Solution of a Practical Vehicle Routing Problem for Monitoring Water Distribution Networks

Reza Atefi[(1)], Manuel Iori[(2)], Majid Salari[(1)] and Dario Vezzali[(3)]

(1) Department of Industrial Engineering, Ferdowsi University of Mashhad,
P.O. Box 91779-48951, Mashhad, Iran
atefi@mail.um.ac.ir, msalari@um.ac.ir
(2) Department of Sciences and Methods for Engineering,
University of Modena and Reggio Emilia,
Via Amendola 2, 42122, Reggio Emilia, Italy
manuel.iori@unimore.it
(3) "Marco Biagi" Foundation, University of Modena and Reggio Emilia,
Largo Marco Biagi 10, 41121, Modena, Italy
dario.vezzali@unimore.it

Abstract

In this work, we introduce a generalization of the well-known Vehicle Routing Problem for a specific application in the monitoring of a Water Distribution Network (WDN). In this problem, multiple technicians must visit a sequence of nodes in the WDN and perform a series of tests to check the quality of water. Some special nodes (i.e., wells) require technicians to first collect a key from a key center. The key must then be returned to the same key center after the test has been performed, thus introducing precedence constraints and multiple visits in the routes. To solve the problem, three mathematical models and an Iterated Local Search have been implemented. The efficiency of the proposed methods is demonstrated by means of extensive computational tests on randomly created instances, as well as on instances derived from a real-world case study.

Keywords: OR in Service Industries; Vehicle Routing Problem; Water Distribution Networks; Mathematical Modeling; Iterated Local Search.

1 Introduction

Water contamination is related to the presence of one or more chemical compounds or pathogens to the extent that they become dangerous to the consumer and might lead to diseases (Naserizade et al., 2018). The risk of accidental contamination of

drinking water is a well-known issue, and, recently, concerns regarding the deliberate contamination of urban water networks have called for additional safeguards.

In general, any threat to urban water networks directly affects the users in the community (Razali, 2015). Indeed, according to a report recently released by the *World Health Organization*, contaminated drinking water is estimated to cause 485,000 diarrhoeal deaths each year (World Health Organization, 2019). The safety of water distribution networks has always been an important issue for the communities. However, many distribution systems in cities around the world face the threat of accidental or intentional contamination during the transportation from treatment plants to consumers due to reverse flows (i.e., the return of contaminated water flows from facilities), old infrastructures, insufficient use of disinfectants, and so forth. Consequently, water contamination in distribution networks is considered as the most diffused cause behind the spread of water-borne diseases (Moon et al., 2002).

In recent years, several studies have been conducted to identify the main sources of water pollution and improve the quality of water thanks to innovative treatment methods and plants, but still an accidental event, such as a large-scale contamination or a destructive attack to the transmission system, can significantly affect both the economy and the society. In 2014, for example, 300,000 consumers in West Virginia were affected by the accidental contamination of their drinking water distribution system caused by 4-Methylcyclohexanemethanol (Seok Jeong and Abraham, 2006). During the same year, as reported by Mukherjee et al. (2017), a spill of benzene from a chemical plant in China accidentally reached the water distribution network. More recently, 27,000 Norwegian consumers were exposed to water contaminated with Clostridium (de Winter et al., 2019).

Supply, treatment, transmission and distribution of drinking water in urban distribution networks require substantial expenses; therefore, not only water in urban distribution networks is considered an essential resource, but also an economic commodity. The results of a study conducted by the *World Bank* show that nearly 15% of treated water is wasted annually in developed countries. This amount arises to a range of 35-60% for developing countries (Zhang et al., 2016). Timely control of *Water Distribution Networks* (WDNs) is thus of fundamental importance, both from an economical and public health point of view.

In this paper, a new variant of the well-known *Vehicle Routing Problem* (VRP) in the context of WDNs is proposed. In this problem, a set of technicians must visit a set of nodes, including wells, reservoirs and treatment plants, within a network to evaluate the water quality. When visiting a well, the technicians need a key to open the well and perform the required tests. Since the technicians do not have the key, they have to visit a specified node at which the key is located, called *key center* in the following, to acquire it. As a result, they need to visit this node before reaching the well. After the tests have been performed, they have to take the key back to its original key center before returning to the depot where they started their route.

In addition to that, it is imposed that all nodes are visited and that the duration of any route performed by a technician does not exceed a maximum traveling time. The aim of the problem is to minimize the sum of the traveled times.

The problem originates from a real-world application that we encountered in Mashhad (Iran), where 5 technicians daily inspect a WDN comprising 3,124 households/shops, 293 reservoirs/tanks, 356 wells and 14 treatment plants. To solve the problem, we propose three *Mixed Integer Linear Programming* (MILP) models, and an *Iterated Local Search* (ILS) algorithm. While the models managed to solve small-size instances with up to 20 nodes, the ILS efficiently tackled cases with up to 200 nodes, allowing us to produce good-quality solutions for randomly created instances, as well as for realistic instances derived from the case study, in short computing times.

The remainder of the paper is organized as follows. In Section 2, the relevant literature is revised. The problem is formally described in Section 3. Sections 4 and 5 present the mathematical models and the ILS algorithm, respectively. Computational results are described in Section 6, and final conclusions and future research directions are discussed in Section 7.

2 Literature Review

The VRP is an iconic class of problems in operations research, with applications in the fields of transportation, distribution, logistics and services. We refer the interested reader to Toth and Vigo (2014) for an extensive overview, and to Mor and Speranza (2022) for a recent survey. The problem we face generalizes the VRP by considering precedence constraints and multiple visits. In this section, we only revise routing problems involving these two features, with a particular focus on real-world applications.

In the context of the *Traveling Salesman Problem* (TSP), precedence constraints were first addressed in the seminal work by Balas et al. (1995), and, since then, have been widely investigated. In Moon et al. (2002), the authors proposed a formulation for the TSP with precedence constraints using a two-commodity network flow model and developed a genetic algorithm based on a topological sorting of customers. In Sarin et al. (2005), novel formulations for the asymmetric TSP and the precedence constrained asymmetric TSP were proposed. To tighten the formulations, the authors proposed and tested valid inequalities. Sun et al. (2018) presented a new model for the time-dependent capacitated profitable tour problem, a generalization of the TSP with time windows and precedence constraints, and developed a tailored labeling algorithm. Salman et al. (2020) describe the precedence constrained generalized TSP, in which customers are partitioned into groups and exactly one visit per group must be performed. They presented a novel branching technique and compared several bounding methods.

Precedence constraints have also been widely studied for problems involving multiple vehicles. Razali (2015) developed a genetic algorithm based on a topological sorting of customers to solve the VRP with precedence constraints. The algorithm includes a route repair method to generate feasible offspring. A VRP variant with time windows, synchronization and precedence constraints was introduced by Haddadene et al. (2016). The authors focused on an attended home health care application, and proposed some exact and heuristic solution methods, including a novel MILP formulation, a greedy heuristic, and three metaheuristics.

Precedence constraints naturally arise in the context of *Pickup-and-Delivery Problems* (PDP), where each demand must be first collected at an origin node before being delivered at a destination node. We refer the reader to Battarra et al. (2014) and Doerner and Salazar-González (2014) for detailed surveys on PDPs for goods transportation and PDPs for people transportation, respectively, and to Koç et al. (2020) for a recent survey on simultaneous PDPs. Recently, Aziez et al. (2020) studied a multi-PDP with time windows. They defined a 2-index formulation, an asymmetric representatives formulation, and a 3-index formulation improved by preprocessing and valid inequalities. The problem was solved exactly using a branch-and-cut algorithm. Dedicated branch-and-cut algorithms were also developed by Hernández-Pérez et al. (2021), to solve the single-vehicle two-echelon one-commodity PDP, and by Wolfinger and Salazar-González (2021), to solve a PDP with split loads and transshipments. The problem addressed in the latter work includes multiple visits to the same node. This is common when split deliveries are allowed, or multiple pickup and delivery operations can be performed at a single node. These generalizations were considered by Bruck and Iori (2017), where non-elementary formulations were proposed for a single-vehicle PDP and then extended to the cases of split deliveries, intermediate drop-offs, and multiple vehicles.

Overall, we may find many routing problems that are inspired by real-world applications and involve precedence constraints and multiple visits. Sigurd et al. (2004) studied an application of a PDP with time windows and precedence constraints arising in the transportation of live animals. In this case, the precedence constraints are given by veterinary rules, imposing that the livestock holdings are visited in a predefined sequence to avoid the spread of potential diseases. The authors proposed a tight formulation of the problem based on a Dantzig-Wolfe decomposition. Quttineh et al. (2013) presented an application in the context of military operations, that was modeled as a generalized VRP with synchronization and precedence constraints. The peculiarity of the problem is due to the nature of the attack, which may require aircraft synchronization, multiple attacks to the same target, and precedence constraints among different targets. The problem was solved by a MILP model.

Furtado et al. (2017) addressed a particular PDP with time windows originating from the oil industry. The aim of the problem is to determine the routing and scheduling of vessels that collect crude oil from offshore platforms and transport it to terminals on the coast. The authors proposed a MILP model, solving it by

means of two different branch-and-cut algorithms. Another valuable example of routing and scheduling in the context of large-scale disaster relief operations was examined by Sabouhi et al. (2019). The authors solved a PDP arising from a case study in the city of Tehran (Iran). They proposed an integrated logistic system to evacuate people from areas affected by natural or man-made disasters. The problem was formulated as a MILP model, and a memetic algorithm was developed to solve large-scale instances. Pereira et al. (2020) studied a particular *Workforce Scheduling and Routing Problem* (WSRP), called multiperiod WSRP with dependent tasks, in which the requested services consist of tasks to be executed along one or more days by teams of workers having different skills. Each customer can be visited more than once in a day, as long as precedence constraints are not violated. A MILP model, a constructive algorithm and an ant colony metaheuristic were proposed.

Recently, an interesting variant of the *Team Orienteering Problem* (TOP), named multi-visit TOP with precedence constraints, was investigated by Hanafi et al. (2020). In this problem, a set of tasks has to be accomplished in a predetermined order by possibly different vehicles. To solve the problem, the authors proposed a compact MILP formulation and a kernel search heuristic.

For what concerns WDNs, the literature mainly contains works on the location of sensors (see, e.g., Rathi and Gupta 2014). The VRP has been applied in many areas, but, to the best of our knowledge, not yet to the inspection of WDNs. In this paper, we fill this lack in the literature and propose exact and heuristic solution methods for a real-world VRP on a WDN.

3 Problem Description

The WDN is an essential infrastructure that consists of many elements, including reservoirs, wells, pipes and treatment plants.

An effective way to constantly monitor a WDN is by means of water quality sensors, which can be positioned all over the network. In cities where these sensor systems have not been installed, technicians are required to regularly visit nodes of the WDN and perform tests. The nodes to be visited, called for simplicity *demand nodes* in the following, are divided into two types:

1. *Type I*: households, shops, reservoirs, tanks and treatment plants. For this kind of nodes, the technicians can directly go on site and perform the required tests. Reservoirs, tanks and treatment plants are characterized by larger service times than households and shops, due to the larger amount of tests that have to be performed;

2. *Type II*: wells. For these nodes, the technicians need a key to access the well and perform the tests. So, they have to visit first a specified key center, and

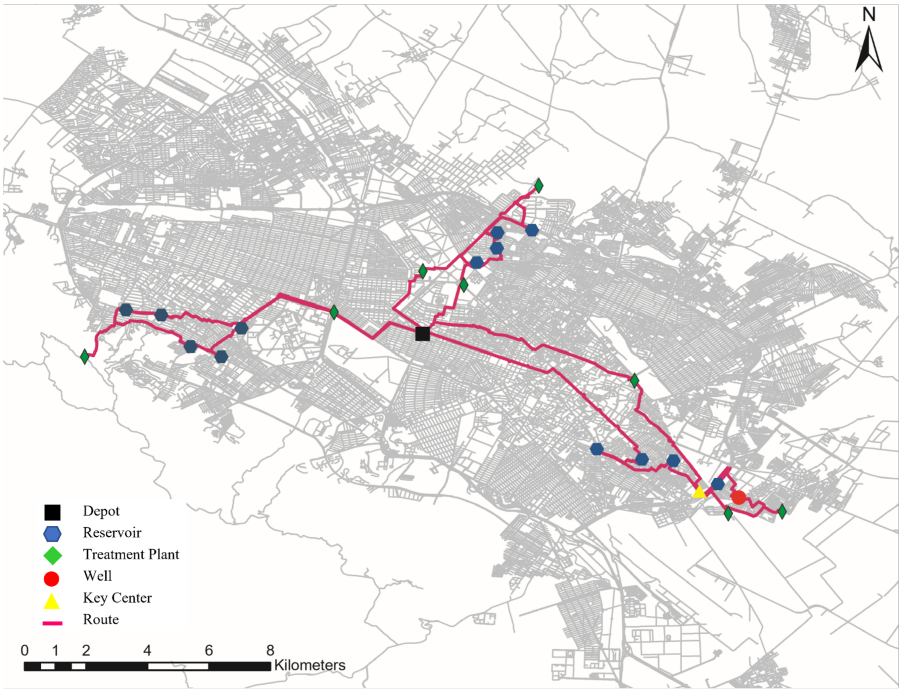

Figure 1: An illustrative example of a VRPWDN solution in Mashhad (Iran)

take the key. Once all tests have been completed at the well, the key needs to be returned to its original key center, thus imposing a second visit.

A simple illustrative example derived from the real-world application we are facing is depicted in Figure 1. It comprises three routes starting and ending at the depot. Two of them (top and left part of the figure) visit just reservoirs and treatment plants, so demand nodes of type I. The third (right part of the figure) also visits a well, and is thus forced to pass twice by the corresponding key center.

Formally, we are given a directed graph $G = (V, A)$, where the node set is $V = \{0, 1, \ldots, n, n+1\}$ and is partitioned as $V = V_1 \cup V_2 \cup V_3 \cup \{0, n+1\}$. Nodes 0 and $n+1$ represent, respectively, the beginning and end of all routes, and in our application coincide with a unique central depot. Sets V_1 and V_2 are associated with, respectively, the demand nodes of types I and II. Set V_3 comprises nodes associated with all key centers. With each node $i \in V_2$, we associate a predecessor $p_i \in V_3$ and a successor node $d_i \in V_3$. In our application, p_i and d_i correspond to a unique key center, so they have the same geographical location, but the models and algorithms that we propose below can also solve the case in which they correspond to different locations.

Each demand node has to be visited exactly once, while each vehicle visits a particular key center at most once for picking up all the keys, and then another single time for delivering all the keys that were previously collected. This implies

that, in case a center has the keys for multiple demand nodes and these nodes are visited by a unique vehicle, then such keys must be collected all together in a unique visit (to p_i), and then later delivered all together in another visit (to d_i). Note that it is not compulsory to visit p_i immediately before i. In other words, the vehicle can collect the key for i but then visit other nodes before reaching i. The same holds for d_i, which is not required to be visited immediately after i.

The graph is complete, and with each arc $(i, j) \in A$ we associate a traveling time c_{ij}. A service time v_i is associated with each node $i \in V$. We suppose that triangle inequality holds for all our instances (i.e., $c_{ij} \leq c_{ik} + v_k + c_{kj}$ for all $i, j, k \in V$). We are also given a set K of homogeneous vehicles based at the central depot. Each vehicle performs a single route.

A route starts and ends at the depot. Its duration is given by the sum of the service and traveling times of the nodes and arcs covered by the vehicle, and it should not exceed a maximum duration L. Whenever a route visits a node i of type II, then it should also visit p_i and d_i. The aim of the *Vehicle Routing Problem for Water Distribution Networks* (VRPWDN) is to visit all demand nodes, while satisfying all constraints and minimizing the sum of the route durations.

The VRPWDN is NP-hard in the strong sense, because it generalizes the well-known VRP. In the next sections, we attempt its solution through mathematical models and heuristic algorithms.

4 Mathematical Models

In this section, we investigate three mathematical models that describe the VRPWDN and are derived from the literature. The first model is based on a time representation of the problem and is inspired by the formulation proposed by Desaulniers et al. (2014) for the VRP with time windows. The second is a flow-based model that builds upon the formulation presented by Kara (2011) and later used by, among others, Karaoglan et al. (2012), Naji-Azimi and Salari (2014), and Allahyari et al. (2015). The third is a node-based model that we derive from the classical Miller, Tucker and Zemlin formulation (see, e.g., Bektaş and Gouveia 2014).

4.1 Time-based Model

Let y_{ik} be a binary variable taking value 1 if node i is visited by vehicle k and 0 otherwise, x_{ijk} be another binary variable taking value 1 if arc (i, j) is covered by vehicle k and 0 otherwise, and t_{ik} be a continuous variable corresponding to the time at which vehicle k arrives at node i. The time-based model for the VRPWDN can be formulated as follows:

$$(\text{VRPWDN}_{\text{tb}}) \quad \min \sum_{k \in K} \sum_{i \in V \setminus \{n+1\}} \sum_{j \in V \setminus \{0\}} (c_{ij} + v_i) x_{ijk} \tag{1}$$

subject to

$$\sum_{k\in K}\sum_{j\in V\setminus\{0\}} x_{ijk} = 1 \qquad i \in V_1 \cup V_2 \qquad (2)$$

$$\sum_{j\in V\setminus\{0\}} x_{0jk} = 1 \qquad k \in K \qquad (3)$$

$$y_{ik} = \sum_{j\in V\setminus\{0\}} x_{ijk} = \sum_{j\in V\setminus\{n+1\}} x_{jik} \qquad i \in V, k \in K \qquad (4)$$

$$\sum_{i\in V\setminus\{n+1\}} x_{i,n+1,k} = 1 \qquad k \in K \qquad (5)$$

$$\sum_{k\in K} t_{0k} = 0 \qquad (6)$$

$$0 \le t_{ik} \le L y_{ik} \qquad i \in V, k \in K \qquad (7)$$

$$t_{jk} \ge t_{ik} + v_i + c_{ij} - M_{ij}(1 - x_{ijk}) \qquad i \in V \setminus \{n+1\}, j \in V \setminus \{0\}, k \in K \qquad (8)$$

$$y_{p_i k} + y_{d_i k} \ge 2 y_{ik} \qquad i \in V_2, k \in K \qquad (9)$$

$$t_{p_i k} + v_{p_i} + c_{p_i i} - M'_i(1 - y_{ik}) \le t_{ik} \le t_{d_i k} - (c_{i d_i} + v_i) y_{ik} \qquad i \in V_2, k \in K \qquad (10)$$

$$x_{ijk} \in \{0, 1\} \qquad i, j \in V, k \in K \qquad (11)$$

$$y_{ik} \in \{0, 1\} \qquad i \in V, k \in K \qquad (12)$$

Objective function (1) is to minimize the total duration of the routes. Constraints (2) impose that each node $i \in V_1 \cup V_2$ has exactly one outgoing arc. Each vehicle starts its route from the depot and such condition is imposed by means of constraints (3). Constraints (4) and (5) ensure that each node i has exactly one incoming and one outgoing arc and that each vehicle k end its route at the depot. Constraints (6) impose that all routes start at time 0. Constraints (7) impose that arrival times are non-negative and limit the duration of each route to be at most L. The time at which vehicle k arrives at node j is modeled by means of constraints (8), in which we set $M_{ij} = L + v_i + c_{ij} - c_{j,n+1}$. Constraints (9) impose that if vehicle k visits node i, then it also visits nodes p_i and d_i. Since p_i may contain keys not only for i but for other nodes, vehicle k may visit p_i but not i, and the same holds for d_i. For this reason, the equation cannot be an equality. Constraints (10), in which we set $M'_i = L + v_{p_i} + c_{p_i,i} - c_{i,n+1}$, guarantee the respect of precedence constraints, by forcing time dependency between visits to p_i, i and d_i. Note that if y_{ik} is equal to 0, constraints (10) become redundant with respect to constraints (7). Constraints (11) and (12) define the domain of the x_{ijk} and y_{ik} variables.

Furthermore, the aforementioned model can be enhanced with the addition of the following valid inequalities

$$(c_{0p_i} + v_{p_i} + c_{p_i i}) y_{ik} \le t_{ik} \qquad i \in V_2, k \in K \qquad (13)$$

$$(c_{0p_i} + v_{p_i} + c_{p_i i} + v_i + c_{i d_i}) y_{ik} \le t_{d_i k} \qquad i \in V_2, k \in K \qquad (14)$$

$$t_{jk} \ge (c_{0i} + v_i + c_{ij}) x_{ijk} \qquad i \in V \setminus \{n+1\}, j \in V \setminus \{0\}, k \in K \qquad (15)$$

which strengthen the values taken by the arrival time variables.

4.2 Flow-based Model

Let f_{ijk} be a variable representing the "load" of vehicle k when traveling along arc $(i,j) \in A$. The load represents the number of nodes visited by vehicle k before it travels along arc (i,j). We can model the VRPWDN as follows:

$$(\text{VRPWDN}_{\text{fb}}) \quad \min \sum_{k \in K} \sum_{i \in V \setminus \{n+1\}} \sum_{j \in V \setminus \{0\}} (c_{ij} + v_i) x_{ijk} \tag{16}$$

subject to (2), (3), (11) and

$$\sum_{j \in V \setminus \{0\}} x_{ijk} = \sum_{j \in V \setminus \{n+1\}} x_{jik} \qquad i \in V, k \in K \tag{17}$$

$$\sum_{i \in V \setminus \{n+1\}} \sum_{j \in V \setminus \{0\}} (c_{ij} + v_i) x_{ijk} \leq L \qquad k \in K \tag{18}$$

$$\sum_{j \in V \setminus \{0\}} f_{0jk} = 0 \qquad k \in K \tag{19}$$

$$\sum_{i \in V \setminus \{n+1\}} f_{i,n+1,k} = \sum_{i \in V \setminus \{n+1\}} \sum_{j \in V \setminus \{0\}} x_{ijk} - 1 \qquad k \in K \tag{20}$$

$$\sum_{j \in V \setminus \{0\}} f_{ijk} \geq \sum_{j \in V \setminus \{n+1\}} (f_{jik} + x_{jik}) \qquad i \in V \setminus \{0, n+1\}, k \in K \tag{21}$$

$$\sum_{j \in V \setminus \{0\}} (f_{p_i jk} - f_{ijk} + x_{ijk}) \leq (n-1)(1 - \sum_{j \in V \setminus \{0\}} x_{ijk}) \qquad i \in V_2, k \in K \tag{22}$$

$$\sum_{j \in V \setminus \{0\}} f_{p_i jk} \geq \sum_{j \in V \setminus \{0\}} x_{ijk} \qquad i \in V_2, k \in K \tag{23}$$

$$\sum_{j \in V \setminus \{0\}} (f_{d_i jk} - f_{ijk}) \geq \sum_{j \in V \setminus \{0\}} x_{ijk} \qquad i \in V_2, k \in K \tag{24}$$

$$0 \leq f_{ijk} \leq (n-1) x_{ijk} \qquad i,j \in V, k \in K \tag{25}$$

As in the previous model, objective function (16) minimizes the total route duration. Constraints (17) correspond to the previous constraints (4) except for the y_{ik} term. The maximum duration of each route is bounded by means of constraints (18). Constraints (19) and (20) impose the load on vehicle k when leaving 0 and entering $n+1$, respectively. Constraints (21) impose the load conservation at node i. Constraints (22)–(24) guarantee the respect of precedence constraints. Constraints (25) impose lower and upper bounds on the f_{ijk} variables.

The above model can be improved by the addition of the following constraints:

$$\sum_{j \in V \setminus \{n+1\}} x_{jp_i k} + \sum_{j \in V \setminus \{0\}} x_{d_i jk} \geq 2 \sum_{j \in V \setminus \{n+1\}} x_{jik} \qquad i \in V_2, k \in K \tag{26}$$

$$\sum_{j \in V \setminus \{0\}} (f_{d_i jk} - f_{p_i jk}) \geq \sum_{l \in V \setminus \{n+1\} : p_i = p_l} \sum_{j \in V \setminus \{0\}} x_{ljk} \qquad i \in V_2, k \in K \tag{27}$$

$$\sum_{l \in V \setminus \{n+1\}} (f_{lik} - f_{ljk}) + n x_{ijk} + (n-2) x_{jik} \leq (n-1) \quad i,j \in V \setminus \{0, n+1\}, k \in K \tag{28}$$

Constraints (26) are equivalent to (9). Constraints (27) enforce an additional relation between the flows leaving p_i and d_i. Constraints (28) are derived from the lifted constraints proposed by Desrochers and Laporte (1991).

4.3 Node-based Model

Let u_{ik} be a variable representing the load on vehicle k after leaving node i. With respect to the previous model, this implies setting $u_{ik} = \sum_{j \in V} f_{ijk}$. We can model the VRPWDN as follows:

$$(\text{VRPWDN}_{\text{nb}}) \quad \min \sum_{k \in K} \sum_{i \in V \setminus \{n+1\}} \sum_{j \in V \setminus \{0\}} (c_{ij} + v_i) x_{ijk} \tag{29}$$

subject to (2), (3), (11), (17), (18) and

$$u_{0k} = 0 \qquad k \in K \tag{30}$$

$$u_{n+1,k} = \sum_{i \in V \setminus \{n+1\}} \sum_{j \in V \setminus \{0\}} x_{ijk} \qquad k \in K \tag{31}$$

$$u_{ik} - u_{jk} + n x_{ijk} \leq (n-1) \qquad i \in V \setminus \{n+1\}, j \in V \setminus \{0\}, k \in K \tag{32}$$

$$u_{p_i k} - u_{ik} + \sum_{j \in V \setminus \{0\}} x_{ijk} \leq n(1 - \sum_{j \in V \setminus \{0\}} x_{ijk}) \qquad i \in V_2, k \in K \tag{33}$$

$$u_{p_i k} \geq \sum_{j \in \{0\}} x_{ijk} \qquad i \in V_2, k \in K \tag{34}$$

$$u_{d_i k} - u_{ik} \geq \sum_{j \in V \setminus \{0\}} x_{ijk} \qquad i \in V_2, k \in K \tag{35}$$

$$0 \leq u_{ik} \leq n \sum_{j \in V \setminus \{0\}} x_{ijk} \qquad i \in V, k \in K \tag{36}$$

For each vehicle k, constraints (30) set the load after leaving node 0, while constraints (31) define the load when arriving at node $n+1$. Constraints (32) impose the load conservation when traveling from node i to node j. Constraints (33)–(35) guarantee the respect of precedence constraints. Constraints (36) impose both the non-negativity of the u_{ik} variables and their relation with the x_{ijk}variables.

The model can be improved by the addition of (26) and of

$$u_{d_i k} - u_{p_i k} \geq \sum_{l \in V \setminus \{n+1\} : p_i = p_l} \sum_{j \in V \setminus \{0\}} x_{ljk} \qquad i \in V_2, k \in K \tag{37}$$

$$u_{ik} - u_{jk} + n x_{ijk} + (n-2) x_{jik} \leq (n-1) \qquad i, j \in V \setminus \{0, n+1\}, k \in K \tag{38}$$

which correspond to the above (27) and (28), respectively.

5 Iterated Local Search

We developed an ILS algorithm with the purpose of finding good-quality VRPWDN solutions in short computing times. The choice of this metaheuristic is motivated

by its simplicity and effectiveness, in addition to the wide applicability it has found on related VRPs (see, e.g., Vansteenwegen et al. 2009, Silva et al. 2015, Haddadene et al. 2016 and Atefi et al. 2018). On the other hand, the need for short computing times is justified by the number of visits usually scheduled in a day in our real-world application, and by the fact that candidate locations might change at the beginning or in the course of a day. Two examples which typically cause a re-scheduling of visits can be a new warning for potential water contamination coming from a household or shop or, when visiting a well, the unfortunate event that the well's door is broken and it is not possible to open it.

Following the general framework proposed by Lourenço et al. (2019), the ILS starts from an initial solution and then improves it by iteratively invoking local search and perturbation procedures. The pseudo-code of the proposed ILS is provided in Algorithm 1. First, we generate an initial solution x_0 by means of a heuristic algorithm (line 1), and then we improve it with a local search procedure (line 2). The current solution, x, is stored as the incumbent, x^*, and inserted in the set of best known solutions obtained during the search, called $BKSet$ (lines 3 and 4). Next, we execute two phases, one after the other.

In the first phase, by applying a perturbation on x followed by a call to the local search (lines 6–8), the algorithm tries to escape from local optima. The perturbation is randomly selected between two tailored procedures. Let $z(x)$ and $l(x)$ be the cost of x and the maximum duration of a route in x, respectively. In case x has better cost than x^*, or same cost but lower maximum duration, then we use it to update x^*. In such a case, we also insert x in $BKSet$. This set contains the β different solutions found during the search and having the smallest $z(x)$ costs, breaking ties by smallest $l(x)$ value. If, instead, x does not improve x^*, then we set $x \leftarrow x^*$ as starting solution to be shaken at the next iteration. This loop is repeated until no improvement is found for max_{iter} iterations.

With the aim of further improving the solution obtained, at line 16 we enter the second ILS phase, in which a new series of improving attempts is performed. The idea is to intensify the search around the solutions contained in $BKSet$. For each such solution, we perform once more a loop of shaking and local search procedures, which is repeated until the same termination condition used above is met. Should one of these attempts manage to improve the incumbent solution, this time only in terms of costs, then the search restarts from the beginning of the first phase.

In the following, we provide the details of the main elements of the algorithm.

5.1 Initialization Procedure

Algorithm 2 gives the `Initialization` procedure that is used to generate an initial solution. At the beginning, $|K|$ routes are built in parallel by randomly selecting a first node $i \in V_1 \cup V_2$ per route. In case i belongs to V_2, then the predecessor

Algorithm 1 Iterated Local Search (ILS)

```
1: x_0 ← Initialization()                                   ▷ Generate an initial solution
2: x ← LocalSearch(x_0)
3: x* ← x
4: BKSet ← {x*}                                  ▷ BKSet: set of best known solutions
5: repeat                                                                  ▷ Phase 1
6:     Shake() ← Rand{S_1, S_2}                ▷ Randomly select a shaking procedures
7:     x ← Shake(x)
8:     x ← LocalSearch(x)
9:     Insert(x, BKSet)
10:    if z(x) < z(x*) OR (z(x) = z(x*) AND l(x) < l(x*)) then
11:        x* ← x
12:    else
13:        x ← x*
14:    end if
15: until no improvement is found for max_iter iterations
16: for j ← 1 to |BKSet| do                                                ▷ Phase 2
17:    x ← BKSet_j                                  ▷ Select the j^th solution ∈ BKSet
18:    repeat
19:        Shake() ← Rand{S_1, S_2}
20:        x ← Shake(x)
21:        x ← LocalSearch(x)
22:        if z(x) < z(x*) then
23:            x* ← x
24:            Insert(x*, BKSet)
25:            Go to line 5
26:        end if
27:    until no improvement is found for max_iter iterations
28: end for
29: return x*
```

and the successor of i (i.e., p_i and d_i) are also inserted into the route. In the next $|V_1 \cup V_2| - |K|$ iterations, a new node is randomly selected and inserted into an existing route. In these iterations, both the node and, in case $i \in V_2$, its predecessor and successor are inserted in the route in the positions that lead to the minimum extra mileage cost. Note that the insertion of node i or tuple (p_i, i, d_i) into an existing route is led by procedure `CheapestInsertion`, which evaluates among the $|K|$ routes the best candidate for the expansion. At line 22, the algorithm checks whether the solution is feasible. If not, then the whole procedure is repeated from scratch.

Algorithm 2 Initialization Procedure

$\mathcal{S}, \mathcal{V} \leftarrow \emptyset$
for $k \leftarrow 1$ **to** $|K|$ **do** ▷ Initialization of $|K|$ routes in parallel
  $i \leftarrow$ `Rand`$\{1, ..., |V_1 \cup V_2|\}$
  $\mathcal{V} \leftarrow \mathcal{V} \cup \{i\}$ ▷ Add i to the set of visited nodes
  if $i \in V_2$ **then**
    $r_k \leftarrow (0, p_i, i, d_i, n+1)$
    `Insert`$(r_k, \mathcal{S})$
  else
    $r_k \leftarrow (0, i, n+1)$
    `Insert`$(r_k, \mathcal{S})$
  end if
end for
for $j \leftarrow 1$ **to** $|V_1 \cup V_2| - |K|$ **do** ▷ Expansion of existing routes
  $i \leftarrow$ `Rand`$\{\{1, ..., |V_1 \cup V_2|\} \setminus \mathcal{V}\}$
  $\mathcal{V} \leftarrow \mathcal{V} \cup \{i\}$
  if $i \in V_2$ **then**
    `CheapestInsertion`$((p_i, i, d_i), r_k \in \mathcal{S})$
  else
    `CheapestInsertion`$(i, r_k \in \mathcal{S})$
  end if
end for
if `Feasible`$(\mathcal{S}) = 1$ **then**
  continue
else
  `Go to line 1`
end if
return $\mathcal{S}$

5.2 Local Search

The `LocalSearch` procedure invokes, one after the other, the following neighborhood searches:

LS1 *Swap intra-route*: swap two sequences with up to three consecutive nodes in the same route. Potential nodes belonging to V_3 are extracted from the two sequences and reinserted after the swap following the minimum extra mileage cost and respecting the precedence constraints;

LS2 *Swap inter-route*: swap two sequences with up to three consecutive nodes from different routes, taking care of nodes belonging to V_3;

LS3 *Relocate intra-route*: remove a sequence with up to three consecutive nodes and reinsert it in a different position within the same route, taking care of

nodes belonging to V_3;

LS4 *Relocate inter-route*: remove a sequence with up to three consecutive nodes and reinsert it in a different route, taking care of nodes belonging to V_3;

LS5 *3-opt*: in a preliminary step, select a route and remove potential nodes belonging to V_3. Following this step, apply the standard 3-opt algorithm to the remaining nodes. After each iteration of the 3-opt algorithm update the solution by reinserting the previously extracted nodes belonging to V_3.

Procedures from LS1 to LS4 have all complexity $O(n^2)$, whereas LS5 has complexity $O(n^3)$. To limit the computational effort required by LS5, a random logic search is added. In particular, a candidate route k is selected randomly and potential nodes belonging to V_3 are removed as follows. For each node i in the route, the saving s_i that could be obtained by removing i and directly connecting the predecessor and successor nodes of i in the route is computed. Then, the probability of removing i is set to $p_i = s_i / \sum_j s_j$. By means of the roulette wheel mechanism, three non-adjacent nodes are selected for removal, and then the resulting route is optimized by a 3-opt algorithm. A threshold of γ iterations is set to limit the number of attempts.

The calls to LS1–LS5 are repeated as long as an improvement is found. Procedure `LocalSearch` hence returns a solution which is a local optimum with respect to all five neighborhoods.

5.3 Shaking Procedure

To perturb a solution, we randomly select, with same probability, one of the two following procedures.

S1 *Shaking 1:* randomly select a route k and execute a random iteration of the 3-opt algorithm to update the order of visits. If the cost of the current solution is not worse than $\alpha z(x^*)$, with α being an input parameter, randomly select a second route k' and perform another 3-opt iteration. The procedure is iterated as long as the cost of the perturbed solution is not worse than $\alpha z(x^*)$;

S2 *Shaking 2:* compute the cost saving obtained by removing any node from the solution, similarly to what is done in LS5. Then use the roulette wheel mechanism to select a node $i \in V \setminus \{0, n+1\}$, and remove i from its route. The removal procedure is iterated until at least α percent of all nodes have been removed. If the selected node belongs to V_2, then its saving is computed as the average cost saving obtained by removing i, p_i, and d_i. At the end of this step, the algorithm invokes the `Initialization` procedure to rebuild a feasible solution.

6 Computational Results

In this section, we present the results of extensive computational tests performed with the aim of assessing the performance of the proposed methods. The mathematical models and the ILS were coded in C++ using Microsoft Visual Studio 2010. The computational tests were executed on a PC equipped with an Intel Core i7 CPU processor @ 2.70 GHz and 6 GB of RAM, using CPLEX 12.3 as MILP solver. In Section 6.1, we describe the sets of randomly-created instances that we used for our tests. The comparison among the mathematical models is reported in Section 6.2, while the behavior of the ILS is analyzed in Sections 6.3 and 6.4. In Section 6.5, we report the results of additional computational experiments performed on a set of realistic instances derived from the case study.

6.1 Randomly-created Instances

We created several random instances with the aim of assessing the performance of the algorithms under different situations. In detail, we created two sets of instances, each comprising different subsets having homogeneous values of $|V_1 \cup V_2|$, $(|V_2|, |V_3|)$ and $|K|$, and composed by three random instances per subset. We obtained the following sets:

- *Small-size*: 18 instances with $|V_1 \cup V_2|$=10, $(|V_2|, |V_3|) \in \{(1,1), (2,1), (2,2)\}$, and $|K| \in \{1,2\}$; 24 instances with $|V_1 \cup V_2|$=15, $(|V_2|, |V_3|) \in \{(3,2), (3,3), (4,2), (4,3)\}$, and $|K| \in \{2,3\}$; 24 instances with $|V_1 \cup V_2|$=20, $(|V_2|, |V_3|) \in \{(2,2), (3,2), (3,3), (5,3)\}$, and $|K| \in \{2,3\}$;
- *Medium- and large-size*: 24 instances with $|V_1 \cup V_2|$=50, $(|V_2|, |V_3|) \in \{(5,5), (8,8), (10,5), (10,8)\}$, and $|K| \in \{5,8\}$; 24 instances with $|V_1 \cup V_2|$=100, $(|V_2|, |V_3|) \in \{(5,5), (10,5), (10,10), (15,10)\}$, and $|K| \in \{10,15\}$; 24 instances with $|V_1 \cup V_2|$=200, $(|V_2|, |V_3|) \in \{(10,10), (20,10), (20,20), (30,20)\}$, and $|K| \in \{15,20\}$.

For each instance, the coordinates of the nodes are integer values randomly selected between 0 and 100. The distances between the nodes are computed as the Euclidean ones, rounded to the second closest digit. The maximum duration is set to $L = 1.5(\sum_{i \in V_1 \cup V_2} \overline{c}_i + |K| \sum_{i \in V_3 \cup \{0\}} \overline{c}_i)/|K|$, where $\overline{c}_i$ is the average travel time of the arcs leaving i, computed as $\overline{c}_i = \sum_{j \in V \setminus \{i\}} c_{ij}/(|V|-1)$ for each node $i \in V \setminus \{n+1\}$. The service time v_i for each node $i \in V_1 \cup V_2 \cup V_3$ is set to a random integer value between 20 and 40.

In the following, a subset of instances is identified by the tuple $(|V_1 \cup V_2|, |V_2|, |V_3|, |K|)$, while a single instance is identified by $(|V_1 \cup V_2|, |V_2|, |V_3|, |K|, u)$, where u is a numerical index going from 1 to 3.

To favor future research on the problem, the randomly-created instances have been made publicly available at `https://github.com/DarioVezzali/VRPWDN`.

6.2 Comparison among the Mathematical Models

In this section, the performance of the three mathematical models from Section 4 is investigated. A time limit of 3,600 CPU seconds was imposed on each execution. The aggregated results that we obtained are reported in Table 1. Each line reports average/total values for a group of three instances having the same numbers of vertices and vehicles. For each group, columns "z_{lb}" and "z_{ub}" give the average lower and upper bound values, respectively, column "%gap" gives the average percentage gap and column "t(s)" the average run time. An entry "tlim" indicates that the time limit was reached for all the three instances in the group. Column "opt" gives the total number of instances solved to proven optimality.

From Table 1, we can observe that just on a few large-size instances the time-based model and the node-based model find better results in terms of average upper bound. Overall, the flow-based model outperforms the other two models in terms of average lower bound, average percentage gap, average run time, and number of optimal solutions obtained. Consequently, we adopted this model to assess the quality of the solutions obtained by the ILS (see Section 6.3).

For all the instances belonging to the medium- and large-size sets, the mathematical models could not obtain proven optimal solutions and the computer frequently ran out of memory because of the large model size. Overall, we can conclude that the results prove the need of a good heuristic for these instances. This need is further motivated by the dimension of the original real-world problem, where the number of visits per day (i.e., around 70) is out of scale if compared to the size of instances solved to optimality within the time limit.

To assess the performance of the proposed valid inequalities, six small-size instances were selected and solved running the three models with and without the addition of the valid inequalities. The results are reported in Table 2. We can notice that the inequalities help improve the performance of all models, by reducing the average percentage gap and execution time, and increasing the number of proven optimal solutions.

6.3 ILS Parameter Tuning

The ILS procedure adopts four main parameters (i.e., α, β, γ and max_{iter}). To set their values, we randomly selected six instances (two with $0 \leq n \leq 20$, two with $50 \leq n \leq 100$, and two with $n = 200$). We then tested the ILS on these instances by attempting all possible combinations of parameter values chosen in the sets $\alpha \in \{0.05.0.10, 0.15, 0.25\}$, $\beta \in \{2, 5, 10, 20\}$, $\gamma \in \{50, 100\}$ and $max_{iter} \in$

Table 1: Comparison of mathematical models (three inst. per line). Best average lower and upper bound values in **boldface**

				time-based					flow-based					node-based				
$\|V_1 \cup V_2\|$	$\|V_2\|$	$\|V_3\|$	$\|K\|$	z_{lb}	z_{ub}	%gap	t(s)	opt	z_{lb}	z_{ub}	%gap	t(s)	opt	z_{lb}	z_{ub}	%gap	t(s)	opt
10	1	1	1	**706.39**	**706.39**	0.00	0.87	3	**706.39**	**706.39**	0.00	1.13	3	**706.39**	**706.39**	0.00	1.42	3
10	1	1	2	**730.91**	**730.91**	0.00	7.45	3	**730.91**	**730.91**	0.00	3.26	3	**730.91**	**730.91**	0.00	12.20	3
10	2	1	1	**743.25**	**743.25**	0.00	6.54	3	**743.25**	**743.25**	0.00	3.13	3	**743.25**	**743.25**	0.00	0.66	3
10	2	1	2	**793.75**	**793.75**	0.00	23.12	3	**793.75**	**793.75**	0.00	9.45	3	**793.75**	**793.75**	0.00	10.96	3
10	2	2	1	**803.96**	**803.96**	0.00	177.59	3	**803.96**	**803.96**	0.00	16.73	3	**803.96**	**803.96**	0.00	44.80	3
10	2	2	2	**833.78**	**833.78**	0.00	110.68	3	**833.78**	**833.78**	0.00	75.01	3	**833.78**	**833.78**	0.00	436.67	3
sum/avg (10)				**768.67**	**768.67**	0.00	54.38	18	**768.67**	**768.67**	0.00	18.12	18	**768.67**	**768.67**	0.00	84.45	18
15	3	2	2	**1036.61**	**1036.61**	0.00	1811.64	3	**1036.61**	**1036.61**	0.00	509.71	3	**1036.61**	**1036.61**	0.00	823.41	3
15	3	2	3	1014.04	**1049.45**	2.88	2054.35	2	**1049.45**	**1049.45**	0.00	1325.68	3	1021.59	**1049.45**	2.41	2419.87	1
15	3	3	2	1065.96	**1114.68**	4.39	tlim	0	**1114.68**	**1114.68**	0.00	1104.10	3	1067.19	**1114.68**	4.28	2973.83	1
15	3	3	3	1101.81	**1160.32**	5.05	tlim	0	**1153.10**	1160.74	0.65	1358.60	2	1096.70	**1160.32**	5.49	tlim	0
15	4	2	2	**1036.61**	**1036.61**	0.00	1822.00	3	**1036.61**	**1036.61**	0.00	510.81	3	**1036.61**	**1036.61**	0.00	825.53	3
15	4	2	3	1081.47	**1084.12**	0.28	2439.70	2	**1084.12**	**1084.12**	0.00	1108.56	3	**1084.12**	**1084.12**	0.00	1637.68	3
15	4	3	2	1073.81	1150.81	6.81	tlim	0	**1142.24**	**1149.80**	0.69	1420.98	2	1082.75	1150.81	6.03	tlim	0
15	4	3	3	1111.81	1203.22	7.72	tlim	0	**1190.64**	**1202.40**	1.03	1430.51	2	1113.29	1204.85	7.73	tlim	0
sum/avg (15)				1065.27	1104.48	3.39	2815.96	10	**1100.93**	**1104.30**	0.30	1096.12	21	1067.36	1104.68	3.24	2435.64	11
20	2	2	2	1225.52	**1275.44**	3.87	3274.98	1	**1243.73**	1277.82	2.61	1354.86	2	1228.48	**1275.44**	3.62	2266.04	2
20	2	2	3	1262.37	1330.87	5.14	tlim	0	**1294.95**	**1329.12**	2.56	1458.53	2	1251.59	1332.91	6.09	tlim	0
20	3	2	2	1197.44	**1269.71**	5.60	tlim	0	**1243.52**	**1269.71**	2.02	1271.07	2	1236.21	**1269.71**	2.58	1531.30	2
20	3	2	3	1220.35	**1301.73**	6.13	tlim	0	**1301.73**	**1301.73**	0.00	1127.63	3	1253.86	**1301.73**	3.58	2510.73	2
20	3	3	2	1210.01	**1289.02**	5.94	tlim	0	**1260.69**	1296.61	2.60	2661.64	1	1215.65	1295.67	5.95	tlim	0
20	3	3	3	1223.85	**1321.84**	7.26	tlim	0	**1281.30**	**1321.84**	2.97	tlim	0	1204.51	1337.03	9.69	tlim	0
20	5	3	2	1200.10	1280.16	6.12	tlim	0	**1241.98**	1280.43	2.91	2986.79	1	1235.59	**1278.32**	3.24	2739.77	1
20	5	3	3	1228.58	1324.16	7.12	tlim	0	**1270.11**	**1322.70**	3.84	2863.57	1	1251.23	1402.59	9.43	2891.87	1
sum/avg (20)				1221.03	**1299.11**	5.90	3559.37	1	**1267.25**	1300.00	2.44	2165.51	12	1234.64	1311.67	5.52	2842.59	8
overall sum/avg				1041.02	**1083.67**	3.38	2333.13	29	**1070.80**	1083.93	0.99	1190.99	51	1046.73	1088.31	3.19	1942.39	37

Table 2: Effect of valid inequalities on six small-size instances

mathematical model	without valid inequalities					with valid inequalities				
	z_{lb}	z_{ub}	%gap	t(s)	opt	z_{lb}	z_{ub}	%gap	t(s)	opt
time-based	1036.25	1077.49	3.24	2484.51	2	1040.67	1076.85	2.81	2288.63	3
flow-based	1062.79	1076.85	1.00	953.69	5	1064.43	1076.85	0.88	751.28	5
node-based	1033.41	1084.99	3.86	1912.63	3	1037.93	1068.09	2.38	1745.36	4

Table 3: ILS parameter tuning. Best configuration in **boldface**

	(γ, max_{iter})															
	$(50,200)$		$(50,500)$		$(50,1000)$		$(50,5000)$		$(100,200)$		$(100,500)$		$(100,1000)$		$(100,5000)$	
(α,β)	t(s)	%gap	t(s)	%gap	t(s)	%gap	t(s)	%gap	t(s)	%gap	t(s)	%gap	t(s)	%gap	t(s)	%gap
(0.05,2)	1.53	0.91	1.79	0.87	2.13	0.84	2.79	0.82	1.56	0.83	1.80	0.79	1.89	0.78	3.28	0.77
(0.05,5)	1.67	0.76	1.83	0.75	2.27	0.73	2.91	0.72	1.79	0.75	1.90	0.74	2.02	0.74	3.49	0.73
(0.05,10)	1.91	0.76	2.18	0.74	2.66	0.73	3.41	0.72	2.08	0.73	2.19	0.73	2.26	0.73	3.91	0.71
(0.05,20)	1.73	0.76	1.93	0.74	2.34	0.73	2.86	0.72	2.20	0.73	2.35	0.72	2.43	0.69	4.67	0.69
(0.10,2)	2.09	0.08	2.33	0.03	2.68	0.02	3.58	0.01	1.91	0.13	1.97	0.11	2.09	0.10	2.58	0.10
(0.10,5)	2.74	0.08	3.25	0.02	**4.02**	**0.00**	5.44	0.00	2.06	0.11	2.38	0.07	2.89	0.06	3.28	0.05
(0.10,10)	3.13	0.08	4.06	0.02	5.04	0.00	6.47	0.00	2.49	0.07	2.61	0.07	3.05	0.06	3.39	0.05
(0.10,20)	3.57	0.08	4.53	0.02	5.23	0.00	8.02	0.00	2.84	0.07	3.11	0.06	3.24	0.06	3.46	0.05
(0.15,2)	1.83	0.43	2.06	0.39	2.30	0.38	3.12	0.35	2.04	0.38	2.37	0.35	2.49	0.35	3.12	0.35
(0.15,5)	2.49	0.40	2.86	0.38	3.13	0.35	5.08	0.35	2.33	0.36	2.59	0.35	3.20	0.34	4.85	0.33
(0.15,10)	3.35	0.40	3.88	0.38	4.16	0.35	6.37	0.35	2.48	0.35	3.79	0.33	4.11	0.33	5.09	0.33
(0.15,20)	4.55	0.40	5.02	0.38	5.23	0.35	8.64	0.35	2.71	0.35	4.26	0.33	4.82	0.33	5.94	0.32
(0.25,2)	2.25	1.34	2.64	1.07	3.30	0.94	4.56	0.92	2.21	0.88	2.27	0.86	2.84	0.86	4.19	0.86
(0.25,5)	2.54	1.18	2.93	0.91	4.05	0.89	5.62	0.89	2.68	0.86	3.16	0.85	3.74	0.85	4.80	0.83
(0.25,10)	3.00	1.16	3.94	0.90	4.94	0.86	7.33	0.86	3.52	0.83	4.86	0.82	6.07	0.82	7.83	0.82
(0.25,20)	3.72	1.16	5.12	0.90	6.31	0.86	8.89	0.86	4.67	0.83	6.13	0.82	6.63	0.82	8.46	0.82

$\{200, 500, 1000, 5000\}$. The results are reported in Table 3. For each combination of parameters, column "t(s)" gives the average ILS run time on the six instances, and column "%gap" gives the average gap computed as the average over the six instances of $100(z-z^*)/z^*$. Here, z is the value of the solution obtained by the given configuration and z^* is the value of the best solution obtained by all configurations.

The configuration with $\alpha = 0.10$, $\beta = 5$, $\gamma = 50$ and $max_{iter} = 1000$ is the one that obtained the best results (highlighted in bold in the table). It could always achieve the best solution values, at the expense of a limited increase in the computing time with respect to configurations adopting a smaller number of iterations. This configuration was thus adopted for all successive ILS tests.

6.4 ILS Evaluation

In this section, we investigate the performance of the ILS. In Table 4, the results of the ILS are compared with those obtained by the best mathematical model (i.e., the flow-based one) on groups of three instances per line. We recall that column "z_{ub}" gives the average upper bound value, column "opt" the number of proven optimal solutions, and column "t(s)" the average run time. The ILS was executed

Table 4: Computational results on small-size instances (three inst. per line)

				flow-based			ILS				
$\|V_1 \cup V_2\|$	$\|V_2\|$	$\|V_3\|$	$\|K\|$	z_{ub}	t(s)	opt	z_{best}	z_{avg}	z_{worst}	σ_z	t(s)
10	1	1	1	706.39	1.13	3	706.39	706.39	706.39	0.00	0.00
10	1	1	2	730.91	3.26	3	730.91	730.91	730.91	0.00	0.00
10	2	1	1	743.25	3.13	3	743.25	743.25	743.25	0.00	0.00
10	2	1	2	793.75	9.45	3	793.75	793.75	793.75	0.00	0.00
10	2	2	1	803.96	16.73	3	803.96	803.96	803.96	0.00	0.00
10	2	2	2	833.78	75.01	3	833.78	833.78	833.78	0.00	0.00
sum/avg (10)				768.67	18.12	18	768.67	768.67	768.67	0.00	0.00
15	3	2	2	1036.61	509.71	3	1036.61	1036.61	1036.61	0.00	0.14
15	3	2	3	1049.45	1325.68	3	1049.45	1049.45	1049.45	0.00	0.22
15	3	3	2	1114.68	1104.10	3	1114.68	1114.68	1114.68	0.00	0.14
15	3	3	3	1160.74	1358.60	2	1155.96	1155.96	1155.96	0.00	0.24
15	4	2	2	1036.61	510.81	3	1036.61	1036.61	1036.61	0.00	0.16
15	4	2	3	1084.12	1108.56	3	1084.12	1084.12	1084.12	0.00	0.22
15	4	3	2	1149.80	1420.98	2	1146.56	1146.56	1146.56	0.00	0.20
15	4	3	3	1202.40	1430.51	2	1202.40	1202.40	1202.40	0.00	0.27
sum/avg (15)				1104.30	1096.12	21	1103.30	1103.30	1103.30	0.00	0.20
20	2	2	2	1277.82	1354.86	2	1275.44	1275.44	1275.44	0.00	0.26
20	2	2	3	1329.12	1458.53	2	1316.03	1316.03	1316.03	0.00	0.33
20	3	2	2	1269.71	1271.07	2	1262.52	1262.52	1262.52	0.00	0.35
20	3	2	3	1301.73	1127.63	3	1301.73	1301.73	1301.73	0.00	0.41
20	3	3	2	1296.61	2661.64	1	1277.25	1277.25	1277.25	0.00	0.35
20	3	3	3	1321.84	tlim	0	1301.37	1301.37	1301.37	0.00	0.61
20	5	3	2	1280.43	2986.79	1	1265.46	1265.46	1265.46	0.00	0.33
20	5	3	3	1322.70	2863.57	1	1303.93	1303.93	1303.93	0.00	0.73
sum/avg (20)				1300.00	2165.51	12	1287.97	1287.97	1287.97	0.00	0.42
overall sum/avg				1083.93	1190.99	51	1079.19	1079.19	1079.19	0.00	0.23

five times on each instance. We report the best, average and worst solution values achieved, as well as their standard deviation, in columns "z_{best}", "z_{avg}", "z_{worst}" and "σ_z", respectively. More in detail, z_{best} gives the average of the best solution values produced on the three instances, z_{avg} the average of the average values, and z_{worst} the average of the worst values. The average computational time is shown in column "t(s)".

According to the results, for those groups of three instances that were all solved to optimality by the flow-based model, the ILS obtained the same optimal values in a shorter computational time. For all the remaining small-size sets, the ILS achieved better values than the flow-based model (without proof of their optimality). In addition, the constantly null average standard deviation among the different runs indicates the robustness of the algorithm on these very simple instances. When comparing the average run times, we can notice that the ILS needed an overall average time of just 0.23 seconds against the 1,190.99 seconds of the flow-based model.

In Table 5, we report the results of the ILS on medium- and large-size instances. On instances having $|V_1 \cup V_2| = 50$ the average standard deviation is 0.00, on those

Table 5: Computational results on medium- and large-size instances (three inst. per line)

				ILS				
$\|V_1 \cup V_2\|$	$\|V_2\|$	$\|V_3\|$	$\|K\|$	z_{best}	z_{avg}	z_{worst}	σ_z	t(s)
50	5	5	5	2583.27	2583.27	2583.27	0.00	1.17
50	5	5	8	2721.90	2721.90	2721.90	0.00	1.59
50	8	8	5	2883.07	2883.07	2883.07	0.00	1.67
50	8	8	8	3001.70	3001.70	3001.70	0.00	2.10
50	10	5	5	2664.02	2664.02	2664.02	0.00	2.09
50	10	5	8	2807.41	2807.41	2807.41	0.00	2.19
50	10	8	5	2863.64	2863.64	2863.64	0.00	1.91
50	10	8	8	3003.07	3003.07	3003.07	0.00	2.52
avg (50)				2816.01	2816.01	2816.01	0.00	1.91
100	5	5	10	4430.65	4430.92	4431.53	0.40	7.61
100	5	5	15	4642.24	4642.45	4643.13	0.39	8.29
100	10	5	10	4507.07	4507.27	4508.07	0.45	7.19
100	10	5	15	4750.73	4750.92	4751.65	0.41	8.17
100	10	10	10	4856.94	4857.16	4857.99	0.47	9.03
100	10	10	15	5062.41	5062.62	5063.43	0.45	9.43
100	15	10	10	4826.28	4826.62	4827.96	0.75	9.18
100	15	10	15	5070.19	5070.50	5071.74	0.69	9.90
avg (100)				4768.31	4768.56	4769.44	0.50	8.60
200	10	10	15	8244.39	8244.97	8246.12	0.82	9.86
200	10	10	20	8636.53	8637.11	8638.33	0.84	10.34
200	20	10	15	8550.63	8551.32	8552.62	0.98	12.27
200	20	10	20	8814.41	8815.00	8816.05	0.82	13.29
200	20	20	15	9128.90	9129.63	9130.63	0.82	13.66
200	20	20	20	9305.35	9305.98	9307.13	0.81	14.96
200	30	20	15	9372.60	9373.86	9375.17	1.16	14.05
200	30	20	20	9497.20	9498.20	9499.67	1.10	15.70
avg (200)				8943.75	8944.51	8945.72	0.92	13.02
overall avg				5509.36	5509.69	5510.39	0.47	7.84

having $|V_1 \cup V_2| = 100$ it becomes 0.50, while on those having $|V_1 \cup V_2| = 200$ it increases to 0.92, thus resulting in an overall average standard deviation of 0.47. This confirms the robustness of the algorithm. Concerning the run time, the ILS took on average 1.91 seconds to solve instances having $|V_1 \cup V_2| = 50$, 8.60 seconds for those having $|V_1 \cup V_2| = 100$, and 13.02 seconds for those having $|V_1 \cup V_2| = 200$. The overall average run time is 7.84 seconds, proving that the method is suitable for a quick use in practical situations.

Finally, Table 6 reports a sensitivity analysis on the average percentage of computational time needed by each ILS component, grouped by set of instances. On the small-size sets, LS2 and LS3 are the most time-consuming local search procedures, while for medium- and large-size sets the largest effort is required by LS1 and LS4.

6.5 Results on Realistic Instances

The flow-based model and the ILS were also tested on a set of realistic instances

Table 6: Percentage of the computational time needed by each ILS component

Set	LS1	LS2	LS3	LS4	LS5	S1	S2
Small-size	3.63%	39.41%	28.65%	6.37%	20.50%	0.37%	1.07%
Medium-size	59.50%	9.02%	1.30%	18.36%	10.69%	0.23%	0.89%
Large-size	50.03%	3.48%	3.33%	32.35%	9.98%	0.13%	0.71%

generated from the WDN in the city of Mashhad (Iran). Our real case study consists of 3,124 households/shops, 293 reservoirs/tanks, 356 wells and 14 treatment plants. For all of these nodes the exact locations were collected.

Following the same rationale described in Section 6.1, we generated 108 realistic instances divided into two sets of *small-size* and *medium- and large-size* instances, each comprising different subsets having homogeneous values of $|V_1 \cup V_2|$, $(|V_2|, |V_3|)$, and $|K|$, and composed by three random instances per subset. The resulting sets are:

- *Small-size*: 12 instances with $|V_1 \cup V_2|$=10, $(|V_2|, |V_3|) \in \{(1,1), (2,1), (2,2)\}$, and $|K| \in \{1,2\}$; 12 instances with $|V_1 \cup V_2|$=15, $(|V_2|, |V_3|) \in \{(1,1), (2,1), (2,2)\}$, and $|K| \in \{1,2,3\}$; 12 instances with $|V_1 \cup V_2|$=20, $(|V_2|, |V_3|) \in \{(1,1), (2,1), (2,2)\}$, and $|K| \in \{2,3\}$;
- *Medium- and large-size*: 12 instances with $|V_1 \cup V_2|$=40, $(|V_2|, |V_3|) \in \{(4,2), (4,3), (6,2), (6,3)\}$, and $|K| \in \{2,3\}$; 12 instances with $|V_1 \cup V_2|$=50, $(|V_2|, |V_3|) \in \{(4,2), (4,3), (6,2), (6,3)\}$, and $|K| \in \{2,3\}$; 12 instances with $|V_1 \cup V_2|$=60, $(|V_2|, |V_3|) \in \{(4,2), (4,3), (6,2), (6,3)\}$, and $|K| \in \{2,3\}$; 12 instances with $|V_1 \cup V_2|$=100, $(|V_2|, |V_3|) \in \{(8,4), (8,5), (10,4), (10,5)\}$, and $|K| \in \{4,5\}$; 12 instances with $|V_1 \cup V_2|$=150, $(|V_2|, |V_3|) \in \{(8,4), (8,5), (10,4), (10,5)\}$, and $|K| \in \{4,5\}$; 12 instances with $|V_1 \cup V_2|$=200, $(|V_2|, |V_3|) \in \{(8,4), (8,5), (10,4), (10,5)\}$, and $|K| \in \{4,5\}$.

For each instance, the coordinates of the nodes were randomly selected among the given real locations. The flow-based model and the ILS were used to run the experiments. The results are reported in the Tables 7 and 8. In Table 7, the results of the ILS are compared with those obtained by the flow-based model. We recall that columns "z_{ub}", "t(s)" and "opt" give the average upper bound value, the average run time and the total number of instances solved to proven optimality by the mathematical model, respectively. Note that an entry "tlim" indicates that the time limit of 3,600 CPU seconds was reached for all the three instances in the group. Conversely, columns "z_{best}", "z_{avg}", "z_{worst}", "σ_z" and "t(s)" give the best, average and worst solution values, the standard deviation and the computational time of the ILS, respectively.

We can notice that on small-size instances, the flow-based model and the ILS obtained the same optimal values on instances having $|V_1 \cup V_2| \in \{10, 15\}$, and on one subset out of four of instances having $|V_1 \cup V_2| = 20$. For the remaining subsets,

Table 7: Computational results on realistic small-size instances (three inst. per line)

				flow-based			ILS				
$\|V_1 \cup V_2\|$	$\|V_2\|$	$\|V_3\|$	$\|K\|$	z_{ub}	t(s)	opt	z_{best}	z_{avg}	z_{worst}	σ_z	t(s)
10	1	1	1	82475.75	12.69	3	82475.75	82475.75	82475.75	0.00	0.00
10	1	1	2	80629.79	8.95	3	80629.79	80629.79	80629.79	0.00	0.00
10	2	1	1	121874.44	5.53	3	121874.44	121874.44	121874.44	0.00	0.00
10	2	2	2	116649.98	135.45	3	116649.98	116649.98	116649.98	0.00	0.00
sum/avg (10)				100407.49	40.66	12	100407.49	100407.49	100407.49	0.00	0.00
15	1	1	1	97874.84	1218.80	2	97874.84	97874.84	97874.84	0.00	0.05
15	1	1	2	186943.98	46.03	3	186943.98	186943.98	186943.98	0.00	0.08
15	2	1	2	126742.54	1204.46	2	126742.54	126742.54	126742.54	0.00	0.12
15	2	2	3	135254.03	2437.79	1	135254.03	135254.03	135254.03	0.00	0.12
sum/avg (15)				136703.85	1226.77	8	136703.85	136703.85	136703.85	0.00	0.09
20	1	1	2	158682.88	1205.87	2	158588.73	158588.73	158588.73	0.00	0.18
20	1	1	3	175830.70	3342.99	1	175655.43	175655.43	175655.43	0.00	0.20
20	2	1	2	170015.90	1466.86	3	170015.90	170015.90	170015.90	0.00	0.20
20	2	2	3	188935.98	tlim	0	187949.09	187949.09	187949.09	0.00	0.26
sum/avg (20)				173366.36	2403.93	6	173052.29	173052.29	173052.29	0.00	0.21
overall sum/avg				136825.90	1223.79	26	136721.21	136721.21	136721.21	0.00	0.10

the ILS achieved better values than the flow-based model (again, without proof of their optimality).

On medium- and large-size instances, the ILS achieved very robust results on instances having $|V_1 \cup V_2| \in \{40, 50, 60\}$. Indeed, the standard deviation is constantly null for all the subgroups, and the run times are very short. The robustness of the ILS slightly decreases for instances having $|V_1 \cup V_2| \in \{100, 150, 200\}$, however remaining acceptable for a practical use. For these instances, the average run times are around 7.32, 9.80 and 11.08 seconds, respectively, thus confirming that the algorithm could efficiently solve realistic instances having a considerable number of nodes in a few seconds.

7 Conclusions

In this paper, we introduced a generalization of the well-known Vehicle Routing Problem (VRP), called VRP for Water Distribution Networks (VRPWDN), that includes precedence constraints among nodes and multiple visits to some of the nodes. The problem is NP-hard in the strong sense and, to the best of our knowledge, has not yet been applied in the context of distribution networks where regular inspections have to be performed to detect potential sources of contamination. To solve the VRPWDN, three alternative mathematical models (time-based, flow-based and node-based) were proposed, and an Iterated Local Search (ILS) algorithm was developed.

Extensive computational tests on randomly generated small-size instances were

Table 8: Comp. results on realistic medium- and large-size instances (three inst. per line)

				ILS				
$\|V_1 \cup V_2\|$	$\|V_2\|$	$\|V_3\|$	$\|K\|$	z_{best}	z_{avg}	z_{worst}	σ_z	t(s)
40	4	2	2	180613.80	180613.80	180613.80	0.00	0.78
40	4	3	3	219227.54	219227.54	219227.54	0.00	0.79
40	6	2	2	195113.71	195113.71	195113.71	0.00	0.85
40	6	3	3	201338.25	201338.25	201338.25	0.00	0.84
avg (40)				199073.33	199073.33	199073.33	0.00	0.82
50	4	2	2	250479.23	250479.23	250479.23	0.00	0.97
50	4	3	3	278476.01	278476.01	278476.01	0.00	0.97
50	6	2	2	263769.77	263769.77	263769.77	0.00	1.18
50	6	3	3	293179.73	293179.73	293179.73	0.00	1.28
avg (50)				271476.19	271476.19	271476.19	0.00	1.10
60	4	2	2	263976.92	263976.92	263976.92	0.00	1.67
60	4	3	3	264029.28	264029.28	264029.28	0.00	1.73
60	6	2	2	222473.25	222473.25	222473.25	0.00	2.07
60	6	3	3	291979.18	291979.18	291979.18	0.00	2.35
avg (60)				260614.66	260614.66	260614.66	0.00	1.96
100	8	4	4	399442.47	399442.63	399443.14	0.30	6.41
100	8	5	5	438923.85	438923.90	438924.05	0.09	7.38
100	10	4	4	344216.84	344217.11	344217.64	0.38	7.72
100	10	5	5	376473.23	376473.42	376473.86	0.28	7.78
avg (100)				389764.10	389764.27	389764.67	0.26	7.32
150	8	4	4	438156.93	438157.42	438158.64	0.72	9.64
150	8	5	5	440416.07	440416.44	440417.48	0.61	8.85
150	10	4	4	466864.57	466864.99	466866.19	0.70	10.54
150	10	5	5	569988.53	569988.92	569990.11	0.68	10.17
avg (150)				478856.53	478856.94	478858.11	0.68	9.80
200	8	4	4	491986.12	491986.81	491988.10	0.86	10.76
200	8	5	5	495553.12	495553.59	495555.14	0.88	9.82
200	10	4	4	646227.25	646227.71	646228.81	0.71	11.93
200	10	5	5	696373.63	696374.24	696375.29	0.73	11.80
avg (200)				582535.03	582535.58	582536.83	0.79	11.08
overall avg				363719.97	363720.16	363720.63	0.29	5.35

performed to compare the performance of the three mathematical models, showing that the flow-based model outperforms the other two in terms of solution quality and speed. On the same instances, the accuracy of the ILS in finding good-quality solutions in a short time was proved. The ILS was also used to perform a series of tests on randomly generated medium- and large-size instances with up to 200 nodes, confirming its efficacy and robustness.

Additional computational tests were executed on small-, medium-, and large-size realistic instances derived from the Mashhad (Iran) distribution network, proving that our methods can be applied with profit even in a practical case.

Interesting future research directions include the application of the developed techniques to other related VRPs with precedence constraints and multiple visits. In addition, we are interested in studying the generalization of the VRPWDN to the case of multiple periods. In this generalization, one should first of all determine in which day inspecting the given nodes, and then creating the routes for each day.

Acknowledgements

This work was supported in part by Research Deputy of Ferdowsi University of Mashhad, under Grant No. 48185, and by University of Modena and Reggio Emilia, under grant FAR 2021. This support is gratefully acknowledged.

References

Allahyari, S., Salari, M., and Vigo, D. (2015). A hybrid metaheuristic algorithm for the multi-depot covering tour vehicle routing problem. *European Journal of Operational Research*, 242(3):756–768.

Atefi, R., Salari, M., Coelho, L. C., and Renaud, J. (2018). The open vehicle routing problem with decoupling points. *European Journal of Operational Research*, 265(1):316–327.

Aziez, I., Côté, J.-F., and Coelho, L. C. (2020). Exact algorithms for the multi-pickup and delivery problem with time windows. *European Journal of Operational Research*, 284(3):906–919.

Balas, E., Fischetti, M., and Pulleyblank, W. R. (1995). The precedence-constrained asymmetric traveling salesman polytope. *Mathematical Programming*, 68(1):241–265.

Battarra, M., Cordeau, J.-F., and Iori, M. (2014). Chapter 6: Pickup-and-delivery problems for goods transportation. In Toth, P. and Vigo, D., editors, *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, pages 161–191. SIAM.

Bektaş, T. and Gouveia, L. (2014). Requiem for the Miller–Tucker–Zemlin subtour elimination constraints? *European Journal of Operational Research*, 236(3):820–832.

Bruck, B. P. and Iori, M. (2017). Non-elementary formulations for single vehicle routing problems with pickups and deliveries. *Operations Research*, 65(6):1597–1614.

de Winter, C., Palleti, V., Worm, D., and Kooij, R. (2019). Optimal placement of imperfect water quality sensors in water distribution networks. *Computers & Chemical Engineering*, 121:200–211.

Desaulniers, G., Madsen, O. B., and Ropke, S. (2014). Chapter 5: The vehicle routing problem with time windows. In Toth, P. and Vigo, D., editors, *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, pages 119–159. SIAM.

Desrochers, M. and Laporte, G. (1991). Improvements and extensions to the Miller–Tucker–Zemlin subtour elimination constraints. *Operations Research Letters*, 10(1):27–36.

Doerner, K. F. and Salazar-González, J.-J. (2014). Chapter 7: Pickup-and-delivery problems for people transportation. In Toth, P. and Vigo, D., editors, *Vehicle Routing: Problems, Methods, and Applications, Second Edition*, pages 193–212. SIAM.

Furtado, M. G. S., Munari, P., and Morabito, R. (2017). The pickup and delivery problem with time windows in the oil industry: model and branch-and-cut methods. *Gestão & Produção*, 24:501–513.

Haddadene, S. R. A., Labadie, N., and Prodhon, C. (2016). A GRASP $\times$ ILS for the vehicle routing problem with time windows, synchronization and precedence constraints. *Expert Systems with Applications*, 66:274–294.

Hanafi, S., Mansini, R., and Zanotti, R. (2020). The multi-visit team orienteering problem with precedence constraints. *European Journal of Operational Research*, 282(2):515–529.

Hernández-Pérez, H., Landete, M., and Rodriguez-Martin, I. (2021). The single-vehicle two-echelon one-commodity pickup and delivery problem. *Computers & Operations Research*, 127:105152.

Kara, I. (2011). Arc based integer programming formulations for the distance constrained vehicle routing problem. In *3rd IEEE International Symposium on Logistics and Industrial Informatics*, pages 33–38. IEEE.

Karaoglan, I., Altiparmak, F., Kara, I., and Dengiz, B. (2012). The location-routing problem with simultaneous pickup and delivery: Formulations and a heuristic approach. *Omega*, 40(4):465–477.

Koç, Ç., Laporte, G., and Tükenmez, İ. (2020). A review of vehicle routing with simultaneous pickup and delivery. *Computers & Operations Research*, 122:104987.

Lourenço, H. R., Martin, O. C., and Stützle, T. (2019). Iterated local search: Framework and applications. In Potvin, J.-Y. and Gendreau, M., editors, *Handbook of Metaheuristics, Third Edition*, pages 129–168. Springer.

Moon, C., Kim, J., Choi, G., and Seo, Y. (2002). An efficient genetic algorithm for the traveling salesman problem with precedence constraints. *European Journal of Operational Research*, 140(3):606–617.

Mor, A. and Speranza, M. G. (2022). Vehicle routing problems over time: a survey. *Annals of Operations Research*, pages 1–21.

Mukherjee, R., Diwekar, U. M., and Vaseashta, A. (2017). Optimal sensor placement with mitigation strategy for water network systems under uncertainty. *Computers & Chemical Engineering*, 103:91–102.

Naji-Azimi, Z. and Salari, M. (2014). The time constrained maximal covering salesman problem. *Applied Mathematical Modelling*, 38(15-16):3945–3957.

Naserizade, S. S., Nikoo, M. R., and Montaseri, H. (2018). A risk-based multi-objective model for optimal placement of sensors in water distribution system. *Journal of Hydrology*, 557:147–159.

Pereira, D. L., Alves, J. C., and Moreira, M. C. d. O. (2020). A multiperiod workforce scheduling and routing problem with dependent tasks. *Computers & Operations Research*, 118:104930.

Quttineh, N.-H., Larsson, T., Lundberg, K., and Holmberg, K. (2013). Military aircraft mission planning: a generalized vehicle routing model with synchronization and precedence. *EURO Journal on Transportation and Logistics*, 2(1-2):109–127.

Rathi, S. and Gupta, R. (2014). Sensor placement methods for contamination detection in water distribution networks: A review. *Procedia Engineering*, 89:181–188.

Razali, N. M. (2015). An efficient genetic algorithm for large scale vehicle routing problem subject to precedence constraints. *Procedia - Social and Behavioral Sciences*, 195:1922–1931.

Sabouhi, F., Bozorgi-Amiri, A., Moshref-Javadi, M., and Heydari, M. (2019). An integrated routing and scheduling model for evacuation and commodity distribution in large-scale disaster relief operations: a case study. *Annals of Operations Research*, 283(1):643–677.

Salman, R., Ekstedt, F., and Damaschke, P. (2020). Branch-and-bound for the precedence constrained generalized traveling salesman problem. *Operations Research Letters*, 48(2):163–166.

Sarin, S. C., Sherali, H. D., and Bhootra, A. (2005). New tighter polynomial length formulations for the asymmetric traveling salesman problem with and without precedence constraints. *Operations Research Letters*, 33(1):62–70.

Seok Jeong, H. and Abraham, D. M. (2006). Operational response model for physically attacked water networks using NSGA-II. *Journal of Computing in Civil Engineering*, 20(5):328–338.

Sigurd, M., Pisinger, D., and Sig, M. (2004). Scheduling transportation of live animals to avoid the spread of diseases. *Transportation Science*, 38(2):197–209.

Silva, M. M., Subramanian, A., and Ochi, L. S. (2015). An iterated local search heuristic for the split delivery vehicle routing problem. *Computers & Operations Research*, 53:234–249.

Sun, P., Veelenturf, L. P., Dabia, S., and Van Woensel, T. (2018). The time-dependent capacitated profitable tour problem with time windows and precedence constraints. *European Journal of Operational Research*, 264(3):1058–1073.

Toth, P. and Vigo, D. (2014). *Vehicle Routing: Problems, Methods, and Applications*. SIAM, 2nd edition.

Vansteenwegen, P., Souffriau, W., Berghe, G. V., and Van Oudheusden, D. (2009). Iterated local search for the team orienteering problem with time windows. *Computers & Operations Research*, 36(12):3281–3290.

Wolfinger, D. and Salazar-González, J.-J. (2021). The pickup and delivery problem with split loads and transshipments: A branch-and-cut solution approach. *European Journal of Operational Research*, 289(2):470–484.

World Health Organization (2019). Drinking-water fact sheet.

Zhang, Q., Wu, Z. Y., Zhao, M., Qi, J., Huang, Y., and Zhao, H. (2016). Leakage zone identification in large-scale water distribution systems using multiclass support vector machines. *Journal of Water Resources Planning and Management*, 142(11):04016042.