[2] N. Azi, M. Gendreau, and J.Y. Potvin, *An exact algorithm for a vehicle routing problem with time windows and multiple use of vehicles*, Eur. J. Oper. Res. **202** (2010), 756–763.

[3] R. Baldacci, E. Bartolini, and A. Mingozzi, *An exact algorithm for the pickup and delivery problem with time windows*, Oper. Res. **59** (2011), 414–426.

[4] R. Baldacci, A. Mingozzi, and R. Roberti, *Recent exact algorithms for solving the vehicle routing problem under capacity and time window constraints*, Eur. J. Oper. Res. **218** (2012), 1–6.

[5] V. Baradaran, A. Shafaei, and A.H. Hosseinian, *Stochastic vehicle routing problem with heterogeneous vehicles and multiple prioritized time windows: Mathematical modeling and solution approach*, Comput. Indus. Eng. **131** (2019), 187–199.

[6] H. Behrooz and R. Boozarjomehry, *Dynamic optimization of natural gas networks under customer demand uncertainties*, Energy **134** (2017), 968–983.

[7] S. Belhaiza, R. M'Hallah, G. Brahim, and G. Laporte, *Three multi-start data-driven evolutionary heuristics for the vehicle routing problem with multiple time windows*, J. Heuristics **25** (2019), 485–515.

[8] K. Braekers, K. Ramaekers, and I.V. Nieuwenhuyse, *The vehicle routing problem: State of the art classification and review*, Comput. Indus. Eng. **99** (2016), 300–313.

[9] J. Bramel and D. Simchi-Levi, *On the effectiveness of set covering formulations for the vehicle routing problem with time windows*, Oper. Res. **45** (1997), 95–301.

[10] O. Braysy and M. Gendreau, *Vehicle routing problem with time windows, part I: Route construction and local search algorithms*, Transp. Sci. **39** (2005), 104–118.

[11] J. Chen and J. Shi, *A multi-compartment vehicle routing problem with time windows for urban distribution – A comparison study on particle swarm optimization algorithms*, Comput. Indus. Eng. **133** (2019), 95–106.

[12] T.C. Chiang and W.H. Hsu, *A knowledge-based evolutionary algorithm for the multiobjective vehicle routing problem with time windows*, Comput. Oper. Res. **45** (2014), 25–37.

[13] G. Clarke and J. Wright, *Scheduling of vehicles from a central depot to a number of delivery points*, Oper. Res. **12** (1964), 568–581.

[14] L. Costa, C.Contardo, and G. Desaulniers, *Exact branch-price-and-cut algorithms for vehicle routing*, Transp. Sci. **43** (2019), 946–985.

[15] G. Croes, *A method for solving traveling salesman problems*, Oper. Res. **6** (1958), 791–812.

[16] K. Doerner, M. Gronalt, R. Hartl, G. Kiechle, and M. Reimann, *Exact and heuristic algorithms for the vehicle routing problem with multiple interdependent time windows*, Comput. Oper. Res. **35** (2008), 3034–3048.

[17] N. El-Sherbeny, *Vehicle routing with time windows: An overview of exact, heuristic and metaheuristic methods*, J. King Saud University **22** (2010), 123–131.

[18] T. Faiz, C. Vogiatzis, and M. Noor-E-Alam, *A column generation algorithm for vehicle scheduling and routing problems*, Comput. Indus. Eng. **130** (2019), 222–236.

[19] M. Fisher, *Optimal solution of vehicle routing problem using minimum K-trees*, Oper. Res. **42** (1994), 626–642.

[20] T. Gschwind, S. Irnich, C. Tilk, and S. Emde, *Branch-cut-and-price for scheduling deliveries with time windows in a direct shipping network*, J. Scheduling **23** (2020), 363–377.

[21] A. Gutierrez, L. Dieulle, N. Labadie, and N. Velasco, *Heuristics for the robust vehicle routing problem with time windows*, Comput. Indus. Eng. **125** (2018), 144–156.

[22] A. Gutierrez-Rodriguez, S. Conant-Pablos, J. Ortiz-Bayliss, and H. Terashima-Marin, *Selecting meta-heuristics for solving vehicle routing problems with time windows via meta-learning*, Expert Syst. With Appl. **118** (2019), 470–481.

[23] M. Hamedi, R. Farahani, and G. Esmaeilian, "*Optimization in natural gas network planning*," *Logistics Operations and Management - Concepts and Models*, R. Zanjirani, S. Farahani, and L. Kardar (eds.), Elsevier Insights, London, UK, 2011, pp. 393–420.

[24] H. Hashimoto, M. Yagiura, and T. Ibaraki, *An iterated local search algorithm for the time-dependent vehicle routing problem with time windows*, Discr. Optim. **5** (2008), 434–456.

[25] P. Hungerlander and C. Truden, *Efficient and easy-to-implement mixed-integer linear programs for the traveling salesperson problem with time windows*, Transp. research procedia **30** (2018), 157–166.

[26] P. Kilby, P. Prosser, and P. Shaw, *Guided Local Search for the Vehicle Routing Problem with Time Windows* pp. 473–486, Springer US Boston, MA 1999.

[27] C. Koc, T. Bektas, O. Jabali, and G. Laporte, *A hybrid evolutionary algorithm for heterogeneous fleet vehicle routing problems with time windows*, Comput. Oper. Res. **64** (2015), 11–27.

[28] N. Kohl and O. Madsen, *An optimization algorithm for the vehicle routing problem with time windows based on Lagrangian relaxation*, Oper. Res. **45** (1997), 395–406.

[29] A. Kolen, A.R. Kan, and H. Trienekens, *Vehicle routing with time windows*, Oper. Res. **35** (1987), 266–273.

[30] I. Kucukoglu, R. Dewil, and D. Cattrysse, *Hybrid simulated annealing and tabu search method for the electric travelling salesman problem with time windows and mixed charging rates*, Expert Syst. With Appl. **134** (2019), 279–303.

[31] S.W. Lin and V. Yu, *A simulated annealing heuristic for the team orienteering problem with time windows*, Eur. J. Oper. Res. **217** (2012), 94–107.

[32] D. Lu and F. Gzara, *The robust vehicle routing problem with time windows: Solution by branch and price and cut*, Eur. J. Oper. Res. **275** (2019), 925–938.

[33] Y. Marinakis, M. Marinaki, and A. Migdalas, *A multi-adaptive particle swarm optimization for the vehicle routing problem with time windows*, Informat. Sci. **481** (2019), 311–329.

[34] M. Mikolajkova, C. Haikarainen, H. Saxen, and F. Pettersson, *Optimization of a natural gas distribution network with potential future extensions*, Energy **125** (2017), 848–859.

[35] M. Mikolajkova, H. Saxen, and F. Pettersson, *Linearization of an MINLP model and its application to gas distribution optimization*, Energy **146** (2018), 156–168.

[36] C. Miller, A. Tucker, and R. Zemlin, *Integer programming formulations and traveling salesman problems*, J. Assoc. Comput. Machinery **7** (1960), 326–329.

[37] D. Miranda and S. Conceicao, *The vehicle routing problem with hard time windows and stochastic travel and service time*, Expert Syst. With Appl. **64** (2016), 104–116.

[38] D.M. Miranda, J. Branke, and S.V. Conceicao, *Algorithms for the multi-objective vehicle routing problem with hard time windows and stochastic travel time and service time*, Appl. Soft Comput. **70** (2018), 66–79.

[39] M. Paolucci, D. Anghinolfi, and F. Tonelli, *Field services design and management of natural gas distribution networks: A class of vehicle routing problem with time windows approach*, Int. J. Production Res. **56** (2018), 1154–1170.

[40] R. Perez-Rodriguez and A. Hernandez-Aguirre, *A hybrid estimation of distribution algorithm for the vehicle routing problem with time windows*, Comput. Indus. Eng. **130** (2019), 75–96.

[41] K. Tan, C. Cheong, and C. Goh, *Solving multiobjective vehicle routing problem with stochastic demand via evolutionary computation*, Eur. J. Oper. Res. **177** (2007), 813–839.

[42] D. Tas, O. Jabali, and T.V. Woensel, *A vehicle routing problem with flexible time windows*, Comput. Oper. Res. **52** (2014), 39–54.

[43] P. Toth and D. Vigo, *The Vehicle Routing Problem*, SIAM, Philadelphia, PA, 2002.

[44] T. Vidal, T. Crainic, M. Gendreau, and C. Prins, *A hybrid genetic algorithm with adaptive diversity management for a large class of vehicle routing problems with time-windows*, Comput. Oper. Res. **40** (2013), 475–489.

[45] C. Wang, D. Mu, F. Zhao, and J. Sutherland, *A parallel simulated annealing method for the vehicle routing problem with simultaneous pickup–delivery and time windows*, Comput. Indus. Eng. **83** (2015), 111–122.

[46] M. Yu and X. Qi, *A vehicle routing problem with multiple overlapped batches*, Transp. Res. Part E **61** (2014), 40–55.

[47] H. Zhang, Q. Zhang, L. Ma, Z. Zhang, and Y. Liu, *A hybrid ant colony optimization algorithm for a multi-objective vehicle routing problem with flexible time windows*, Informat. Sci. **490** (2019), 166–190.

[48] J. Zhang, F. Yang, and X. Weng, *An evolutionary scatter search particle swarm optimization algorithm for the vehicle routing problem with time windows*, IEEE Access **6** (2018), 63468–63485.

[49] W. Zhang, D. Yang, G. Zhang, and M. Gen, *Hybrid multiobjective evolutionary algorithm with fast sampling strategy-based global search and route sequence difference-based local search for VRPTW*, Expert Syst. With Appl. **145** (2020), 1–16, art.no. 113151.

[50] Y. Zhou and J. Wang, *A local search-based multiobjective optimization algorithm for multiobjective vehicle routing problem with time windows*, IEEE Syst. J. **9** (2015), 1100–1113.

[51] J. Zhoua, J. Peng, G. Liang, and T. Deng, *Layout optimization of tree-tree gas pipeline network*, J. Petroleum Sci. Eng. **173** (2019), 666–680.