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

Reza Atefi[a] and Manuel Iori[b] and Majid Salari[a] and Dario Vezzali[b,c]

[a]Department of Industrial Engineering, Ferdowsi University of Mashhad, P.O. Box 91779-48951, Mashhad, Iran; [b]Department of Sciences and Methods for Engineering, University of Modena and Reggio Emilia, Via Amendola 2, 42122, Reggio Emilia, Italy;[c]"Marco Biagi" Foundation, University of Modena and Reggio Emilia, Largo Marco Biagi 10, 41121, Modena, Italy

ARTICLE HISTORY
Compiled December 3, 2023

Abstract
In this work, we introduce a generalisation of the 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 centre. The key must then be returned to the same key centre after the test has been performed, thus introducing precedence constraints and multiple visits in the routes. To solve the problem, a Mixed Integer Linear Programming model 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 and real-world instances.

KEYWORDS
OR in Service Industries; Vehicle Routing Problem; Water Distribution Networks; Mixed Integer Linear Programming; 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 consumers and may lead to diseases. According to a report released by the *World Health Organization*, contaminated drinking water is estimated to cause 485,000 diarrhoeal deaths each year (World Health Organization, 2019).

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. Moreover, supply, treatment, and distribution of drinking water in urban networks require substantial expenses. Timely control of *Water Distribution Networks* (WDNs) is thus of fundamental importance.

In this paper, a new variant of the *Vehicle Routing Problem* (VRP) in the context of WDNs is proposed. In this problem, a set of technicians must visit a set of nodes

CONTACT Majid Salari. Email: msalari@um.ac.ir

within a network to evaluate the quality of water. 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 must visit a specified node at which the key is located, called *key centre* 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 must take the key back to its original key centre before returning to the depot where they started their route. Note that it is not compulsory to visit the key centre immediately before and after the well; in other words, the technicians can keep the key with them while visiting other nodes. If a key centre has the keys for several demand nodes and these nodes are visited by the same vehicle, the keys can be collected in a single visit. 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 travelling time. The aim of the problem is to minimise the total route duration.

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, 14 treatment plants, 356 wells and 8 key centres. Apart from the real-world application, the problem is of broad interest as it models routing problems in service industries where equipment pieces must be collected from an equipment centre before the execution of the service and then returned to the same equipment centre at the end of the activity. To solve the problem, we propose a *Mixed Integer Linear Programming* (MILP) model, and an *Iterated Local Search* (ILS) algorithm. While the model 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 in short computing times.

In brief, our work makes a number of valuable contributions, namely:

- define a new mathematical formulation for a practical VRP variant with precedence constraints, multiple visits, and maximum route duration constraints;
- implement an ILS algorithm using classical neighbourhood structures adapted to the specific characteristics of the problem and a new intensification phase;
- evaluate the performance of the proposed methods on randomly created instances;
- perform a comparison with the *Asymmetric Distance-Constrained VRP* (AD-VRP) using the benchmark instances by Almoustafa et al. (2013);
- compare the ILS solutions with those used in practice by the company, showing that the ILS can lead to good improvements in the real-world application;
- solve larger realistic instances derived from the real case study, thus demonstrating that the ILS performs well also on a larger scale.

The remainder of the paper is organised as follows. In Section 2, the relevant literature is revised. The problem is formally described in Section 3. Sections 4 and 5 present the MILP model 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 to Toth and Vigo (2014) for an extensive overview, and to Mor and Speranza (2022) for a recent survey. The problem we face generalises the VRP by considering precedence constraints and multiple visits. In this section, we only revise routing problems involving these two features, with a focus on real-world applications and algorithmic aspects.

Precedence constraints were first addressed in the seminal work by Balas et al. (1995) on the asymmetric *Travelling Salesman Problem* (TSP). Since then, they have been widely investigated for problems involving a single vehicle (see, e.g., Moon et al. 2002, Sarin et al. 2005, Sun et al. 2018, and Salman et al. 2020), as well as multiple vehicles (see, e.g., Razali 2015 and Haddadene et al. 2016). In addition, precedence constraints naturally arise for *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 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.

Recently, Aziez et al. (2020) studied a multi-PDP with time windows. 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 generalisations 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, 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.

A common application that shares many similarities to the problem presented here is the *Technician Routing and Scheduling Problem*, which is a generalisation of the *Workforce Scheduling and Routing Problem* (see, e.g., Castillo-Salazar et al. 2016 for a detailed review). Within this context, Zamorano and Stolletz (2017) addressed a related real-world application arising at a company providing external maintenance services for electric forklifts. The authors defined a novel MILP formulation and solved it using two alternative branch-and-price algorithms based on different decomposition schemes. Another interesting related application, originating in the context of electronic transaction equipment maintenance and repair services and involving some precedence constraints, was described by Mathlouthi et al. (2018), who later solved it by means of a *Tabu Search* algorithm (Mathlouthi et al., 2021).

A similar problem structure may also be found in healthcare and home healthcare logistics. Recent works in this field are those of Anaya-Arenas et al. (2021), who addressed a real-world routing application with precedence constraints arising in the transportation of biomedical samples from specimen collection centres to specific laboratories, and Polnik et al. (2021), who addressed a real-world home healthcare problem with time windows and synchronisation constraints.

We notice that none of the cited applications require the technicians to collect, use and later bring back to a centre a specific equipment. This feature imposes at least two visits to the equipment centre and is the main characteristic of the new problem we address in this paper.

Given the NP-hardness of the VRP and of many of its variants, solving these problems by complete enumeration may take too long. This particularly applies to practical applications where the problem scale is large. For this reason, metaheuristic algorithms are frequently used to tackle real-world VRPs (see, e.g., Elshaer and Awad 2020). Among single solution-based metaheuristics, the ILS is one of the most popular, and it is also used to solve many PDP variants (see, e.g., Subramanian et al. 2010, 2013, Li et al. 2015, and Haddadene et al. 2016 for relevant implementations).

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

In many problems arising in service industries, technicians must visit different nodes in a network and perform some required services. The nodes to be visited, called for simplicity *demand nodes* in the following, can be divided into two types:

(1) *Type I*: regular nodes. For these nodes, the technicians can go directly on site and perform the required services;
(2) *Type II*: special nodes. For these nodes, the technicians need an equipment to perform the services. So, they must visit first a specified equipment centre, and take the equipment. Once all services have been completed at the special node, the equipment must be returned to its original equipment centre, thus imposing a second visit.

A simple illustrative example is depicted in Figure 1. It comprises two routes starting and ending at the depot. One of them visits just regular nodes, so demand nodes of type I. The second also visits a special node, and is thus forced to pass twice by the corresponding equipment centre.

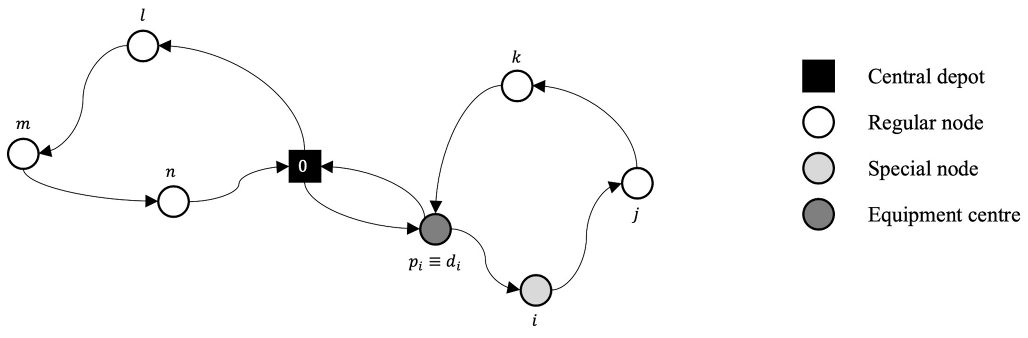


Figure 1. An illustrative example of the problem

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 equipment centres. With each node $i \in V_2$, we associate a predecessor $p_i \in V_3$ and a successor $d_i \in V_3$. In our application, p_i and d_i correspond to a unique equipment centre, 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 must be visited exactly once, while each vehicle visits a particular equipment centre at most once for picking up all the equipment pieces, and then another single time for delivering all the equipment pieces that were previously collected. This implies that, in case a centre has the equipment pieces for multiple demand nodes and these nodes are visited by a unique vehicle, then such equipment pieces must be collected all together in a unique visit (to p_i), and then later delivered all together in another visit (to d_i). We recall that it is not compulsory to visit p_i immediately before i. In other words, the vehicle can collect the equipment 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 travelling time t_{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., $t_{ij} \leq t_{ik} + v_k + t_{kj}$ 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 travelling times of the nodes and arcs covered by the vehicle, and it must not exceed a maximum duration L. Whenever a route visits a node i of type II, then it must visit also p_i and d_i.

In our application (described in detail in Section 6.6 below), equipment pieces are keys, equipment centres are key centres, regular nodes are households, shops, reservoirs, tanks and treatment plants, and special nodes are wells. The resulting problem, which we call *Vehicle Routing Problem for Water Distribution Networks* (VRPWDN), requires visiting all demand nodes, while satisfying precedence constraints, multiple visits, and maximum route duration constraints, with the aim of minimising the total route duration. The VRPWDN is NP-hard in the strong sense, because it generalises the VRP. In the next sections, we attempt its solution through a MILP model and an ILS algorithm.

4. Flow-based Model

In this section, we define a flow-based model that formally describe the VRPWDN. Such model 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). Two alternative mathematical models, which provided weaker computational results, are reported in the supplemental online material.

Let x_{ijk} be a binary variable taking value 1 if arc (i, j) is covered by vehicle k and 0 otherwise, and f_{ijk} be a variable representing the "load" of vehicle k when travelling 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:

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

subject to

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

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

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

$$\sum_{i \in V \setminus \{n+1\}} \sum_{j \in V \setminus \{0\}} (t_{ij} + v_i) x_{ijk} \leq L \qquad k \in K \qquad (5)$$

$$\sum_{j \in V \setminus \{0\}} f_{0jk} = 0 \qquad k \in K \qquad (6)$$

$$\sum_{i \in V \setminus \{n+1\}} f_{i,n+1,k} = \sum_{i \in V \setminus \{n+1\}} \sum_{j \in V \setminus \{0\}} x_{ijk} - 1 \qquad k \in K \qquad (7)$$

$$\sum_{j \in V \setminus \{0\}} f_{ijk} \geq \sum_{j \in V \setminus \{n+1\}} (f_{jik} + x_{jik}) \qquad i \in V \setminus \{0, n+1\}, k \in K \qquad (8)$$

$$\sum_{j \in V \setminus \{0\}} (f_{p_i jk} - f_{ijk} + x_{ijk}) \leq (n-1)(1 - \sum_{j \in V \setminus \{0\}} x_{ijk}) \qquad i \in V_2, k \in K \qquad (9)$$

$$\sum_{j \in V \setminus \{0\}} f_{p_i jk} \geq \sum_{j \in V \setminus \{0\}} x_{ijk} \qquad i \in V_2, k \in K \qquad (10)$$

$$\sum_{j \in V \setminus \{0\}} (f_{d_i jk} - f_{ijk}) \geq \sum_{j \in V \setminus \{0\}} x_{ijk} \qquad i \in V_2, k \in K \qquad (11)$$

$$x_{ijk} \in \{0, 1\} \qquad i, j \in V, k \in K \qquad (12)$$

$$0 \leq f_{ijk} \leq (n-1) x_{ijk} \qquad i, j \in V, k \in K \qquad (13)$$

Objective function (1) minimises the total route duration. 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) ensure that each node i has exactly one incoming and one outgoing arc. The maximum duration of each route is bounded by means of constraints (5). Constraints (6) and (7) impose the load on vehicle k when leaving 0 and entering $n + 1$, respectively. Constraints (8) impose the load conservation at node i. Constraints (9)–(11) guarantee the respect of precedence constraints. Constraints (12) define the domain of the x_{ijk} variables, while constraints (13) impose lower and upper bounds on the f_{ijk} variables.

We removed some unnecessary variables from the model by fixing their values to 0, namely: $x_{ip_ik} = x_{d_iik} = 0$ for $i \in V_2, k \in K$; $x_{0d_jk} = 0$ and $x_{0jk} = 0$ for $j \in V_2, k \in K$; $x_{p_i,n+1,k} = 0$ and $x_{i,n+1,k} = 0$ for $i \in V_2, k \in K$. Furthermore, the above model can be improved by the addition of the following constraints:

$$\sum_{j \in V \setminus \{n+1\}} x_{jp_ik} + \sum_{j \in V \setminus \{0\}} x_{d_ijk} \geq 2 \sum_{j \in V \setminus \{n+1\}} x_{jik} \qquad i \in V_2, k \in K \qquad (14)$$

$$\sum_{j \in V \setminus \{0\}} (f_{d_ijk} - f_{p_ijk}) \geq \sum_{l \in V \setminus \{n+1\} : p_i = p_l} \sum_{j \in V \setminus \{0\}} x_{ljk} \qquad i \in V_2, k \in K \qquad (15)$$

$$\sum_{l \in V \setminus \{n+1\}} (f_{lik} - f_{ljk}) + n x_{ijk} + (n-2) x_{jik} \leq (n-1) \qquad i, j \in V \setminus \{0, n+1\}, k \in K \qquad (16)$$

Constraints (14) 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 (15) enforce that the difference of loads leaving d_i and p_i must be greater than or equal to the number of arcs covered when going from p_i to d_i. Constraints (16) are derived from the lifted constraints proposed by Desrochers and Laporte (1991).

5. Iterated Local Search

We developed an ILS algorithm with the purpose of finding good-quality VRPWDN solutions within a limited computational effort. 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, Subramanian et al. 2010, 2013, Li et al. 2015, Silva et al. 2015, Haddadene et al. 2016) as well as on practical applications (see, e.g., Atefi et al. 2018 and Anaya-Arenas et al. 2021).

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 constructive heuristic algorithm (line 1), and then we improve it with a local search procedure (line 2). The incumbent solution, x^*, is inserted in the set of best known solutions obtained during the search, called $BKSet$ (line 3). Next, we execute two phases, one after the other.

In the first phase, a perturbation is applied to x^* followed by a local search on the perturbed solution, x' (lines 5–7). The perturbation is randomly selected between two shaking procedures described in the following. Let $z(x) = \sum_{k \in K} z_k(x)$ be the total route duration for solution x and let $l(x) = \max_{k \in K} z_k(x)$ be the maximum route duration for solution x, where $z_k(x) = \sum_{i \in \mathcal{V}_k(x) \setminus \{n+1\}} \sum_{j \in \mathcal{V}_k(x) \setminus \{0\}} (t_{ij} + v_i)$ is the total route duration of vehicle k for solution x and $\mathcal{V}_k(x) \subseteq V$ is the set of nodes visited by vehicle k for solution x. In case $x^{*\prime}$ has lower total route duration than x^*, or same total route duration but lower maximum route duration, then we use it to update x^* (line 9). In such a case, we also insert $x^{*\prime}$ in $BKSet$ (line 10). This set contains at most the best β solutions found during the search, having the lowest $z(x)$ and breaking ties by lowest $l(x)$. If, instead, $x^{*\prime}$ does not improve x^*, then x^* is used 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 13 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 (lines 16–18), 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 total route duration, then the search restarts from the beginning of the first phase (line 22). To the best of our knowledge, no similar intensification phase can be found in the best-known ILS implementations for PDPs.

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

5.1. Initialisation Procedure

Algorithm 2 gives the `Initialisation` 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 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 into the route in the positions that lead to the minimum extra route duration, which is computed as $ed(h, i, j) = t_{hi} + v_i + t_{ij} - t_{hj}$ if, for example, node i is inserted between node h and node j. Note that the insertion of node i or tuple (p_i, i, d_i) into an existing

Algorithm 1 Iterated Local Search (ILS)

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

route is led by the `CheapestInsertion` procedure, which evaluates among the $|K|$ routes the best candidate for the expansion. In particular, in case tuple (p_i, i, d_i) is to be inserted into an existing route, node i is initially inserted into the route in the position that leads to the minimum extra route duration; then, p_i and d_i are inserted into the route in the positions that lead to the minimum extra route duration between node 0 and node i and between node i and node $n + 1$, respectively, in such a way that precedence constraints are satisfied. At line 22, the algorithm checks whether the solution is feasible. If not, then the whole procedure is repeated from scratch.

5.2. Local Search

The `LocalSearch` procedure invokes, one after the other, the following neighbourhood structures:

LS1 *Swap intra-route*: after removing the nodes belonging to V_3 (i.e., the equipment centres), two sequences with a maximum of three consecutive nodes (i.e., sequences of one, two, or three consecutive nodes) within the same route are swapped. The equipment centres that were previously removed from the route are reinserted after the swap in the positions that lead to the minimum extra route duration and satisfy precedence constraints. An example of swap intra-route between two sequences, the first involving only node j and the second only node k, is given in Figure 2-(a). Note that in this example, like in the following

Algorithm 2 Initialisation Procedure

```
 1: $\mathcal{S}, \mathcal{V} \leftarrow \emptyset$
 2: for $k \leftarrow 1$ to $|K|$ do                         ▷ Initialisation of $|K|$ routes in parallel
 3:     $i \leftarrow$ Rand$\{\{1, ..., |V_1 \cup V_2|\} \setminus \mathcal{V}\}$
 4:     $\mathcal{V} \leftarrow \mathcal{V} \cup \{i\}$                    ▷ Add $i$ to the set of visited nodes
 5:     if $i \in V_2$ then
 6:         $r_k \leftarrow (0, p_i, i, d_i, n+1)$
 7:         Insert($r_k, \mathcal{S}$)
 8:     else
 9:         $r_k \leftarrow (0, i, n+1)$
10:         Insert($r_k, \mathcal{S}$)
11:     end if
12: end for
13: for $j \leftarrow 1$ to $|V_1 \cup V_2| - |K|$ do                 ▷ Expansion of existing routes
14:     $i \leftarrow$ Rand$\{\{1, ..., |V_1 \cup V_2|\} \setminus \mathcal{V}\}$
15:     $\mathcal{V} \leftarrow \mathcal{V} \cup \{i\}$
16:     if $i \in V_2$ then
17:         CheapestInsertion($(p_i, i, d_i), r_k \in \mathcal{S}$)
18:     else
19:         CheapestInsertion($i, r_k \in \mathcal{S}$)
20:     end if
21: end for
22: if Feasible($\mathcal{S}$) $= 1$ then
23:     continue
24: else
25:     Go to line 1
26: end if
27: return $\mathcal{S}$
```

ones, nodes p_i and d_i have different positions to make it as general as possible;

LS2 *Swap inter-route*: after removing the nodes belonging to V_3, two sequences with a maximum of three consecutive nodes are swapped between two routes. The equipment centres that were previously removed from the routes are reinserted after the swap in the positions that lead to the minimum extra route duration and satisfy precedence constraints. If some nodes belonging to V_3 are no longer needed in one of the two routes, their reinsertion is avoided in that route. An example of swap inter-route between nodes j and n is given in Figure 2-(b);

LS3 *Relocate intra-route*: after removing the nodes belonging to V_3, a sequence with a maximum of three consecutive nodes is removed and reinserted in a different position within the same route. The equipment centres that were previously removed from the route are reinserted after the relocation in the positions that lead to the minimum extra route duration and satisfy precedence constraints. An example of relocate intra-route of node i is given in Figure 2-(c);

LS4 *Relocate inter-route*: after removing the nodes belonging to V_3, a sequence with a maximum of three consecutive nodes is removed and reinserted in a different route. The equipment centres that were previously removed from the route are reinserted after the relocation in the positions that lead to the minimum extra route duration and satisfy precedence constraints. If some nodes belonging to V_3 are no longer needed in one of the two routes, their reinsertion is avoided in that route. An example of relocate inter-route of node n is given in Figure 2-(d);

LS5 *3-opt*: after removing the nodes belonging to V_3, the classical 3-opt algorithm of Lin (1965) is applied to the remaining nodes. The equipment centres that were previously removed from the route are reinserted in the positions that lead to the minimum extra route duration and satisfy precedence constraints. An example of the 3-opt algorithm applied to subpath (l, j, k, m) is given in Figure 2-(e).

All the neighbourhood structures invoked by the `LocalSearch` procedure are classical efficient neighbourhood structures from the vehicle routing literature that have been adapted to the problem at hand. Similar neighbourhood structures can be found, for example, in Subramanian et al. (2010). What is new is the way in which the nodes belonging to V_3 are treated within these algorithms. Indeed, unlike the work of Subramanian et al. (2010), no feasibility check is needed after each local search move, as feasibility is guaranteed by the removal and reinsertion of the equipment centres.

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 duration 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 optimised by a 3-opt algorithm. A threshold of γ iterations is set to limit the number of attempts.

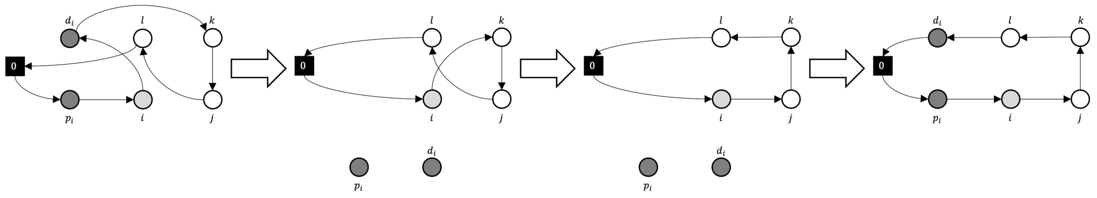
Each time the `LocalSearch` procedure is invoked, LS1–LS5 are repeated sequentially, from LS1 to LS5, as long as an improvement is found. If at least one of the neighbourhood structures finds an improvement, the entire search is repeated starting from LS1. Note that each neighbourhood structure looks for the first improvement and implements it; then the search continues until all possible combinations have been checked (except LS5, for which a maximum number of iterations is set).

5.3. Shaking Procedure

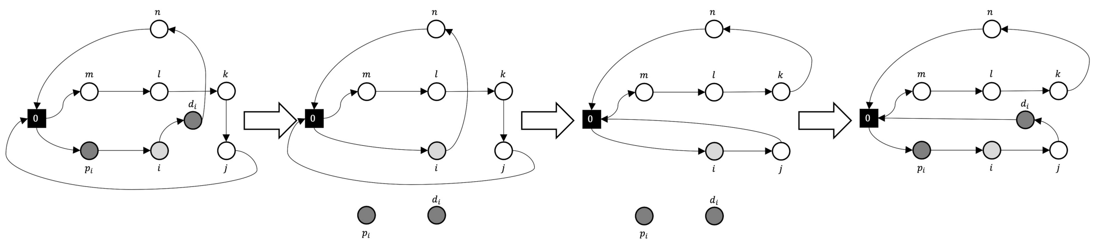
To perturb a solution, we randomly select, with same probability, one of the two following procedures.

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 total route duration 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 total route duration of the perturbed solution is not worse than $\alpha z(x^*)$;

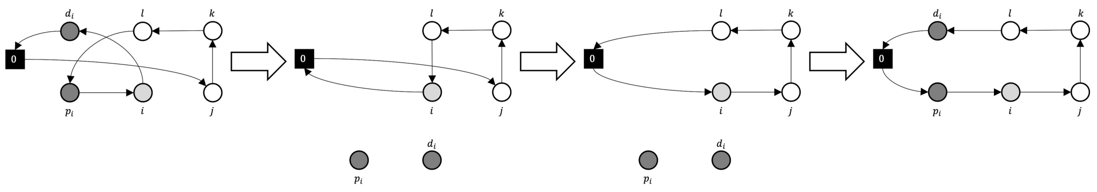
S2 *Shaking 2:* compute the duration saving obtained by removing any node from the solution, similarly to what is done in LS5. Then use the roulette wheel mechanism to select a node $i \in V \setminus \{0, n+1\}$, and remove i from its route. The removal procedure is iterated until at least α percent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



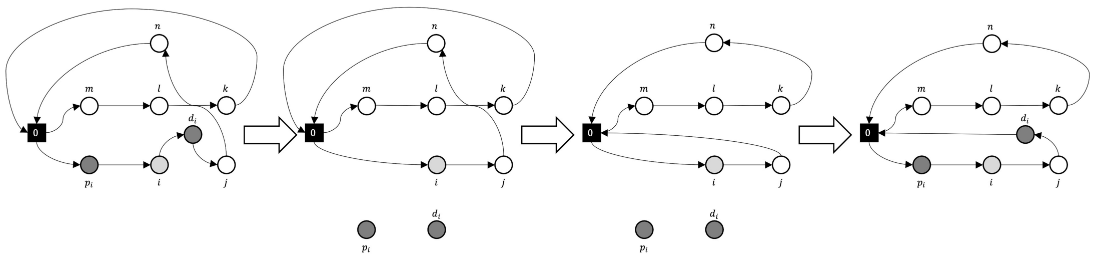
(a) Swap intra-route



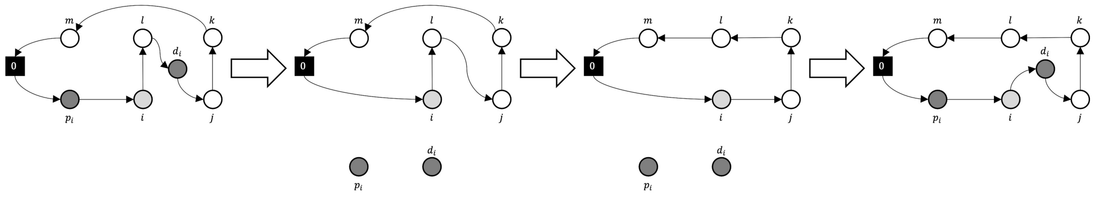
(b) Swap inter-route



(c) Relocate intra-route



(d) Relocate inter-route



(e) 3-opt

Figure 2. Illustration of the neighbourhood structures

6. Computational Results

In this section, we present the results of the computational tests performed to assess the performance of the proposed methods. The mathematical models and the ILS were coded in C++. 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. A time limit of 3,600 CPU seconds was imposed on each execution. In Section 6.1, we describe the sets of randomly-created instances that we used for our tests. The results obtained for the flow-based model are reported in Section 6.2, while the behaviour of the ILS is analysed in Sections 6.3 and 6.4. A comparison on benchmark instances from the ADVRP literature is reported in Section 6.5, while in Section 6.6 we report the results of additional computational experiments performed on realistic instances derived from the case study.

6.1. Randomly-created Instances

We created several random instances to assess 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*: 24 instances with $|V_1 \cup V_2|$=10, $(|V_2|, |V_3|) \in \{(1,1), (2,1), (2,2), (4,2), (4,3)\}$, and $|K| \in \{1,2\}$; 30 instances with $|V_1 \cup V_2|$=15, $(|V_2|, |V_3|) \in \{(3,2), (3,3), (4,2), (4,3), (7,3), (7,4)\}$, and $|K| \in \{2,3\}$; 30 instances with $|V_1 \cup V_2|$=20, $(|V_2|, |V_3|) \in \{(2,2), (3,2), (3,3), (5,3), (10,4), (10,6)\}$, and $|K| \in \{2,3,4\}$;
- *Medium- and large-size*: 30 instances with $|V_1 \cup V_2|$=50, $(|V_2|, |V_3|) \in \{(5,5), (8,8), (10,5), (10,8), (25,10), (25,15)\}$, and $|K| \in \{5,8,10\}$; 30 instances with $|V_1 \cup V_2|$=100, $(|V_2|, |V_3|) \in \{(5,5), (10,5), (10,10), (15,10), (50,20), (50,30)\}$, and $|K| \in \{10,15,20\}$; 30 instances with $|V_1 \cup V_2|$=200, $(|V_2|, |V_3|) \in \{(10,10), (20,10), (20,20), (30,20), (100,40), (100,50)\}$, and $|K| \in \{15,20,30\}$.

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, and converted into times using a fixed multiplier of 30 km/h (i.e., assuming that this is the average speed of vehicles along the network). The maximum duration is set to $L = 1.5(\sum_{i \in V_1 \cup V_2} \bar{t}_i + |K| \sum_{i \in V_3 \cup \{0\}} \bar{t}_i)/|K|$, where $\bar{t}_i$ is the average travel time of the arcs leaving i, computed as $\bar{t}_i = \sum_{j \in V \setminus \{i\}} t_{ij}/(|V|-1)$ 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 favour future research on the problem, the randomly-created instances have been made publicly available at `https://github.com/regor-unimore/Vehicle-Routing-Problem-for-Water-Distribution-Networks`.

6.2. Flow-based Model Performance

In this section, the performance of the flow-based model from Section 4 is investigated. 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 computed as $100(z_{ub} - z_{lb})/z_{ub}$, and column "Sec." the average run time. In particular, the lower bound values are those obtained at the end of the execution (i.e., those values obtained upon proving optimality or reaching a time/memory limit). 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. To assess the performance of the proposed valid inequalities, all instances were solved with and without the addition of the valid inequalities. More detailed computational results, comparing the performance of the flow-based model with two alternative models that we developed, are reported in the supplemental online material.

Table 1. Performance of the flow-based model (three instances per line).

				without valid inequalities					with valid inequalities				
$\|V_1 \cup V_2\|$	$\|V_2\|$	$\|V_3\|$	$\|K\|$	z_{lb}	z_{ub}	%gap	Sec.	opt	z_{lb}	z_{ub}	%gap	Sec.	opt
10	1	1	1	706.39	706.39	0.00	1.68	3/3	706.39	706.39	0.00	1.13	3/3
10	1	1	2	730.91	730.91	0.00	4.31	3/3	730.91	730.91	0.00	3.26	3/3
10	2	1	1	743.25	743.25	0.00	3.79	3/3	743.25	743.25	0.00	3.13	3/3
10	2	1	2	793.75	793.75	0.00	10.81	3/3	793.75	793.75	0.00	9.45	3/3
10	2	2	1	803.96	803.96	0.00	17.28	3/3	803.96	803.96	0.00	16.73	3/3
10	2	2	2	833.78	833.78	0.00	80.97	3/3	833.78	833.78	0.00	75.01	3/3
10	4	2	2	935.49	935.49	0.00	164.14	3/3	935.49	935.49	0.00	74.20	3/3
10	4	3	2	1058.61	1058.61	0.00	323.86	3/3	1058.61	1058.61	0.00	274.33	3/3
sum/avg (10)				825.77	825.77	0.00	75.85	24/24	825.77	825.77	0.00	57.16	24/24
15	3	2	2	1036.61	1036.61	0.00	584.13	3/3	1036.61	1036.61	0.00	509.71	3/3
15	3	2	3	1046.02	1049.45	0.33	1729.31	2/3	1049.45	1049.45	0.00	1325.68	3/3
15	3	3	2	1109.70	1114.68	0.45	1298.50	2/3	1114.68	1114.68	0.00	1104.10	3/3
15	3	3	3	1152.65	1161.09	0.73	1650.61	2/3	1153.10	1160.74	0.65	1358.60	2/3
15	4	2	2	1036.61	1036.61	0.00	581.15	3/3	1036.61	1036.61	0.00	510.81	3/3
15	4	2	3	1084.12	1084.12	0.00	1483.72	3/3	1084.12	1084.12	0.00	1108.56	3/3
15	4	3	2	1141.60	1149.80	0.71	1692.32	2/3	1142.24	1149.80	0.69	1420.98	2/3
15	4	3	3	1187.63	1207.11	1.61	1187.63	2/3	1190.64	1202.40	1.03	1430.51	2/3
15	7	3	3	1342.60	1342.60	0.00	1152.71	3/3	1342.60	1342.60	0.00	908.80	3/3
15	7	4	3	1366.61	1406.45	2.92	tlim	0/3	1372.35	1406.45	2.46	tlim	0/3
sum/avg (15)				1150.42	1158.85	0.67	1496.01	22/30	1152.24	1158.35	0.48	1327.77	24/30
20	2	2	2	1243.47	1278.85	2.77	2069.77	2/3	1243.73	1277.82	2.61	1354.86	2/3
20	2	2	3	1298.19	1338.84	3.04	1742.10	2/3	1294.95	1329.12	2.56	1458.53	2/3
20	3	2	2	1239.80	1274.10	2.69	1669.25	2/3	1243.52	1269.71	2.02	1271.07	2/3
20	3	2	3	1301.73	1301.73	0.00	1428.79	3/3	1301.73	1301.73	0.00	1127.63	3/3
20	3	3	2	1255.36	1299.51	3.40	3381.88	1/3	1260.69	1296.61	2.60	2661.64	1/3
20	3	3	3	1276.88	1321.84	3.40	tlim	0/3	1281.30	1321.84	2.97	tlim	0/3
20	5	3	2	1243.50	1283.32	3.10	3315,16	1/3	1241.98	1280.43	2.91	2986.79	1/3
20	5	3	3	1265.11	1326.69	4.64	tlim	0/3	1270.11	1322.70	3.84	2863.57	1/3
20	10	4	4	1738.23	1878.33	8.06	tlim	0/3	1754.91	1847.21	5.07	tlim	0/3
20	10	6	4	1848.47	1913.72	3.53	tlim	0/3	1856.02	1902.28	2.39	tlim	0/3
sum/avg (20)				1371.07	1421.69	3.46	2800.69	11/30	1374.89	1414.95	2.70	2452.41	12/30
overall sum/avg				1136.47	1157.56	1.48	1556.21	57/84	1138.48	1154.97	1.14	1366.40	60/84

From Table 1, we observe that the flow-based model finds optimal solutions for all the small-size instances with $|V_1 \cup V_2| = 10$, while for those with $|V_1 \cup V_2| = 15$ and $|V_1 \cup V_2| = 20$, the computer frequently ran out of memory because of the large model size. Therefore, we cannot prove the optimality of these solutions. However, we notice that the valid inequalities improve the performance of the model by reducing the average percentage gap and execution time. As one might expect, the average lower and upper bound values increase with the number of demand nodes but, more interestingly, for the same total number of demand nodes, the average lower and upper bound values increase more for those group of instances in which the number of special nodes is higher. This is understandable since an increase in the number of special nodes leads to an increase in the number of detours to the equipment centres (especially if a technician must visit more than one equipment centre within the same route). In particular, looking at the results obtained with the addition of the valid inequalities, we observe an increase in the average lower and upper bound values of up to 39% for instances having $|V_1 \cup V_2| = 10$ when $|V_2|$ goes from 1 to 4, up to 26% for instances having $|V_1 \cup V_2| = 15$ when $|V_2|$ goes from 3 to 7, and up to 44% for instances having $|V_1 \cup V_2| = 20$ when $|V_2|$ goes from 2 to 10. Similarly, we notice an increase both in terms of average percentage gap and execution time, which is justified by the increased complexity of the instances. Conversely, the number of instances solved to optimality decreases when the number of demand nodes increases and, for the same total number of demand nodes, when the number of special nodes increases. 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., between 53 and 55) is larger than the size of instances solved to optimality within the time limit.

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 $max_{iter} \in \{200, 500, 1000, 5000\}$. The results are reported in Table 2. For each combination of parameters, column "Sec." 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 $100(z_f - z_{all})/z_{all}$. Here, z_f is the value of the solution obtained by the given configuration f and $z_{all} = \min_f\{z_f\}$ is the value of the best solution obtained by all configurations.

Table 2. 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)	Sec.	%gap	Sec.	%gap	Sec.	%gap	Sec.	%gap	Sec.	%gap	Sec.	%gap	Sec.	%gap	Sec.	%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

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 3, the results of the ILS are compared with those obtained by the flow-based model on groups of three instances per line. We recall that columns "z_{lb}" and "z_{ub}" give the average lower and upper bound values, respectively, column "opt" the number of proven optimal solutions, and column "Sec." the average run time. The ILS was executed five times on each instance. We report the best, average and worst solution values achieved, as well as the average gap computed as $100(z_{avg} - z_{lb})/z_{avg}$, in columns "$z_{best}$", "$z_{avg}$", "$z_{worst}$" and "%gap", 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 "Sec.".

Table 3. Computational results on small-size instances (three instances per line)

				flow-based				ILS				
$\|V_1 \cup V_2\|$	$\|V_2\|$	$\|V_3\|$	$\|K\|$	z_{lb}	z_{ub}	Sec.	opt	z_{best}	z_{avg}	z_{worst}	%gap	Sec.
10	1	1	1	706.39	706.39	1.13	3/3	706.39	706.39	706.39	0.00	0.00
10	1	1	2	730.91	730.91	3.26	3/3	730.91	730.91	730.91	0.00	0.00
10	2	1	1	743.25	743.25	3.13	3/3	743.25	743.25	743.25	0.00	0.00
10	2	1	2	793.75	793.75	9.45	3/3	793.75	793.75	793.75	0.00	0.00
10	2	2	1	803.96	803.96	16.73	3/3	803.96	803.96	803.96	0.00	0.00
10	2	2	2	833.78	833.78	75.01	3/3	833.78	833.78	833.78	0.00	0.00
10	4	2	2	935.49	935.49	74.20	3/3	935.49	935.49	935.49	0.00	0.00
10	4	3	2	1058.61	1058.61	274.33	3/3	1058.61	1058.61	1058.61	0.00	0.00
sum/avg (10)				825.77	825.77	57.16	24/24	825.77	825.77	825.77	0.00	0.00
15	3	2	2	1036.61	1036.61	509.71	3/3	1036.61	1036.61	1036.61	0.00	0.14
15	3	2	3	1049.45	1049.45	1325.68	3/3	1049.45	1049.45	1049.45	0.00	0.22
15	3	3	2	1114.68	1114.68	1104.10	3/3	1114.68	1114.68	1114.68	0.00	0.14
15	3	3	3	1153.10	1160.74	1358.60	2/3	1155.96	1155.96	1155.96	0.25	0.24
15	4	2	2	1036.61	1036.61	510.81	3/3	1036.61	1036.61	1036.61	0.00	0.16
15	4	2	3	1084.12	1084.12	1108.56	3/3	1084.12	1084.12	1084.12	0.00	0.22
15	4	3	2	1142.24	1149.80	1420.98	2/3	1146.56	1146.56	1146.56	0.38	0.20
15	4	3	3	1190.64	1202.40	1430.51	2/3	1202.40	1202.40	1202.40	0.98	0.27
15	7	3	3	1342.60	1342.60	908.80	3/3	1342.60	1342.60	1342.60	0.00	0.21
15	7	4	3	1372.35	1406.45	tlim	0/3	1403.00	1403.00	1403.00	2.18	0.28
sum/avg (15)				1152.24	1158.35	1327.77	24/30	1157.20	1157.20	1157.20	0.43	0.21
20	2	2	2	1243.73	1277.82	1354.86	2/3	1275.44	1275.44	1275.44	2.49	0.26
20	2	2	3	1294.95	1329.12	1458.53	2/3	1316.03	1316.03	1316.03	1.60	0.33
20	3	2	2	1243.52	1269.71	1271.07	2/3	1262.52	1262.52	1262.52	1.50	0.35
20	3	2	3	1301.73	1301.73	1127.63	3/3	1301.73	1301.73	1301.73	0.00	0.41
20	3	3	2	1260.69	1296.61	2661.64	1/3	1277.25	1277.25	1277.25	1.30	0.35
20	3	3	3	1281.30	1321.84	tlim	0/3	1301.37	1301.37	1301.37	1.54	0.61
20	5	3	2	1241.98	1280.43	2986.79	1/3	1265.46	1265.46	1265.46	1.86	0.33
20	5	3	3	1270.11	1322.70	2863.57	1/3	1303.93	1303.93	1303.93	2.59	0.73
20	10	4	4	1754.91	1847.21	tlim	0/3	1833.52	1833.52	1833.52	4.29	0.46
20	10	6	4	1856.02	1902.28	tlim	0/3	1902.28	1902.28	1902.28	2.43	0.53
sum/avg (20)				1374.89	1414.95	2452.41	12/30	1403.95	1403.95	1403.95	2.07	0.44
overall sum/avg				1138.48	1154.97	1366.40	60/84	1150.63	1150.63	1150.63	1.06	0.23

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 in terms of average upper bound values (without proof of their optimality), with an average percentage gap from the average lower bound values of 0.43% for instances having $|V_1 \cup V_2| = 15$ and 2.07% for instances having $|V_1 \cup V_2| = 20$, which is acceptable given the significant advantage in terms of computational time.

In Table 4, in which column "σ_z" gives the average standard deviation, 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 having $|V_1 \cup V_2| = 100$ it becomes 0.56, while on those having $|V_1 \cup V_2| = 200$ it increases to 0.92, thus resulting in an overall average standard deviation of 0.49. This confirms the robustness of the algorithm. Concerning the run time, the ILS took on average 2.09 seconds to solve instances having $|V_1 \cup V_2| = 50$, 9.60 seconds for those having $|V_1 \cup V_2| = 100$, and 14.73 seconds for those having $|V_1 \cup V_2| = 200$. The overall average run time is 8.81 seconds, proving that the method is suitable for a quick use in practical situations.

Table 4. Computational results on medium- and large-size instances (three instances per line)

				ILS				
$\|V_1 \cup V_2\|$	$\|V_2\|$	$\|V_3\|$	$\|K\|$	z_{best}	z_{avg}	z_{worst}	σ_z	Sec.
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
50	25	10	10	3402.31	3402.31	3402.31	0.00	2.53
50	25	15	10	4364.78	4364.78	4364.78	0.00	3.10
avg (50)				3029.52	3029.52	3029.52	0.00	2.09
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
100	50	20	20	6376.07	6376.67	6377.64	0.67	12.99
100	50	30	20	6891.37	6892.08	6893.25	0.95	14.25
avg (100)				5141.40	5141.72	5142.64	0.56	9.60
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
200	100	40	30	9815.12	9815.76	9817.16	0.87	19.17
200	100	50	30	10806.37	10807.30	10808.64	0.97	24.04
avg (200)				9217.15	9217.91	9219.15	0.92	14.73
overall avg				5796.02	5796.38	5797.10	0.49	8.81

Finally, Table 5 reports a sensitivity analysis on the number of times each ILS component is invoked, column "#", and the average percentage of computational time needed, column "%", grouped by set of instances. We notice that the number of times LS1–LS5 are invoked increases with the size of the instances, and the same holds for S1 and S2. Also, 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. This is due to the increasing number of intra-route or inter-route combinations that must be checked by the ILS components, as well as the removal/reinsertion mechanism of the nodes belonging to V_3, the number of which increases with the size of the instances.

Table 5. Number of times each ILS component is invoked and percentage of the computational time needed

Set	LS1		LS2		LS3		LS4		LS5		S1		S2	
	#	%	#	%	#	%	#	%	#	%	#	%	#	%
Small-size	6540	5.37	6540	37.16	6540	27.91	6540	7.39	6540	20.70	894	0.38	979	1.09
Medium-size	8525	56.45	8525	10.15	8525	1.29	8525	21.44	8525	9.62	1107	0.22	1054	0.83
Large-size	9798	48.25	9798	4.38	9798	3.15	9798	34.51	9798	8.92	1118	0.12	1101	0.67

6.5. Comparison on Benchmark ADVRP Instances

In this section, we evaluate the effectiveness of the proposed ILS in solving the ADVRP, a variant of the Distance-Constrained VRP in which the distance matrix is asymmetric. The problem was introduced in the seminal work of Laporte et al. (1987) and recently studied by Almoustafa et al. (2013). The VRPWDN corresponds to the ADVRP when $V_2 = V_3 = \emptyset$ and $v_i = 0$ for all $i \in V$. Furthermore, travelling distances are considered instead of travelling times. The performance of the ILS was assessed on the benchmark ADVRP instances proposed by Almoustafa et al. (2013). In particular, the authors created 92 instances, 72 of which are small-size and 20 are large-size. For each instance, three problems were solved by attempting different values of the maximum travelling distance $D_{\max}$. The first attempted value, $D_{\max(1)}$, was set to $+\infty$, while the other values were set to $D_{\max(i)} = 0.90LT(i-1)$ for $i \in \{2,3\}$, with $LT(i)$ representing the longest route in the solution obtained with maximum travelling distance $D_{\max(i-1)}$.

The results of our algorithm were compared with the three methods proposed by Almoustafa et al. (2013), namely a flow-based ADVRP formulation solved using CPLEX (CPLEX), a single-start branch-and-bound with random selection (RND), and a multi-start branch-and-bound (MSBB).

Summary results are shown in Tables 6-8, where, mimicking the tables found in Almoustafa et al. (2013), row "#Inst" gives the number of attempted instances (after removing those instances for which both the RND and the MSBB could not find an optimal solution), row "#Feas" gives the number of feasible solutions obtained, row "#No Feas" gives the number of instances for which no feasible solution was found, and row "Avg. %Gap (all Feas)" gives the average percentage gap from the best solutions found, computed for those instances in which all algorithms found a feasible solution. Row "Avg. Time (all Feas)" gives the average time, expressed in seconds, computed for those instances in which all algorithms found a feasible solution. Rows "Avg. %Gap (Feas)" and "Avg. Time (Feas)" give, respectively, the average percentage gap from the best solutions found, computed for those instances in which the algorithm found a feasible solution, and the average time, expressed in seconds, computed for those instances in which the algorithm found a feasible solution. Additional row "%Feas" gives the percentage of instances for which the algorithm found a feasible solution.

Note that we can compare our results with those of Almoustafa et al. (2013) for all the three sets of problem instances because we used the exact same values computed by the authors for $D_{\max(2)}$ and $D_{\max(3)}$, which were not originally made available but were kindly provided to us by the authors. Therefore, as an additional contribution of our work and to encourage future research on the problem, we decided to republish in our repository the ADVRP benchmark instances accompanied by the values of $D_{\max(2)}$ and $D_{\max(3)}$.

Table 6. Computational results on 92 ADVRP instances with $D_{\max(1)} = +\infty$

	72 small ADVRP instances				20 large ADVRP instances				All 92 ADVRP instances			
	CPLEX	RND	MSBB	ILS	CPLEX	RND	MSBB	ILS	CPLEX	RND	MSBB	ILS
#Inst	72	72	72	72	20	20	20	20	92	92	92	92
#Feas	72	72	72	72	2	20	20	20	74	92	92	92
#No Feas	0	0	0	0	18	0	0	0	18	0	0	0
Avg. %Gap (all Feas)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
Avg. Time (all Feas)	304.12	0.01	0.01	1.21	1883.16	0.03	0.02	4.06	346.79	0.01	0.01	1.29
Avg. %Gap (Feas)	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.10	0.00	0.00	0.00	0.02
Avg. Time (Feas)	304.12	0.01	0.01	1.21	1883.16	0.34	0.38	7.95	346.79	0.08	0.09	2.67
%Feas	100.00	100.00	100.00	100.00	10.00	100.00	100.00	100.00	80.43	100.00	100.00	100.00

CPLEX, RND, and MSBB were run on a PC equipped with an Intel Core 2 CPU processor @ 2.40 GHz, while the ILS was run on a PC equipped with an Intel Core i7 CPU processor @ 2.70 GHz.

Table 7. Computational results on 92 ADVRP instances with $D_{\max(2)} = 0.90LT(1)$

	72 small ADVRP instances				20 large ADVRP instances				All 92 ADVRP instances			
	CPLEX	RND	MSBB	ILS	CPLEX	RND	MSBB	ILS	CPLEX	RND	MSBB	ILS
#Inst	72	72	72	72	20	20	20	20	92	92	92	92
#Feas	70	68	69	70	4	20	20	20	74	88	89	90
#No Feas	2	4	3	2	16	0	0	0	18	4	3	2
Avg. %Gap (all Feas)	0.00	0.02	0.01	0.00	2.36	0.00	0.00	0.00	0.13	0.02	0.01	0.00
Avg. Time (all Feas)	885.01	19.24	62.18	0.99	8567.97	0.78	0.77	3.45	1311.84	18.22	58.76	1.13
Avg. %Gap (Feas)	0.00	0.02	0.02	0.00	2.36	0.01	0.01	0.06	0.13	0.01	0.02	0.01
Avg. Time (Feas)	585.32	14.58	63.49	0.97	8567.97	6.22	6.09	6.22	1016.82	12.68	50.59	2.13
%Feas	97.22	94.44	95.83	97.22	20.00	100.00	100.00	100.00	80.43	95.65	96.74	97.83

CPLEX, RND, and MSBB were run on a PC equipped with an Intel Core 2 CPU processor @ 2.40 GHz, while the ILS was run on a PC equipped with an Intel Core i7 CPU processor @ 2.70 GHz.

Remarkably, the ILS proved to be effective in finding feasible solutions for the ADVRP. Indeed, among the three problems, the percentage of instances for which the algorithm found a feasible solution varied between 97.83% and 100.00%, thus showing a good consistency and outperforming the other methods. When solving the first problem, the average percentage gap from the best solutions found was very small, while the average computing times were slightly higher than those of the RND and the MSBB but much smaller than those of CPLEX. The better performance of the RND and the MSBB in this problem is motivated by the fast polynomial *Hungarian* method on which they are based, that is used as a lower bounding procedure to solve the linear relaxation of the TSP obtained from the original flow-based ADVRP formulation. The results show that this method is particularly effective in solving problems in which the maximum travelling distance is unconstrained. However, when solving the second and third problems (i.e., in which the maximum travelling distance is constrained and these constraints become increasingly tight), the average computing times of the RND and the MSBB considerably increase compared to the ILS. This can be imputed to the fact that the relaxation solved by the Hungarian method becomes more and more distant from the set of feasible solutions, and hence less effective in practice.

In terms of solution quality, on small-size instances (which are comparable in size to our randomly-created instances) the ILS always performs as well, if not better, than the RND and the MSBB, while on large-size instances (which are larger than our randomly-created instances) a small average percentage gap from the best solutions found still remains.

Overall, the ILS proved to be competitive with the RND and the MSBB in terms

Table 8. Computational results on 73 ADVRP instances with $D_{\max(3)} = 0.90LT(2)$

	53 small ADVRP instances				20 large ADVRP instances				All 73 ADVRP instances			
	CPLEX	RND	MSBB	ILS	CPLEX	RND	MSBB	ILS	CPLEX	RND	MSBB	ILS
#Inst	53	53	53	53	20	20	20	20	73	73	73	73
#Feas	53	50	50	53	1	19	19	19	54	69	69	72
#No Feas	0	3	3	0	19	1	1	1	19	4	4	1
Avg. %Gap (all Feas)	0.01	0.02	0.00	0.00	0.00	0.00	0.00	0.00	0.01	0.01	0.00	0.00
Avg. Time (all Feas)	1177.78	16.72	65.37	0.65	4032.28	4.67	4.66	2.24	1233.75	16.48	64.18	0.68
Avg. %Gap (Feas)	0.01	0.02	0.00	0.00	0.00	0.00	0.00	0.11	0.01	0.01	0.00	0.03
Avg. Time (Feas)	1367.35	16.72	65.37	0.64	4032.28	87.30	86.69	4.51	1416.71	36.15	71.24	1.66
%Feas	100.00	94.34	94.34	100.00	5.00	95.00	95.00	95.00	73.97	94.52	94.52	98.63

CPLEX, RND, and MSBB were run on a PC equipped with an Intel Core 2 CPU processor @ 2.40 GHz, while the ILS was run on a PC equipped with an Intel Core i7 CPU processor @ 2.70 GHz.

of solution quality and outperformed these two methods in terms of percentage of instances for which the algorithm found a feasible solution and solution time when the maximum travelling distance is constrained. This further motivates the choice of the algorithm and confirms its potential in solving more general problems like ours.

6.6. Results on Realistic Instances

The ILS was also tested on realistic instances derived from the case study that motivated our research, namely the daily inspection of the water distribution network of the city of Mashhad (Iran), which consists of 3,124 households/shops (regular nodes), 293 reservoirs/tanks (regular nodes), 14 treatment plants (regular nodes), 356 wells (special nodes) and 8 key centres (equipment centres).

We first collected ten real-world instances and the corresponding solutions implemented by the company and compared them with those obtained by the ILS. Each instance represents a working day involving five technicians, each of whom inspects around 10 or 11 nodes daily (so that the entire network is inspected around every two years). For each instance, the company provided the exact locations of the nodes visited, the service times and the sequence of visits. The real distances were computed with ArcGIS and the maximum duration L was set to 8 hours. The comparison between the results obtained by the company and those obtained by the ILS is reported in Table 9, where columns "z_{best}", "z_{avg}", "z_{worst}", "σ_z" and "Sec." give the best, average and worst solution values, the standard deviation and the computational time of the ILS, respectively. Column "z_{real}" indicates the real solution value, while column "%gap" gives the percentage gap between z_{real} and z_{avg}, computed as $100(z_{real} - z_{avg})/z_{real}$. In this case, no comparison with the flow-based model could be made due to the memory limitations encountered in solving medium-size instances using CPLEX.

The results show that the ILS can effectively solve real-world instances, improving the solutions found by the company by 12.85% on average. A simple illustrative comparison of two solutions, one implemented by the company and the other obtained by the ILS, is depicted in Figure 3. The two solutions comprise the five routes of Day 2. We can notice that the solutions implemented by the company are more balanced than those obtained by the ILS, as the five technicians visit a similar number of nodes. However, the ILS performs better in minimising the total route duration (578.01 vs 634.05).

With the aim of testing the ILS on larger-size realistic instances, we also generated

Table 9. Computational results on real-world instances

					ILS					Company	
$\|V_1 \cup V_2\|$	$\|V_2\|$	$\|V_3\|$	$\|K\|$	Day	z_{best}	z_{avg}	z_{worst}	σ_z	Sec.	z_{real}	%gap
54	6	2	5	1	611.40	611.40	611.40	0.00	1.63	687.42	11.06
53	7	1	5	2	578.01	578.01	578.01	0.00	2.07	634.05	8.84
55	6	2	5	3	448.88	448.88	448.88	0.00	2.27	544.81	17.61
54	6	2	5	4	489.93	489.93	489.93	0.00	2.35	540.97	9.43
54	8	4	5	5	541.30	541.30	541.30	0.00	2.68	594.19	8.90
54	6	3	5	6	416.19	416.19	416.19	0.00	2.19	553.13	24.76
53	6	2	5	7	494.49	494.49	494.49	0.00	1.99	540.67	8.54
55	5	2	5	8	486.00	486.00	486.00	0.00	2.46	530.30	8.35
55	7	3	5	9	417.59	417.59	417.59	0.00	1.89	539.88	22.65
54	5	2	5	10	524.86	524.86	524.86	0.00	2.00	572.65	8.35
overall avg					500.87	500.87	500.87	0.00	2.15	573.81	12.85

48 new instances starting from the water distribution network of Mashhad. In detail, we generated 48 realistic 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: 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 $|K| \in \{4,5\}$.

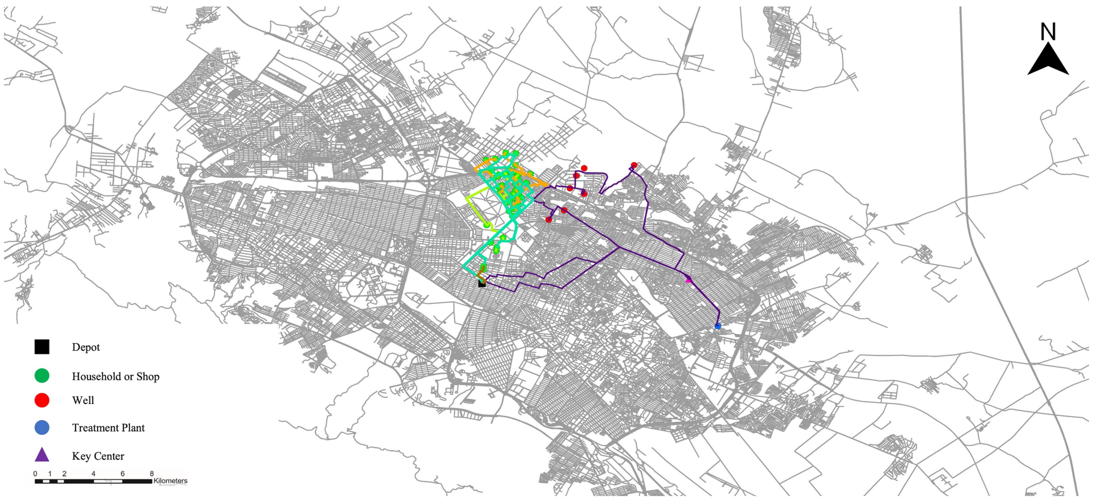
For each instance, the geographical coordinates of the nodes were randomly selected among the given exact locations and the real distances were computed using ArcGIS. Average service times were considered for the inspection of the nodes. The results are reported in Table 10. We recall that columns "z_{best}", "z_{avg}", "z_{worst}", "σ_z" and "Sec." give the best, average and worst solution values, the standard deviation and the computational time of the ILS, respectively.

The ILS achieved very robust results on instances having $|V_1 \cup V_2| = 60$, for which the standard deviation is constantly null 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. The results show that the ILS scales very well and can efficiently solve instances that are four times larger than those commonly faced in the case study.

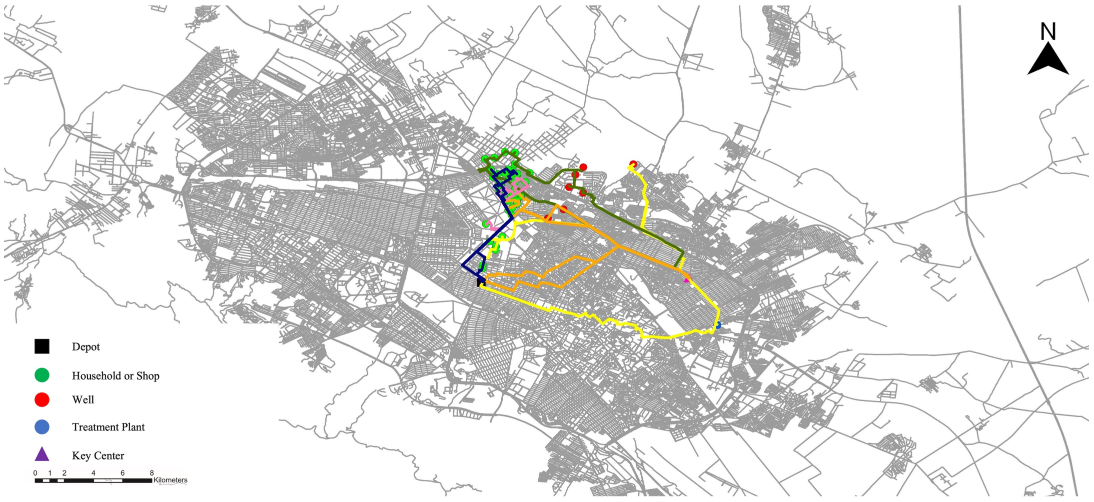
7. Conclusions

In this paper, we introduced a generalisation 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 must be performed to detect potential sources of contamination. To solve the VRPWDN, a flow-based model was proposed, and an Iterated Local Search (ILS) algorithm was developed.

Extensive computational tests on randomly generated small-size instances were performed to evaluate the performance of the flow-based model. 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. The ILS also proved to be effective in finding feasible solutions for the Asymmet-



(a) Day 2 - ILS



(b) Day 2 - Company

Figure 3. An illustrative comparison of VRPWDN solutions in Mashhad (Iran)

ric Distance-Constrained VRP, for which we used the benchmark instances proposed by Almoustafa et al. (2013). Additional computational tests were executed on ten real-world instances, comparing the solutions obtained by the ILS with those implemented by the company, and on larger-size realistic instances derived from the distribution network of Mashhad (Iran), proving that our methods can be applied with profit in a practical case and on a larger scale.

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 generalisation of the VRPWDN to the case of multiple periods. In this generalisation, one should first of all determine in which day inspecting the given nodes, and then creating the routes for each day. We are also interested in finding other related real-world applications and comparing the results of the proposed algorithms with the real-world solutions.

Table 10. Computational results on realistic medium- and large-size instances (three instances per line)

				ILS				
$\lvert V_1 \cup V_2 \rvert$	$\lvert V_2 \rvert$	$\lvert V_3 \rvert$	$\lvert K \rvert$	z_{best}	z_{avg}	z_{worst}	σ_z	Sec.
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				427942.58	427942.86	427943.57	0.43	7.54

Acknowledgements

The authors thank two anonymous reviewers whose comments greatly improved the quality of the work. Thanks also to Samira Almoustafa and Said Hanafi for sharing full details of the instances used in their paper.

Funding Details

This work was supported by the Research Deputy of Ferdowsi University of Mashhad, under Grant No. 48185, and the National Recovery and Resilience Plan (NRRP), Mission 04 Component 2 Investment 1.5–NextGenerationEU, Call for tender n. 3277 dated 30/12/2021, Award Number: 0001052 dated 23/06/2022. This support is gratefully acknowledged.

Disclosure Statement

The authors report there are no competing interests to declare.

Data Availability Statement

The data that support the findings of this study are openly available in “figshare” at `https://doi.org/10.6084/m9.figshare.20721424`.

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.

Almoustafa, S., Hanafi, S., and Mladenović, N. (2013). New exact method for large asymmetric distance-constrained vehicle routing problem. *European Journal of Operational Research*, 226(3):386–394.

Anaya-Arenas, A. M., Prodhon, C., Renaud, J., and Ruiz, A. (2021). An iterated local search for the biomedical sample transportation problem with multiple and interdependent pickups. *Journal of the Operational Research Society*, 72(2):367–382.

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*, 68:241–265.

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*, pages 161–191. SIAM, Philadelphia.

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.

Castillo-Salazar, J. A., Landa-Silva, D., and Qu, R. (2016). Workforce scheduling and routing problems: literature survey and computational study. *Annals of Operations Research*, 239:39–67.

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*, pages 193–212. SIAM, Philadelphia.

Elshaer, R. and Awad, H. (2020). A taxonomic review of metaheuristic algorithms for solving the vehicle routing problem and its variants. *Computers & Industrial Engineering*, 140:106242.

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.

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. Budapest, Hungary: 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.

Laporte, G., Nobert, Y., and Taillefer, S. (1987). A branch-and-bound algorithm for the asymmetrical distance-constrained vehicle routing problem. *Mathematical Modelling*, 9(12):857–868.

Li, J., Pardalos, P. M., Sun, H., Pei, J., and Zhang, Y. (2015). Iterated local search embedded adaptive neighborhood selection approach for the multi-depot vehicle routing problem with simultaneous deliveries and pickups. *Expert Systems with Applications*, 42(7):3551–3561.

Lin, S. (1965). Computer Solutions of the Traveling Salesman Problem. *Bell System Technical Journal*, 44(10):2245–2269.

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*, pages 129–168. Springer, Cham, 3rd edition.

Mathlouthi, I., Gendreau, M., and Potvin, J.-Y. (2018). Mixed integer linear programming for a multi-attribute technician routing and scheduling problem. *INFOR: Information Systems and Operational Research*, 56(1):33–49.

Mathlouthi, I., Gendreau, M., and Potvin, J.-Y. (2021). A metaheuristic based on tabu search for solving a technician routing and scheduling problem. *Computers & Operations Research*, 125:105079.

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*, 314:255–275.

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

Polnik, M., Riccardi, A., and Akartunalı, K. (2021). A multistage optimisation algorithm for the large vehicle routing problem with time windows and synchronised visits. *Journal of the Operational Research Society*, 72(11):2396–2411.

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.

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.

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.

Subramanian, A., Drummond, L. M., Bentes, C., Ochi, L. S., and Farias, R. (2010). A parallel heuristic for the Vehicle Routing Problem with Simultaneous Pickup and Delivery. *Computers & Operations Research*, 37(11):1899–1911.

Subramanian, A., Uchoa, E., and Ochi, L. S. (2013). A hybrid algorithm for a class of vehicle routing problems. *Computers & Operations Research*, 40(10):2519–2531.

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, Philadelphia, 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. Available at `https://www.who.int/news-room/fact-sheets/detail/drinking-water`. [Online; accessed 21 November 2023].

Zamorano, E. and Stolletz, R. (2017). Branch-and-price approaches for the multiperiod technician routing and scheduling problem. *European Journal of Operational Research*, 257(1):55–68.