Noname manuscript No.
(will be inserted by the editor)

Exact and hyper-heuristic solutions for the distribution-installation problem from the VeRoLog 2019 challenge

Ahmed Kheiri · Leena Ahmed · Burak Boyacı · Joaquim Gromicho · Christine Mumford · Ender Özcan · Ali Selim Dirikoç

Received: date / Accepted: date

Abstract This work tackles a rich VRP problem integrating a Capacitated Vehicle Routing Problem with Time Windows (CVRPTW), and a Service Technician Routing and Scheduling Problem (STRSP) for delivering various equipment based on customers' requests, and the subsequent installation by a number of technicians. The main objective is to reduce the overall costs of hired resources, and the total transportation costs of trucks/technicians. The problem was the topic of the fourth edition of the VeRoLog Solver Challenge in cooperation with the ORTEC company. Our contribution to research is the development of a mathematical model for this problem and a novel hyper-heuristic algorithm to solve the problem based on a population of solutions. Experimental results on two datasets of small and real-world size revealed the success of the hyper-heuristic approach in nding optimal solutions in a shorter computational time, when compared to our exact model. The results of the large size dataset were also compared to the results of the eight nalists

KTP Scheme, Grant/Award Number: KTP11692

A. Kheiri, B. Boyacı
Centre for Transport and Logistics, Department of Management Science, Lancaster University, Lancaster, LA1 4YX, UK E-mail: fa.kheiri, b.boyacig@lancaster.ac.uk

L. Ahmed, C. Mumford
School of Computer Science, Cardiff University, Queen's Buildings, 5 The Parade, Roath, Cardiff, CF24 3AA, UK E-mail: fahmedlh, mumfordclg@cardiff.ac.uk

J. Gromicho
VU Amsterdam, De Boelelaan 1105, 1081 HV Amsterdam, The Netherlands & ORTEC, Houtsingel 5, 2719 EA Zoetermeer, The Netherlands E-mail: joaquim.gromicho@ortec.com

E. Özcan
School of Computer Science, University of Nottingham, Jubilee Campus, Wollaton Road, Nottingham, NG8 1BB, UK E-mail: ender.ozcan@nottingham.ac.uk

A.S. Dirikoç
Department of Management Science, Lancaster University E-mail: a.dirikoc@lancaster.ac.uk

in the competition and were found to be competitive, proving the potential of our developed hyper-heuristic framework.

Keywords Transportation · Optimisation · Routing · Metaheuristics · Hyper-heuristics · VRP

1 Introduction

VeRoLog, the Euro Working Group on Vehicle Routing and Logistics optimisation, has been organising challenges for the routing community, where each challenge aims to promote the design and development of an applicable and e ective algorithm for a particular Vehicle Routing Problem (VRP). The organisation of the challenges is a collaborative e ort with companies, such as PTV, the leading German company in developing software solutions and consultations in tra c, transportation, and logistics, who organised the rst and second editions in 2014 and 2015. The third edition in 2017 was organised by ORTEC, one of the largest providers of advanced planning and optimisation solutions and services. They provided a real-world VRP problem involving the pickup and delivery of tools to measure milk quality at a number of farms, for a cattle improvement company.

This paper addresses the problem introduced in the VeRoLog Solver Challenge 2019 [1], which was organised by VeRoLog in cooperation with ORTEC. The challenge presented a new and exciting variant of a Vehicle Routing Problem (VRP), based on a real-world problem of one of ORTEC's clients. This problem is about the delivery of equipment, such as vending machines, to satisfy customers' requests, and the scheduling of a number of technicians each with a certain set of skills, who are required to install the equipment at least a day after the delivery. More formally, the overall problem is an integrated version of the Capacitated Vehicle Routing Problem with Time Windows (CVRPTW) (Laporte, 2009), and the Service Technician Routing and Scheduling Problem (STRSP) (Cordeau et al., 2010). The problem is highly complex, combining two interacting and co-dependent NP-hard routing problems into a single model, each problem having its own set of constraints, making it a unique and challenging topic for VRP researchers and practitioners. The key issue in this challenge is to nd e cient and low cost routes for both the delivery and installation, while scheduling the appropriate technicians according to their skills and working days for a given instance. The main objective is related to the total costs, aiming to reduce the total number of dispatched trucks and hired technicians on a single day, and within the planning horizon, in addition to minimising the penalties incurred due to equipment awaiting installation. Many companies operating in the delivery and equipment installation businesses would bene t signi cantly from an e ective and e cient solution to the proposed problem.

[1] https://verolog2019.ortec.com/ - last access: February 28, 2020

For such real-world complex problems, exact approaches, such as, mathematical programming often fail to provide solutions as the size of the instances get larger, and so heuristic-based search methods are preferred. As we have investigated di erent solution methods for the integrated problem from the VeRoLog Solver Challenge, we had a similar observation. Hence, in this study, we provide a mathematical model for the problem and report the results on a number of small instances. Additionally, we also present a population-based hyper-heuristic approach for the large scale problem instances. The proposed solution methods indeed provide competitive results with respect to the highest ranked scores for each challenge instance.

The remainder of the paper is organised as follows: Section 2 reviews the previous literature on the VRP, focusing on the CVRPTW and the technician routing problem. Section 3 provides a description of the tackled problem and Section 4 formulates the mathematical model, de ning the objectives and the problem constraints. Section 5 describes the applied hyper-heuristic framework. Finally Sections 6 and 7 present the results and conclusion, respectively.

2 Related Work

The problem we deal with from the VeRoLog Solver Challenge 2019 is a unique Vehicle Routing Problem (VRP), that combines a pickup and delivery problem with time windows with the scheduling of technicians to install the delivered equipment. However, it has much in common with some well-known variants of the VRP, thus in the following subsection we provide a brief overview of relevant previous studies. Following this, we cover selection hyper-heuristics in relation to this study.

2.1 An Overview of History and Variants of VRP

The VRP (Vehicle Routing Problem) is a generic term for a class of combinatorial problems that are concerned with the design of e cient routes for a number of vehicles serving a set of customers' requests, originating and ending at a depot location. A solution to a VRP instance is a tour for every vehicle, such that all customers' locations are visited, and each vehicle nishes its tour at a depot. An optimal solution is the one in which the total distance of all tours is minimised along with the associated costs. The term VRP was rst introduced in (Dantzig and Ramser, 1959) as a truck dispatching problem, where they modelled the problem of how a homogeneous eet of vehicles can serve the oil demand of a number of gas stations from a central hub, with a minimum travelling distance. This method was then generalised in (Clarke and Wright, 1964) to a linear optimisation problem: how to serve a number of customers located around a central depot, using a eet of vehicles with varying capacities. This has been known since then as the VRP problem which is one of the most researched topics in the eld of operations research and logistics.

Over the past decades, the VRP has grown more popular in the literature, and other variations have been developed to model real-life scenarios.

The Capacitated VRP (CVRP), which imposes a capacity constraint on the size of vehicles, is considered one of the elementary variants of VRP from which other variants originated. Amongst exact mathematical approaches to the CVRP, branch-and-cut (Augerat et al., 1995; Lysgaard et al., 2004) and algorithms based on the set partitioning formulation (Balinski and Quandt, 1964; Fukasawa et al., 2006) are the most popular approaches. Many studies also applied heuristic methods (Fisher and Jaikumar, 1981) and genetic algorithms (GAs) have also been widely used, usually by combining them with local search techniques (Prins, 2004; Mester and Braysy, 2007; Nagata, 2007; Nagata and Braysy, 2009).

A generalisation of the CVRP is the CVRP with Time Windows, which imposes a time interval (\time window") on the delivery of each customer's request. Exact methods for the CVRPTW have been successful for cases with up to 100 customers (Kolen et al., 1987), and as a result heuristic and metaheuristic methods have been preferred for solving instances of larger scale. Examples of heuristic methods applied to the CVRPTW problem can be found in (Solomon, 1987; Potvin and Rousseau, 1993; Russell, 1995; Sontrop et al., 2005), and other studies that applied metaheuristic methods such as genetic algorithms, ant colony optimisation, tabu search, and simulated annealing are in (Belhaiza et al., 2014; Cheng and Wang, 2009; Ding et al., 2012; Tavakkoli-Moghaddam et al., 2011).

In our proposed problem, customers' deliveries are scheduled within a planning period, with each customer requiring one or more visits during this period, similar to the Multi-Period VRP Problem (MPVRP). In contrast to the MPVRP, where service days are known and the frequency of customers visits is predetermined, the visits in our case are scheduled within prede ned time windows. The MPVRP is a well-studied variant in the literature of VRP. In a paper by Rahimi-Vahed et al. (2013), a path relinking algorithm is applied to the multi-depot periodic vehicle routing problem. This is done by generating a reference set of elite solutions, and combining characteristics from those solutions to nd even better solutions. The computational results show that this method produces good results in both run-time and solution quality. Archetti et al. (2015) present three ways to formulate the multi-period vehicle routing problem with time windows, then they solve the problem using a branch-and-cut algorithm. The algorithm was able to nd good solutions for small problems with 10 customer orders, but was unable to nd many good solutions for larger problems. Alonso et al. (2008) present a tabu search algorithm for the periodic vehicle routing problem with multiple trips and accessibility restrictions such that not every vehicle can visit every customer. When tested on randomly generated test cases, it performed reasonably well with regards to solution quality. Furthermore the computation time was manageable for instances up to 1000 orders. We refer the reader to (Campbell and Wilson, 2014) for more literature on MPVRP.

The multi-compartment VRP, and the multi-commodity VRP concentrate on delivering di erent types of commodities to the customer either using a single vehicle, or by splitting them to several vehicles, thus requiring multiple visits to the same customer. Mirzaei and W hlk (2017) conducted research on two variants of the MCVRP, one concentrates on split deliveries for di erent commodities, and the second focuses on delivering all commodities by a single vehicle. They proposed a branch-and-price method and compared the optimal costs of the two variants. The computational results were presented for instances with up to 100 customers, and the algorithm optimally solved instances with up to 50 customers and four commodities. Heuristic examples include (Cattaruzza et al., 2014) who proposed an iterated local search algorithm for solving the multi-commodity multi-trip VRP with the objective of minimising the number of used vehicles. In (Gu et al., 2019) the authors addressed the commodity constrained split delivery VRP, where multiple commodities can be mixed in a single vehicle while satisfying the capacity constraint, and similar to our problem, each customer can be visited more than once, and each visit should deliver only one commodity type. They proposed a heuristic based on adaptive large neighbourhood search (ALNS) and tested their approach on benchmark instances. Among the metaheuristic methods applied to the MCVRP, genetic algorithms are the most common so far. One example can be found in (Zhang and Chen, 2014), which describes a VRP encountered in frozen food delivery. Similar to our model, they associated a penalty cost for late delivery based on the types of products. A GA was proposed for solving the model on instances with real data.

Our problem involves loading-unloading operations from the depot to a customer location, similar to the classical VRP. The di erence in our case is that a single customer demand can be split into several requests, if the customer requires more than one machine type, thus several visits by di erent vehicles may be required. This draws a similarity to the class of VRP problems with Split Deliveries (SDVRP) (Dror and Trudeau, 1989). In the SDVRP, the constraint that each customer is visited by only one vehicle is relaxed, and thus customers' demand can be split between several vehicles for delivery. The rst heuristic approaches to solve the SDVRP were introduced by Dror and Trudeau (1989, 1990). After these studies, most of the subsequent work focused on metaheuristic or hybrid schemes. One example is the work of Archetti and Speranza (2012) who applied a tabu search algorithm, and Boudia et al. (2007) used a genetic algorithm combined with a local search procedure. Hybrid algorithms have since grown in popularity. Examples are found in (Archetti et al., 2008; Cheng and Wang, 2009). Many exact models have also been proposed for this problem and one example is the study in (Archetti et al., 2011). For further literature on SDVRP we refer to (Archetti and Speranza, 2012).

Recently, attention has been paid to a class of VRPs that model multiple vehicle trips under the name \Multiple Trips Vehicle Routing Problem" (MTVRP). A clear improvement can be obtained by allowing a single vehicle to perform multiple trips, especially when the vehicle capacity is limited and

the replenishment of stock is required. The problem assumes that trucks can visit the depot more than once in the time horizon of the problem. Example of studies that modelled multiple depot visits are (Tsirimpas et al., 2008; Tatarakis and Minis, 2009). A survey paper on the literature of MTVRP for further details can be found in (Cattaruzza et al., 2014).

Finally, we mention the Workforce Routing and Scheduling Problem (WSRP), which is the focus of the second part of our model that involves the routing of the technicians for the installation of machines at the delivery points. There are some scenarios where personnel are required to perform tasks in certain locations, and hence require some sort of transportation to these locations. These scenarios are known as the Workforce Routing and Scheduling Problem (WRSP), which has become a widespread term used by many service providers.

A similar problem to the WRSP, is the Service Technician Routing and Scheduling Problem (STRSP), which involves designing the least cost routes for vehicles carrying a number of service technicians. Each task demands the allocation of a technician with the required skills set, and this may also be associated with a time window. One of the pioneering publications in this field is the work of Cordeau et al. (2010) who solved a real life technician scheduling problem for a large telecommunication company set as a competition by the French Operational Research Society in 2007. In this paper an adaptive large neighbourhood search algorithm is implemented. In (Xu and Chiu, 2001), the authors concentrated on a field technician scheduling problem in the telecommunications industry, and their purpose was to maximise the number of served requests as well as considering the request's priority and the technician's skill level. A local search algorithm, a Greedy Randomised Adaptive Search Procedure (GRASP) and a greedy heuristic algorithm were proposed to solve the problem. Kovacs et al. (2012) studied the service technician routing and scheduling problem with the objective of minimising the total routing and outsourcing costs. The authors used an adaptive large neighbourhood search algorithm for solving the problem on artificial and real-world instances. Pillac et al. (2013) proposed a parallel matheuristic approach for solving a variant of the TRSP in which a number of technicians with a set of accompanying skills, tools and spare parts need to be scheduled and routed within given time windows. The study dealt with the availability of tools and spare parts for the technicians and routing them to the depot for the replenishment of tools. Xie et al. (2017) used an iterated local search algorithm to solve the TRSP. They studied a variant where it was given which technicians can serve which orders. The algorithm was benchmarked on instances ranging from 25 to 100 orders and compared to an ALNS algorithm, where it was found that it performs significantly better on large instances with fast computational times.

In addition to technician routing, similar problems can also be found in other fields where scheduling is important, such as the home health care (Bertels and Fahle, 2006), and the scheduling of security personnel (Misir et al., 2011).

The problem studied in this paper is a unique version considering its set of constraints and the integration with the sta rostering and routing problem. To the best of our knowledge there is no similar version investigated in the literature, and the best matching study to our problem description is by Bae and Moon (2016) where they extended the Multi-Depot Vehicle Routing Problem with Time Windows (MDVRPTW) to a problem of service vehicles used for delivery and installation of electronics. They developed a mixed integer programming model, a heuristic and a genetic algorithm, and compared their performances. There are di erences between this study and our problem, for example we consider a longer planning horizon, and deal with multiple types of machines. We also allow multiple visits to the customer by di erent vehicles, while their model only allows a single visit for both delivery and installation. They also assign a maximum period of time between delivery and installation that must not be exceeded, while in our model we restrict this time by imposing a penalty.

2.2 Selection Hyper-heuristics

Hyper-heuristics are general purpose search methodologies for solving di cult combinatorial problems. They operate at an abstract and higher level than heuristics, that is, over the low level heuristics space (Drake et al., 2020). One of the earliest hyper-heuristic frameworks proposed requires that hyper-heuristics should not use any speci c knowledge from the solution domain (Cowling et al., 2000), a feature which makes them applicable to problem instances with di erent characteristics or even di erent problems, without a need for further algorithmic or parametric adjustments. This feature forms a principle concept of hyper-heuristics in past and modern research. Hyper-heuristics are de ned as *"heuristics to choose heuristics"* in (Cowling et al., 2000), although the rst attempt to design hyper-heuristics dates to as early as 1963 (Fisher, 1963). Burke et al. (2019) identi es two categories based on the nature of the heuristic search space: *selection* and *generation* hyper-heuristics. In the former class, a heuristic is selected from an existing repository of heuristics to try to discover the behaviour of these heuristics in order to enable/disable some of them during the search process; while in the latter, new heuristics are built by discovering the characteristics of the input heuristics. The approach in this study uses the former type of hyper-heuristics, which are based on a single-point based search framework with two consecutive operations that work iteratively to improve a single initial solution through *heuristic selection* and *move acceptance*. With the existence of a de ned number of low level heuristics, the selection method chooses one of these heuristics and applies it to a solution in hand, generating a new solution. The move acceptance decides on the acceptance of the new solution based on the tness/objective evaluation. Heuristic selection can be carried out using simple methods such as random selection or by selecting from a pre-de ned ordering of the low level heuristics, or it can incorporate learning by de ning some probability measures

for the performance of low level heuristics. For the most recent advances and classi cation of selection hyper-heuristics we refer to (Drake et al., 2020).

There are di erent criteria to classify selection hyper-heuristics, and one of them is the solution nature, where selection hyper-heuristics are classi- ed based on this measure into single point or multiple point. Multiple point (population) selection hyper-heuristics utilise multiple current solutions during the search, while single point (single solution) hyper-heuristics are based on a single current solution that is iteratively improved during the search. The majority of the previous studies on selection hyper-heuristics present approaches based on single-point-based search, and only a few used a population of solutions or a mixed approach alternating between using single and multiple solutions for the search. Moreover, those previously proposed population based approaches are mostly a hybrid between a selection hyper-heuristic and an evolutionary algorithm framework.

Cowling et al. (2002) investigated a genetic algorithm based on hyper-heuristics for the personnel scheduling problem. A GA is implemented and applied as a high level selector, and a set of low level heuristics are used at each generation to locally improve the quality of each individual, where the low level heuristics are applied in any sequence. Sabar and Kendall (2015) proposed a Monte-Carlo tree search hyper-heuristic framework that tries to identify good sequences of heuristics using the Monte-Carlo search tree. A memory mechanism containing a population of solutions is utilised, and at each iteration a solution from the population is selected, and the population is subsequently updated using several updating rules. Lei et al. (2015) proposed a memetic algorithm based on hyper-heuristics to solve an examination timetabling problem. Their approach constructs several heuristic lists based on graph colouring heuristics and applies evolutionary operators to generate new lists. A local search method is used to further optimise the solutions. Hsiao et al. (2012) implemented a hyper-heuristic based on variable neighbourhood search (VNS) iterating in two stages, rst using a population of solutions, and the second stage uses only a single solution. Their approach consists of two main steps, $shaking$ and $local\ search$. The shaking phase improves the exploration of the search space, and the local search step looks for the local optima. A population of solutions is used in the shaking stage, where the authors argued that the diversity of solutions is important in the rst stages of the search to explore the right search path, and after a period of time the best solution is picked from the population. Tournament selection is used to lter un t solutions from the population. Lehrbaum and Musliu (2012) introduced a hyper-heuristic that alternates between working on a single solution and a population of solutions. Their algorithm starts by scoring the available local search heuristics, and a serial phase working with single solutions starts by applying the heuristics sequentially according to their quality scores. A parallel phase uses a population of solutions, and a heuristic is applied to each individual in the population. The algorithm switches back to the serial phase whenever a global improvement is found (i.e., better than the best found solution so far) .

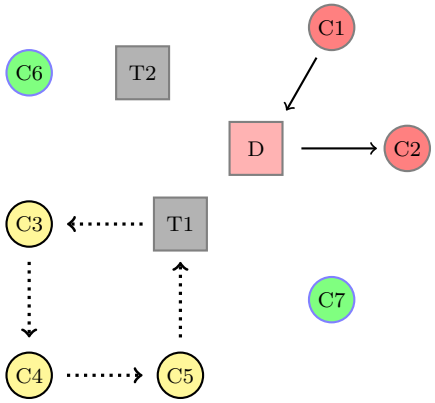
We also discuss here the successful application of hyper-heuristics in different variants of VRP problems, since the VRP is considered an active research field in hyper-heuristics. In (Pisinger and Ropke, 2007) the Adaptive Very Large Neighbourhood Search (AVLNS) hyper-heuristic was successful in finding the state-of-the-art results for many benchmark instances of several variants of VRP. Also the CVRP with time windows is one of HyFlex (hyper-heuristic flexible framework) problem domains, for which newly developed approaches in hyper-heuristics are tested (Walker et al., 2012). Other VRP problems solved using selection hyper-heuristics include the periodic VRP (Chen et al., 2016), dial a ride (Urra et al., 2015), urban transit routing (Ahmed et al., 2019), and inventory routing (Kheiri, 2020).

The previous VeRoLog challenge 2016-2017 tackled a rich VRP problem related to a cattle improvement company that regularly measures the milk quality at a number of farms using specialised tools. These tools have to be delivered to a number of farms (customers) on request and picked up again a few days after delivery. The key challenge is how to schedule the deliveries to satisfy the requests, whilst at the same time design efficient routes for the pick-ups and deliveries. The second place winner on this challenge used a hyper-heuristic approach based on an online selection method (Kheiri et al., 2019).

3 Description of the Problem

The real-world problem from VeRoLog Solver Challenge 2019 can formally be stated as follows. There are a number of customers $Cr = \{cr_1, cr_2 \dots cr_{jCrj}\}$ geographically spread at different locations, and a depot located at H_0. The distance between any two locations i and j is given by $d_{i,j}$. The purpose is to respond to customers' requests by delivering machines and getting them installed by a technician within a defined time horizon T given as a consecutive number of days.

An unlimited number of identical trucks (i.e. vehicles) $K = \{k_1, k_2, \dots\}$ can be hired to transport the machines to the customers. They are located at the depot each with a maximum capacity C. Also, a number of machines $M = \{m_1, m_2 \dots m_{jMj}\}$ are available to be delivered to the customers at their request, and there are different types of machines with different sizes expressed in the same size unit as the truck capacity. The machines are all located in the depot, with enough machines to satisfy all the demand. A set of customer requests $R = \{r_1, r_2 \dots r_{jRj}\}$ should be satisfied. The requests are known at the start of the planning period. A single request $r_i = \{cr_i, w_i, m_i, n_i\}$ asks for one machine type $m_i \in M$, of quantity n_i, for exactly one customer cr_i, and w_i is the associated time window to deliver the request, where w_i is specified by the earliest day e_i and the latest day l_i to deliver request r_i. A request of the same type of machines cannot be split and should be delivered by the same truck, and if a customer requires another machine type, a separate request is made. Each truck journey on a day should start and end at the depot location H_0


C1
C6
T2
D
C2
C3
T1
C7
C4
C5

nician home location, maximum travel distance per day, maximum number of installations per day, and which machines they have the skill to install.

The last point to mention, is that the technician should install a delivered machine as soon as possible after the delivery, and for each delayed installation of request r_i, a penalty Cl_i is added to the cost, and each machine type has a di erent penalty value.

The main objective is to reduce the overall costs associated with trucks, technicians and idle machines costs. The trucks/technicians total cost is constituted of the following parts: the cost of hiring a truck/technician per day, the cost of hiring a truck/technician within the planning horizon T, and the cost per unit of distance for the travelling of truck/technician. The distance between coordinates (x_1, y_1) and (x_2, y_2) is de ned as the ceiling of the Euclidean distance, $\lceil \sqrt{(x_1 - x_2)^2 + (y_1 - y_2)^2} \rceil$. In addition, there is the cost for penalising idle machines that remain without installation for more than one day. This penalty cost is dependant on which machine it is and the number of days it was idle. The objective function is described by Equation 1 in the following section.

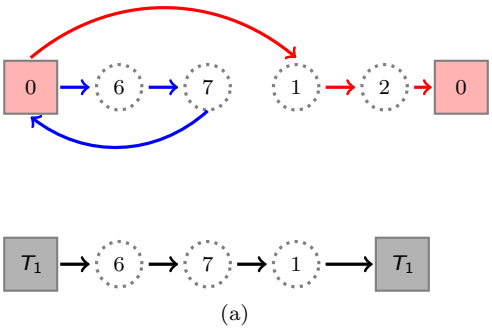
A solution gives, for each day in the planning horizon, the routes followed by each truck/technician. Assuming that $\{1, 6, 7, 0, 1, 2\}$ is the route of a single truck in one of the planning days. The rst element in the route `1' is the truck ID, followed by the requests ID's that this truck served. The ID `0' refers to the depot, and it means that truck 1 visited the depot after serving requests `6' and `7' and was loaded to serve requests `1' and `2'. Each series of requests before the truck goes back to the depot is named \tour". In this route, there are two tours given as $\{6, 7\}$ and $\{1, 2\}$. The start and end of the truck journey at the depot is not explicitly written in this route format.

The technician routes are very similar, starting with the technician ID, followed by the ID's of the requests that this technician served. Also, the start and end of the technician journey at their home location is not explicitly mentioned in the solution.

3.1 Problem Instances

We have used two datasets of instances, one of them has been developed for this work and one, referred to as the $hidden$ dataset, was used in the competition to evaluate the participants' algorithms in the VeRoLog Solver Challenge. Each of the instances provides di erent types of information such as the weights of the objective function components, the maximum truck capacity, the number of days in the planning horizon, and the maximum travel distance allowed by each truck. The details of the requests, locations given as x, y coordinates (i.e. depot, technicians homes, customers) and technicians are also given. The characteristics of these datasets are provided in Table 1.

The $small$ dataset, which includes instances of sizes varying between 6 to 16 requests, is developed speci cally for this work. The reason for generating this dataset is to provide an ideal size of instances for testing the mathematical



(a)

Day = 1
Number of Trucks = 2
1 6 7 0 1 2
2 3 4 5
Number of Technicians = 0

Day = 2
Number of Trucks = 1
1 8 10 0 9
Number of Technicians = 1
1 6 7 1

(b)

Fig. 2: Solution example, and the routes of truck 1 in day 1, and technician 1 in day 2. The red and blue arcs represent two di erent tours. The pink and grey coloured boxes represent the depot and technician T_i respectively

model which can only be applied on instances of such sizes. It is also essential to test our developed hyper-heuristic approach on instances with di erent characteristics and scales and to compare its performance to the exact model by its ability of nding optimal solutions in a short duration of time.

The hidden dataset was used to assess the performance of the competitors algorithms and rank the nalists in the restricted resource challenge[2]. This dataset contains instances of large sizes up to 900 requests. The number of di erent types of machines vary between 3 and 7 in each instance, and the number of technicians range from 25 to 125. The highest variation can be found in the costs of using trucks and technicians, distance costs, and the costs per day that trucks and technicians are used. These values range from 10 to 100,000. We refer the reader to (Gromicho et al., 2019) for a comprehensive description of the problem and the formal challenge rules[3].

[2] The solvers of the finalists were run on the hidden dataset for a limited computational times determined by the challenge rules

[3] A detailed description of the challenge and the datasets is also provided here: https://verolog2019.ortec.com/

Table 1: Characteristics of the small and hidden instances

Instance	Days	Truck Capacity	Truck Max Distance	Truck Distance Cost	Truck Day Cost	Truck Cost	Technician Distance Cost	Technician Day Cost	Technician Cost	Machines	Locations	Requests	Technicians
Small_01	4	18	1090	10000	1000	1000	10000	10000	10	3	5	6	5
Small_02	5	18	905	100	10000	10	1000	100	100	4	8	8	10
Small_03	6	18	2000	10000	100	100	100	1000	1000	5	10	10	15
Small_04	7	18	2000	10000	10	1000	10000	10000	10000	6	9	12	20
Small_05	4	18	2000	1000	10000	10000	100000	100	100000	7	10	14	25
Small_06	5	18	2000	10	1000	100000	10	1000	10	3	10	16	5
Small_07	6	18	2000	100000	100	10000	10000	100	100	4	6	6	10
Small_08	7	18	1045	1000	10	1000	100000	10	1000	5	8	8	15
Small_09	4	18	970	10	100	100000	1000	100	10000	6	8	10	20
Small_10	5	18	2000	100	100000	10	100000	10	100000	7	10	12	25
Small_11	6	18	1195	100000	1000	100000	100000	100000	10	3	9	14	5
Small_12	7	18	980	1000	10000	1000	10000	1000	100	4	9	16	10
Small_13	4	18	2000	100000	10000	10000	100	10	1000	5	4	6	15
Small_14	5	18	465	100	1000	100	10	10	10000	6	8	8	20
Small_15	6	18	620	10	1000	10	1000	100000	100000	7	8	10	25
Small_16	7	18	775	100000	100000	100	10	100000	10	3	7	12	5
Small_17	4	18	2000	10	10000	10	1000	100000	100	4	9	14	10
Small_18	5	18	1135	100000	100000	100000	100	1000	1000	5	13	16	15
Small_19	6	18	830	100	100000	100000	100	1000	10000	6	8	6	20
Small_20	7	18	2000	1000	100	10000	100	10000	100000	7	9	8	25
Small_21	4	18	2000	100	100000	10000	10	100000	10	3	8	10	5
Small_22	5	18	980	1000	10	100	1000	10000	100	4	11	12	10
Small_23	6	18	2000	10000	10	100	10	100	1000	5	10	14	15
Small_24	7	18	960	10000	10	1000	100000	10000	10000	6	12	16	20
Small_25	4	18	2000	10	100	10	10000	10	100000	7	10	6	25
Hidden_01	15	18	1620	1000	1000	100	100	10	10	3	54	150	25
Hidden_02	25	18	1350	10	100000	1000	100000	1000	100	4	112	300	50
Hidden_03	35	18	1060	10000	1000	100000	100	1000	1000	5	163	450	75
Hidden_04	45	18	1040	10	10000	100	10	10000	10000	6	217	600	100
Hidden_05	55	18	2000	1000	100000	10000	100000	100000	100000	7	270	750	125
Hidden_06	15	18	1205	100	10	10	10	10	10	3	306	900	25
Hidden_07	25	18	980	1000	100000	1000	10000	1000	100	4	59	150	50
Hidden_08	35	18	1030	10000	100	100000	1000	100	1000	5	116	300	75
Hidden_09	45	18	2000	1000	100	1000	100000	100	10000	6	167	450	100
Hidden_10	55	18	950	10	10000	100000	1000	10000	100000	7	220	600	125
Hidden_11	15	18	2000	10000	10000	10	100000	100	10	3	254	750	25
Hidden_12	25	18	1405	10000	100	10000	1000	100000	100	4	310	900	50
Hidden_13	35	18	2000	100000	100	1000	10000	100000	1000	5	68	150	75
Hidden_14	45	18	1430	10000	1000	100000	100	100000	10000	6	117	300	100
Hidden_15	55	18	1350	1000	100	10	10	10	100000	7	173	450	125
Hidden_16	15	18	1170	100	100000	100	1000	10000	10	3	205	600	25
Hidden_17	25	18	2000	100000	10000	100	10000	100	100	4	260	750	50
Hidden_18	35	18	1435	100	100000	1000	10	10	1000	5	313	900	75
Hidden_19	45	18	1010	100000	10	10	10000	100000	10000	6	60	150	100
Hidden_20	55	18	1205	10	10	100	10000	1000	100000	7	125	300	125
Hidden_21	15	18	1230	100	1000	10000	1000	100	10	3	154	450	25
Hidden_22	25	18	1500	10	10	10000	100	1000	100	4	206	600	50
Hidden_23	35	18	1100	100000	1000	100000	100	10000	1000	5	266	750	75
Hidden_24	45	18	1290	100000	10	10	10	10	10000	6	317	900	100
Hidden_25	55	18	1160	100	10000	10000	100000	10000	100000	7	68	150	125

4 Mathematical Formulation of the CVRP for Delivery and Installation of Machines

Sets and Indices

R: requests ($i, j \in R$)
R_0: requests and the depot ($R_0 = H_0 \cup R$)
K: vehicles ($k \in K, |K| = \max_t\{M_t\}$)
S: technicians ($s \in S$)
R_s: requests that technician s can install and the home location of technician s ($R_s = \{s, i : a_{si} = 1, \forall i \in R\} \cup H_s$)

Parameters

T: number of days in the entire planning horizon
H_0: location of the depot
D: maximum distance a vehicle can travel per day
M_t: upper bound on the number of visits a vehicle can do to depot on day t
d_{ij}: distance between request/depot/home i and j
e_i: earliest (rst) day that request i can be delivered
l_i: latest (last) day that request i can be delivered
C: vehicle capacity
c_i: capacity needed to deliver request i
a_{si}: 1, if technician s is eligible to satisfy request i; 0, otherwise
H_s: home location of technician s
D_s: maximum distance that technician s can travel per day
N_s: maximum number of installations that technician s can do per day
CI_i: cost of delaying the installation of request i per day
CV: cost of using a vehicle any day during the planning horizon
CT: cost of using a technician any day during the planning horizon
CVU: cost of using a vehicle per day
CTU: cost of using a technician per day
CVT: cost of travelling unit distance by a vehicle
CTT: cost of travelling unit distance by a technician

Decision Variables

x_{ijk}^t: 1, if vehicle k visits {request j}/depot right after {request i}/depot on day t; 0, otherwise
z_{ijs}^t: 1 if technician s visits {request j}/home right after {request i}/home on day t; 0, otherwise
m_k: 1 if vehicle k is used during the planning horizon; 0, otherwise
r_s: 1 if technician s is used during the planning horizon; 0, otherwise
v_k^t: 1 if vehicle k is used during day t; 0, otherwise
p_s^t: 1 if technician s is used during day t; 0, otherwise
w_i^t: 1 if request i is delivered on day t; 0, otherwise
y_i^t: 1 if request i is installed on day t; 0, otherwise

q_{ik}: upper bound on the weight of the machines on vehicle k right after leaving {request i}/depot
g_{is}: number of visits done by technician s before visiting {request i}/home
b_i: number of days installation of request j is delayed after its delivery

Mathematical Modelling

$$\min \overbrace{\sum_{k2K} \mathsf{CV} m_k + \sum_{t=1}^{T} \sum_{k2K} \mathsf{CVU} v_k^t + \sum_{t=1}^{T} \sum_{\substack{k2K,\\ i,j2R_0,i\neq j}} \mathsf{CVT} d_{ij} x_{ijk}^t}^{\text{vehicle cost}}$$
$$+ \underbrace{\sum_{s2S} \mathsf{CT} r_s + \sum_{t=1}^{T} \sum_{s2S} \mathsf{CTU} p_s^t + \sum_{t=1}^{T} \sum_{\substack{s2S,\\ i,j2R_s,i\neq j}} \mathsf{CTT} d_{ij} z_{ijs}^t}_{\text{technician cost}} + \underbrace{\sum_{i2R} \mathsf{CI}_i b_i}_{\text{idling cost}} \tag{1}$$

subject to

$$\sum_{j2R_0,i\neq j} x_{ijk}^t = \sum_{j2R_0,i\neq j} x_{jik}^t \qquad \forall i \in R_0, k \in K, t \in [1, T] \tag{2}$$

$$\sum_{i2R} x_{H_0ik}^t = \sum_{i2R} x_{iH_0k}^t \leq M_t v_k^t \qquad \forall k \in K, t \in [1, T] \tag{3}$$

$$\sum_{i,j2R_0,i\neq j} d_{ij} x_{ijk}^t \leq D \qquad \forall k \in K, t \in [1, T] \tag{4}$$

$$v_k^t \leq m_k \qquad \forall k \in K, t \in [1, T] \tag{5}$$

$$x_{ijk}^t \leq v_k^t \qquad \forall i, j \in R_0, k \in K, t \in [1, T] \tag{6}$$

$$\sum_{k2K,j2R_0,i\neq j} x_{ijk}^t = w_i^t \qquad \forall i \in R, t \in [e_i, l_i] \tag{7}$$

$$\sum_{t=e_i}^{l_i} w_i^t = 1 \qquad \forall i \in R \tag{8}$$

$$q_{jk} \leq q_{ik} - x_{ijk}^t (C + c_j) + C \qquad \forall i \in R_0, j \in R, i \neq j, k \in K, t \in [1, T] \tag{9}$$

$$q_{H_0k} = C \qquad \forall k \in K \tag{10}$$

$$\sum_{j2R_s,i\neq j} z_{ijs}^t = \sum_{j2R_s,i\neq j} z_{jis}^t \qquad \forall i \in R_s, s \in S, t \in [1, T] \tag{11}$$

$$\sum_{i2R} z_{H_sis}^t = \sum_{i2R} z_{iH_ss}^t = p_s^t \qquad \forall s \in S, t \in [1, T] \tag{12}$$

$$\sum_{i,j2R_s,i\neq j} d_{ij} z_{ijs}^t \leq D_s \qquad \forall s \in S, t \in [1, T] \tag{13}$$

$$\sum_{i2R_s,j2R,i\neq j} z_{ijs}^t \leq N_s \qquad \forall s \in S, t \in [1, T] \tag{14}$$

$$p_s^t \leq r_s \qquad \forall s \in S, t \in [1, T] \qquad (15)$$

$$z_{ijs}^t \leq p_s^t \qquad \forall i, j \in R_s, s \in S, t \in [1, T] \qquad (16)$$

$$\sum_{s2S, j2R_s, i6=j} z_{ijs}^t = y_i^t \qquad \forall i \in R, t \in [e_i + 1, T] \qquad (17)$$

$$\sum_{t=e_i+1}^{T} y_i^t = 1 \qquad \forall i \in R \qquad (18)$$

$$g_{js} \leq g_{is} - z_{ijs}^t(1 + N_s) + N_s \qquad \forall i \in R_s, j \in R, i \neq j, s \in S, t \in [1, T] \qquad (19)$$

$$g_{H_0 s} = N_s \qquad \forall s \in S \qquad (20)$$

$$\sum_{u=t}^{t+4} p_s^u \leq 5 - p_s^{t+5} \qquad \forall s \in S, t \in [1, T-5] \qquad (21)$$

$$\sum_{u=t}^{t+4} p_s^u \leq 5 - p_s^{t+6} \qquad \forall s \in S, t \in [1, T-6] \qquad (22)$$

$$\sum_{u=T\ \ 5}^{T\ \ 1} p_s^u \leq 5 - p_s^T \qquad \forall s \in S \qquad (23)$$

$$\sum_{t=e_i+1}^{T} ty_i^t - \sum_{t=e_i}^{l_i} tw_i^t - 1 = b_i \qquad \forall i \in R \qquad (24)$$

$$q_{ik} \in \mathcal{Z}_{\ 0};\ \ x_{ijk}^t, v_k^t, m_k \in \{0, 1\} \qquad \forall i, j \in R_0, k \in K, t \in [1, T], i \neq j \qquad (25)$$

$$g_{is} \in \mathcal{Z}_{\ 0};\ \ z_{ijs}^t, p_s^t, r_s \in \{0, 1\} \qquad \forall s \in S, i, j \in R_s, t \in [1, T], i \neq j \qquad (26)$$

$$b_i \in \mathcal{Z}_{\ 0};\ \ w_i^t, y_i^t \in \{0, 1\} \qquad \forall i \in R, t \in [1, T] \qquad (27)$$

The exact model for the given problem is formulated by Eqs. (1)-(27). The objective function (1) is composed of three types of costs: the vehicle (rst three summations), technician (next three summations), and idling cost (last summation). The sum of the vehicle hiring cost, the vehicle cost per day, and the vehicle cost per distance is equal to the total vehicle cost. Similarly the sum of the personnel hiring cost, the personnel cost per day, and the personnel cost per distance is equal to the personnel cost. The idling cost is calculated by multiplying the cost of idling all the machines at each request by the di erence between the delivery and installation days.

In the mathematical model, constraints (2)-(10) are vehicle, constraints (11)-(23) are technician, and constraints (24) are idle time related constraints. Constraints (25)-(27) de ne the domains of the variables.

Constraints (2) and (11) are balance equations for the vehicles and technicians respectively. They ensure that in any day, in any vehicle and personnel route, the number of arcs entering to a location of a request, depot or home should be equal to the number of arcs exiting from the same location.

Constraints (3) and (12) ensure that both the vehicles and technicians start/end their routes from/at the depot and home respectively. For the tech-

nicians, the problem dictates upper bounds for the number of installations (N_s) and travel distance (D_s) for any given day. Therefore, in an optimal solution, if a technician is used on any given day, they should leave and return their home only once. On the other hand, for the vehicles, in addition to the total travel distance on any given day (D), there is an upper bound on the weight of the machines that are being carried by the vehicle, de ned as vehicle capacity (C), at any given time of the day. This makes it possible for a given vehicle on any given day to deliver more than its capacity by making multiple visits to the depot. Because of the di erence of the restrictions on the vehicles and technicians, constraints (12) are satis ed with an equality to p_s^t whereas constraints (3) are satis ed with an inequality to v_k^t. The former constraints force technicians to leave and return to their home only once if they are working on day t, whereas the latter constraints force vehicles to have an equal number of trips that leave from and return to the depot, and these trips can only occur if the truck is operating on day t. M_t on the RHS of the constraints (3) is calculated by counting the number of orders that can receive a delivery on day t. Note that, the maximum value that M_t can take on di erent days also sets the upper bound for the maximum number of hired vehicles in the planning horizon.

Constraints (4) and (13) restrict the total travel distance for each day of the vehicles and the technicians, respectively. In addition, constraints (14) set the maximum number of installations for a technician on a single day. Similarly, in the problem de nition, there is a limit set on the total weight of machines a truck can carry. This is ensured by constraints (9) and (10). The variable q_{H_0k}, that represents the weight of the machines right after the vehicle is leaving the depot, is set to be C for any truck by constraints (10). Since constraints (9) ensure that the weight on the truck decreases at every request stop by the weight of the delivery of the same request, and the q_{ik}'s are de ned as non-negative integer variables, no truck can carry more than its capacity.

Constraints (5) and (15) ensure that in order to use a vehicle or a technician respectively in any day of the planning horizon, we need to hire them rst. Similarly, constraints (6) and (16) ensure that if a vehicle or a technician travel between two requests on any given day, they are already hired for the day.

Constraints (7) and (17) establish the relationship between the routing (x_{ijk}^t and z_{ijs}^t) and service, i.e. delivery (w_i^t) and installation (y_i^t) variables for the vehicles and technicians respectively. According to constraints (7), if the location of a request is visited by a vehicle, then the machines ordered by this request are delivered between the rst (e_i) and the last (l_i) days the delivery can be done. Similarly, according to constraints (17), if the location of a request is visited by a technician, then the installation that is ordered by this request is done at least one day after the earliest day that the request can be delivered.

Constraints (8) and (18) ensure that all the deliveries and installations are done within their prede ned times respectively. Constraints (8) ensure that each request is delivered by one of the vehicles. These constraints restrict each request to be delivered between their rst (e_i) and the last (l_i) days the delivery

can be made. Similarly, constraints (18) ensure that each request is installed by one of the technicians. Since the earliest installation day is one day after the delivery of the machines, the constraints only consider the days after the first day that the request can be delivered.

Constraints (9) and (10) prevent subtours in vehicle routes. Constraints (10) assign the maximum weight (C) a vehicle can carry when leaving the depot to variables q_{ik} on any given day. Constraints (9) subtract the weight of the machines that are being delivered from q_{ik} every time a vehicle visits a request. Keeping track of each q_{ik} and forcing q_{jk} less than or equal to q_{ik} if request j is visited immediately after request i or depot by vehicle k (i.e. $x^t_{ijk} = 1$) prevent the formation of subtours. Note that, since a vehicle can make multiple visits to the depot on any given day, these constraints are not being forced for the trips to the depot.

Constraints (19) and (20) prevent subtours in technician routes. Constraints (20) assign the maximum number of visits a technician can make (N_s) to variables g_{is} on any given day. Note that, since the maximum number of installations that a technician can make is an upper bound for the maximum number of visits in a day, we use this constant in our model. Constraints (19) subtract 1 from g_{is} every time a technician visits a node. Similar to vehicle subtour elimination constraints, since g_{js} is forced to be less than or equal to g_{is} if request j is visited immediately after request i or home H_s by technician s (i.e., $z^t_{ijs} = 1$), subtours never form in technician routes.

Constraints (21)-(22) ensure that the solution complies with the working day restrictions for the technicians. According to constraints (21) and (22), if a technician works four or fewer consecutive days, i.e. LHS is less than or equal to 4, he/she can work either the next day or the day after the next day unless he/she is working more than five consecutive days. If a technician works five consecutive days, i.e. LHS is equal to 5, the technician cannot work the next two consecutive days. Constraints (23) ensure that this restriction still holds at the end of the planning horizon and prevents any technician from working more than 5 days in the last 6 days.

Constraints (24) calculate the idling time, the difference between the delivery day and the installation day, for all requests. Since b_i is defined as a non-negative integer, these constraints also ensure that machines are installed at least one day after the delivery for every request.

5 Hyper-heuristics Methodology of CVRP for Delivery and Installation of Machines

In this section we describe the hyper-heuristic framework applied to this problem and discuss solution initialisation and representation and the low level heuristics set.

5.1 Population-based Hyper-heuristic Framework

Following the description of the selection hyper-heuristic framework in Section 2.2, most of the previously proposed solution methodologies in selection hyper-heuristics utilise a single solution during the search process and iteratively improve it, while some other methodologies adopt the idea of using a population of solutions during the search as a whole, or during some part of it. Our proposed framework is based on a population of solutions from which one of them will be selected and applied to a selection hyper-heuristic at each step in the search. We are motivated in this work to use a population of solutions as we believe that this provides diversity in the search and allows better exploration of new areas in the search space. The process starts by initialising

Algorithm 1: Algorithm of the population based hyper-heuristic

```
1  Let S_current, S_new, S_best, S_global be current, new, best and global solutions respectively;
2  Let HH = [hh_1, hh_2, ..., hh_n] be the combinations of selection hyper-heuristics;
3  Let pop = [sol_1, sol_2, ..., sol_popsize] be the solutions in the population;
4  Let LLH = [llh_1, llh_2, ..., llh_|LLH|] be the set of low level heuristics;
5  pop ← InitialGeneration();
6  repeat
7      sol_i ← SelectRandomly(pop);
8      S_current ← sol_i;
9      S_best ← S_current;
10     hh_j ← SelectRandomly(HH);
11     repeat
12         llh ← Select(hh_j, LLH);
13         S_new ← ApplyLLH(llh, S_current) ;
14         if Accept(hh_j, S_new, S_current) then
15             S_current = S_new;
16         if S_current isBetterThan S_best then
17             S_best ← S_current;
18     until TerminationCriteria;
19     if S_best isBetterThan S_global then
20         S_global ← S_best;
21     sol_i ← S_best;
22     Shuffle(sol_i);
23 until timeLimit;
24 return S_global;
```

a number of solutions using a generation method to create an initial population $pop = \{sol_1, sol_2, ... sol_{pop_{size}}\}$. A number of selection hyper-heuristics $HH = \{hh_1, hh_2, ... hh_n\}$ combining di erent selection and move acceptance methods are implemented.

A solution sol_i and a selection hyper-heuristic hh_j are randomly selected from pop and HH respectively, where sol_i will serve as $S_{current}$ to hh_j. The selection hyper-heuristic hh_j selects a heuristic (or sequence of heuristics) and applies it to $S_{current}$ to create new solution S_{new}, which is checked for feasibil-

ity, and rejected if it is not feasible (i.e. violates at least one of the problem constraints). If S_{new} is feasible it will be evaluated and the decision of its acceptance is made by the move acceptance component of hh_j. The best found solution S_{best} is replaced by S_{new} if it is better. The iteration between the selection and acceptance components continues until the termination criteria are satis ed, that is until the global time limit is exceeded or when there is no improvement on the best obtained solution for a certain number of iterations.

After hh_j terminates, S_{best} is checked against the best found global solution S_{global} and replaces it if it is better. Afterwards, S_{best} is shu ed by randomly selecting and applying a series of low level heuristics. The number of steps to shu e a solution is tuned by the user, and is constant during the search. This shu ing is necessary in order to avoid the possibility of early convergence and to refresh the population. Next, a solution from the population and a selection hyper-heuristic are randomly selected and the same steps mentioned above are repeated. This iterates until the speci ed time for running an instance passes. Algorithm 1 outlines our applied framework.

For this framework we have tested a total of eight selection hyper-heuristics combining the selection methods: Simple Random (SR), Sequence-based Selection Hyper-heuristic (SS), and the move acceptance methods: Record-to-Record (RR), Na•ve acceptance (Na•ve), Great Deluge (GD), and Simulated Annealing (SA).

SR is the most basic heuristic selection method. It randomly selects a low level heuristic at each step according to a uniform probability distribution. SS on the other hand applies sequences of heuristics to the solution, instead of single applications. This selection method learns and identi es the sequences of heuristics most likely to improve the current solution. More details of this method can be found in (Kheiri and Keedwell, 2015, 2017).

The move acceptance methods applied accept non-worsening solutions, and worsening solutions are accepted using di erent criteria.

In Na•ve and SA, the worsening solutions are accepted with a certain probability. This probability is xed for Na•ve which is prede ned by the user, while in SA, the probability varies in time and it is calculated using the following formula:

$$p_t = e^{\frac{\Delta f}{\Delta F(1-\frac{t_{current}}{t_{limit}})}} \tag{28}$$

Where Δf is the change in the cost at time $t_{current}$, ΔF is the expected maximum change in the cost, and t_{limit} is the time limit. GD is a threshold move acceptance method which allows worsening solutions if their cost (objective) value is less than or equal to a threshold value referred to as \level" (τ_t) which gets updated at each time step (t) during the search. The level is initially set to the initial cost. The update formula for the level is as follows:

$$\tau_t = f + \Delta F \times (1 - \frac{t_{current}}{t_{limit}}) \tag{29}$$

where f is the nal expected cost value, ΔF is the maximum expected change in the cost value, and $t_{current}$, t_{limit} is the time at the current step, and the time

limit respectively. RR is a variant of GD which accepts worsening solutions that are not much worse than the best solution in hand to an extent based on the following formula:

$$obj(S_{new}) \leq obj(s_{best}) + fr \times obj(S_{best}) \quad (30)$$

Where fr is a factor that is updated during the search, starting with a large value and gradually decreasing.

5.2 Solution Representation Scheme

The described hyper-heuristic framework requires initialising a number of solutions to build a population, and this is achieved using an initial generation method. The structure of each of the initialised solutions is demonstrated by Figure 3. Each solution is composed of two main components: truck visits, and technician visits. The truck visits component corresponds to the schedule of trucks during the planning horizon which can be modelled as four levels: days, trucks dispatched on each day, tours performed by each truck, and the requests to deliver on each tour. Similarly, the technician visits correspond to the technicians' schedule composed of three levels: days, technicians scheduled on each day, and visits performed by each technician. The initial generation method randomly produces these schedules, while ensuring the nal constructed solution is feasible. The main focus of the initial generation method is the feasibility of the solution and not its quality.

The feasibility of the solution must also be maintained during the hyper-heuristic operation. A feasibility test is implemented to ensure that the constraints described in Section 3 still hold after each application of low level heuristic(s). A single violation of any of these constraints results in rejecting the solution. This test prevents the evaluation of too many infeasible solutions which can consume valuable search time.

5.3 Low Level Heuristics

The hyper-heuristic controls a total of twenty ve low level heuristics to improve the quality of a given initial solution. These low level heuristics perform swap and insert operations for requests in truck and technician routes. Low level heuristics are restricted, as needed, to only produce routes that respect some of the constraints. For example, some low level heuristics perform operations between di erent days in the planning period; in this case if the operation involves delivery requests, the time windows of these requests must be respected and any installations that as a consequence violate the time windows constraints must be rescheduled. If it involves installation requests, the delivery of these requests must be ensured at least the day before.

– **LLH0**: selects a random day, a random truck route and a random tour, and swaps any two randomly selected requests on this tour.

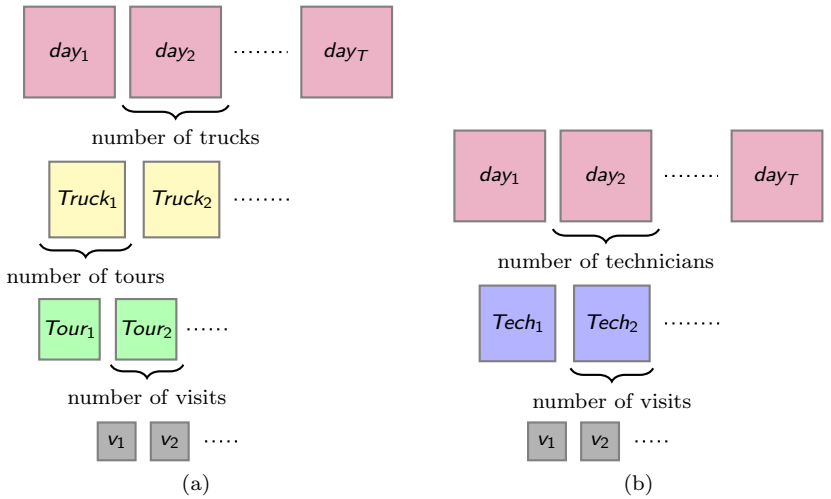
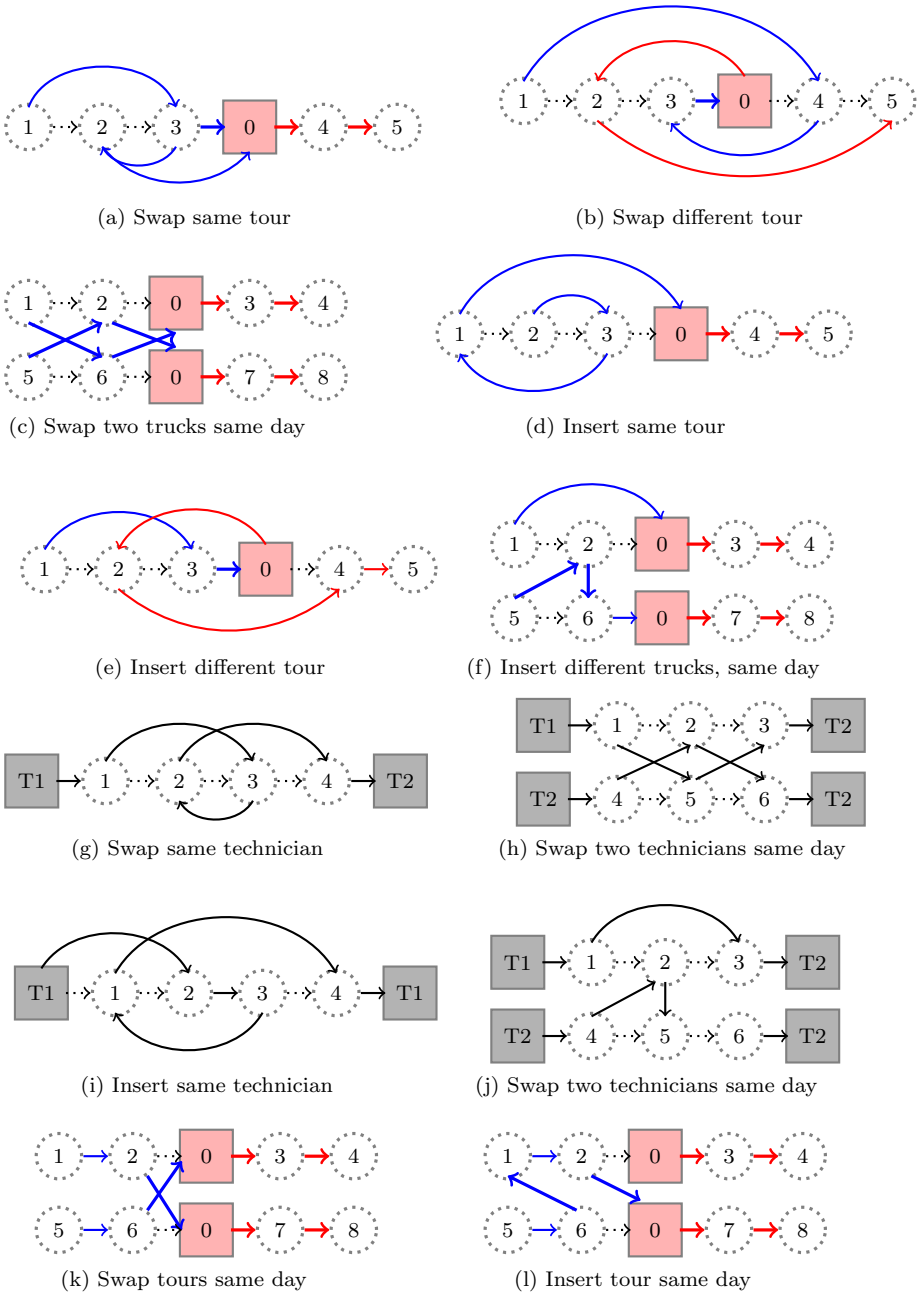


Fig. 3: Solution representation

- **LLH1**: selects a random day, a random truck route, and two di erent random tours, and swaps any two randomly selected requests on each tour.
- **LLH2**: selects a random day, two di erent random truck routes, two random tours from each route, and swaps two randomly selected requests from each tour.
- **LLH3**: selects two di erent random days, two random truck routes from each day, and two random tours from each route, and swaps two randomly selected requests from each tour.
- **LLH4**: selects a random day, a random technician scheduled on this day, and swaps two randomly selected requests of this technician.
- **LLH5**: selects a random day, two di erent random technicians scheduled on this day, and swaps two randomly selected requests of these technicians.
- **LLH6**: selects two di erent random days, and two random technicians, and swaps two randomly selected requests of these technicians.
- **LLH7**: selects a random day, a random truck route, a random tour, and two random positions on this tour. The request on the rst position is inserted into the second position.
- **LLH8**: selects a random day, a random truck route, two di erent random tours on the selected route, and a random position on each tour. The request on the rst position of the rst tour, is inserted into the second position of the second tour.
- **LLH9**: selects a random day, two di erent random truck routes, a random tour on each route, and a random position on each tour. The request on the rst position is inserted into the second position.
- **LLH10**: selects two di erent random days, a random truck route on each day, a random tour on each route, and a random position on each tour. The request on the rst position is inserted into the second position.

- **LLH11**: selects a random day, a random technician scheduled on this day, and two random positions on the technician route. The request on the first position is inserted into the second position.
- **LLH12**: selects a random day, two different random technicians, and a random position on each technician route, and inserts the request on the first position into the second position.
- **LLH13**: selects two different random days, a random technician on each day, and a random position on each technician route. The request on the first position is inserted into the second position.
- **LLH14**: selects a random day, two different random truck routes, and a random tour on each route, and swaps the two selected tours.
- **LLH15**: selects two different random days, a random truck route on each day, and a random tour on each route, and swaps the two selected tours.
- **LLH16**: selects a random day, two different random truck routes, and a random position on each route. The tour on the first position is inserted into the second position.
- **LLH17**: selects two different random days, a random truck route on each day, and a random position on each route. The tour on the first position is inserted into the second position.
- **LLH18**: selects a random day, two different random truck routes, and two random positions. A block of consecutive requests starting at the first position is swapped with another block of requests starting at the second position. The size of the block is randomly selected.
- **LLH19**: selects two different random days, a random truck route on each day, and two random positions on each route. A block of visits starting at each of the positions are swapped with each other.
- **LLH20**: selects a random day and two different random technicians, and swaps two blocks of requests for these technicians.
- **LLH21**: selects two random different days and two random technicians from each day, and swaps two blocks of requests of these technicians.
- **LLH22**: selects a random day, and two different random truck routes. A block of requests is moved from the first route to the second route at randomly selected positions.
- **LLH23**: selects two different random days and two random truck routes. A block of requests is moved from the first route to the second route at a randomly selected positions.
- **LLH24**: selects a random day and two different random technicians, and moves a block of requests from the first technician to the second technician.
- **LLH25**: selects two different random days and two random technicians, and moves a block of requests from the first technician to the second technician.



(a) Swap same tour

(b) Swap different tour

(c) Swap two trucks same day

(d) Insert same tour

(e) Insert different tour

(f) Insert different trucks, same day

(g) Swap same technician

(h) Swap two technicians same day

(i) Insert same technician

(j) Swap two technicians same day

(k) Swap tours same day

(l) Insert tour same day

Fig. 4: Some selected low level heuristic descriptions. Blue and red arrows represent two di erent tours. Dashed arrows are edges removed by the application of the low level heuristic

6 Experimental Results

The experiments were performed on several machines. CPLEX 12.10 and Gurobi 9.0 together with C# and Python were used for the exact technique and Microsoft Visual Studio 2017 C++ for the heuristic method. The experiments for the heuristic method on the hidden dataset were performed on a device with the speci cations: Intel Core i5 at 2.3GHz with memory of 8GB. On both datasets, the experiments were designed according to the competition rules, which required nine runs per instance with nine di erent random seeds also determined by the competition rules. The run time for each instance in both datasets was also calculated according to the competition rules, where it has been speci ed that each instance is run for a limited time on the user machine calculated with the formula: $T_{limit} = fb \times (10 + |R|)$, where T_{limit} is the time limit for running an instance according to the user local machine, fb is a factor calculated by a benchmark tool provided by the competition to estimate the equivalent time on any machine compared to the organisers core machine, and $|R|$ is the number of delivery requests in the instance. To tune these parameters we have followed two approaches: a manual approach where a series of extensive experiments were performed to ne tune the design parameters. We arrived at a combination of parameter values that resulted in a relatively better performance across a subset of public instances. The second approach is using the irace package (Lopez-Ibanez et al., 2016) to automatically tune the parameters on the ve instances with the highest variance in the small dataset. irace performed a maximum of 8000 experiments and ran for about 9 hours to nd the top four con gurations. The best con guration in the top four was selected to perform another round of experiments on the small dataset with the same experimental design (i.e. nine runs per instance with the random seed values set by the competition rules). The parameter settings of the hyper-heuristic using the two approaches are shown in Table 2. The results of the small dataset reported in the next section are the ones found using the manual tuning. For convenience, the developed approach is denoted as POHH.

Table 2: The algorithm parameters and the chosen values

Parameter	Tuning	irace
Population size (pop_{size})	2-5	3
Limit on iterations without improvement	10^5	63144
Naïve acceptance probability	0.1	0.1
RR factor (fr) starting value	10	10.64
Number of iterations to shuffle solutions	10	7

6.1 Results on the Small Dataset

Table 3 provides the results of the small instances dataset for the exact and the hyper-heuristic method. For the exact model, the upper and lower bounds are provided for each instance. The lower bound indicates that an optimal solution was not found for a particular instance. The results of the hyper-heuristic experiments are reported in terms of the minimum and maximum objective values achieved in the nine runs, the average of the nine runs and the standard deviation. The time in seconds is the time that was required to nd the reported results by the exact model, and the time limit for each run of POHH. The time was normalised to its equivalent in the standard machine using the calibration tool provided by the competition. We also note that the execution time of the exact model on any instance was limited to up to 30 minutes.

From Table 3 we can directly compare the performances of the two models in terms of nding optimal solutions and the time required to nd them. The exact model was able to nd optimal solutions for 12 instances out of the 25 with CPLEX and Gurobi, and no feasible solution with CPLEX or Gurobi was found for the instance Small_09. For the rest of the instances the same upper bounds were found by CPLEX and Gurobi, while we notice that for some instances Gurobi was able to nd better lower bounds. Comparing the results of the exact model to the minimum value in the nine runs, POHH was able to nd the optimal solutions in all twelve instances where the exact model found optimal solutions. In the other cases where no optimal solution was proved by the exact model, the POHH algorithm either found the same upper bound or, in the case of seven of the instances, a better upper bound was discovered. Also, a feasible solution for Small_09 was found by POHH. Although the POHH was able to nd the same value for the upper bound as Gurobi and CPLEX in many instances, we cannot yet argue that this upper bound is the optimal solution to these instances. In terms of run time, POHH achieved improved run times in most of the cases. The exact model in many instances required more than 3000 seconds to nd a solution, while POHH required less than 30 seconds on the same instances.

We also compared the results on the small dataset using the manually tuned parameters with the best con guration found by the irace package. The results of the two approaches were very similar using the minimum value in the nine runs as comparison base. Both found the same minimum in all the instances, except for Small_18, in which the irace con guration found a new best result of \479,817,740", and Small_23, in which the manual con guration was better.

6.2 Results on the Hidden Dataset

As mentioned previously, the hidden dataset was used to assess the competitors' algorithms in the restricted resources challenge, and according to the

results of this challenge, eight teams were selected as nalists. We have followed the same competition rules as the other nalists team, with this set of experiments to fairly compare and justify our results.

Table 4 summarises the results of the top six teams, including our hyper-heuristic approach, for each instance ranked based on their best found solution from best to worst. For each instance, the results are reported for the best six teams out of nine using the average of the nine runs and the minimum objective value.

Considering the minimum and average objective values obtained over nine runs for each hidden instance, POHH is relatively competitive between the -nalists. The POHH achieved a position in the top six in 15 instances out of 25. For 11 out of 25 instances, including: Hidden_02, Hidden_07, Hidden_09, Hidden_11, Hidden_13, Hidden_16, Hidden_17, Hidden_19, Hidden_21, Hidden_22, Hidden_25, POHH performs better than $at\ least$ half of the nalist approaches in terms of average objective value. Except the instances Hidden_13 and Hidden_17, the same phenomena is observed with those instances with respect to the minimum objective values. The best achieved results are found on instances Hidden_07, Hidden_11, Hidden_16, and Hidden_21, where POHH is ranked the fourth based on both the average and the minimum objective values. These instances are all of di erent sizes: 150, 450, 600, and 750 requests, re ecting the ability of POHH to perform well on instances with varying characteristics and complexities.

We have also ranked our approach amongst the eight nalists using the same method used in the competition. A ranking score is calculated per instance for each submitted solver by removing the two best and worst solutions found by this solver. We then take the average objective value of these ve solutions as score for the algorithm, and rank all methods accordingly. The average of all ranking scores for the instances represents the nal mean rank of the solver, which was used to order the competitors from the rst to the last. Figure 5 displays the ranking of the POHH algorithm among the eights nalists based on this method. It is clearly seen that our method was able to produce results competitive with the nalists by achieving a better nal mean rank than three teams, and an insigni cant di erence from the ranks of the third, fourth, and fth teams.

Although the proposed algorithm did not succeed in improving any current best known solutions on hidden instances, it performed well (see Section 6.1) on small instances with few requests, and the results on the hidden instances are considered reasonably good.

6.3 Performance Analysis of POHH

Figure 6 visualises the six sample instances, including Small_01, 03, 06 and Hidden_01, 03, 06 that we used for the analysis purposes, re ecting the varying characteristics of each instance. These instances were arbitrarily chosen to represent varying sizes. The visualisation was obtained with the tools provided

Table 3: Summary of the results. The table shows upper bound (or optimum), lower bound (if not optimum), percentage deviation, solution time (in second)

	CPLEX				Gurobi				POHH				
Instance	Upper Bound	Lower Bound	Deviation	Time	Upper Bound	Lower Bound	Deviation	Time	Min	Max	Avg	Std	Time
Small_01	36,016,020			5	36,016,020			0	36,016,020	36,016,020	36,016,020	0	16
Small_02	1,786,430			10	1,786,430			2	1,786,430	1,786,520	1,786,440	30	18
Small_03	32,085,900	14,442,700	55	1,800	32,085,800	21,461,179	33	1,800	32,085,800	32,086,000	32,085,922	83	20
Small_04	18,816,347	14,825,629	21	1,807	18,816,347			1,797	18,816,347	18,816,555	18,816,393	92	22
Small_05	128,025,356	112,613,264	12	1,800	128,124,700	112,490,056	12	1,800	128,014,700	128,015,356	128,015,034	237	24
Small_06	180,524	154,910	14	1,805	180,524	171,931	5	1,800	180,524	180,524	180,524	0	26
Small_07	239,350,700			92	239,350,700			27	239,350,700	239,350,700	239,350,700	0	16
Small_08	66,877,075	34,378,156	49	1,800	66,877,075			408	66,877,075	66,878,040	66,877,826	426	18
Small_09	Infeasible Solution			1,800	Infeasible Solution			1,800	1,073,410	1,073,510	1,073,421	33	20
Small_10	133,887,780	133,873,627	0	1,800	133,887,780			270	133,887,780	133,887,790	133,887,789	3	22
Small_11	537,212,505	341,632,540	36	1,800	537,613,730	363,037,030	32	1,800	537,212,505	537,213,820	537,212,866	577	24
Small_12	16,209,060	6,570,854	59	1,800	16,205,455	6,528,676	60	1,800	16,203,435	16,205,395	16,204,949	581	26
Small_13	221,514,230			202	221,514,230			367	221,514,230	221,514,230	221,514,230	0	16
Small_14	211,980	101,866	52	1,802	212,005	113,930	46	1,800	211,980	212,080	212,003	35	18
Small_15	2,148,165			122	2,148,165			33	2,148,165	2,151,980	2,149,312	1,442	20
Small_16	333,416,760	138,661,467	58	1,800	333,416,760	158,241,503	53	1,800	333,416,760	333,422,700	333,418,473	2,203	22
Small_17	758,620	615,961	19	1,810	758,620	660,245	13	1,800	758,620	758,620	758,620	0	24
Small_18	480,118,510	194,470,467	59	1,842	479,818,890	219,958,625	54	1,800	479,817,740	479,821,390	479,819,453	969	26
Small_19	877,960			7	877,960			0	877,960	878,320	878,080	180	16
Small_20	1,763,630	1,413,949	20	1,802	1,763,630			528	1,763,630	1,766,945	1,764,367	1,462	18
Small_21	1,074,748	893,161	17	1,802	1,074,748			1,255	1,074,748	1,074,748	1,074,748	0	20
Small_22	8,326,840	6,880,399	17	1,802	8,326,840	7,542,245	9	1,800	8,326,840	8,326,840	8,326,840	0	22
Small_23	35,499,570	23,517,476	34	1,802	35,499,570	27,966,620	21	1,800	35,499,539	35,499,580	35,499,563	19	24
Small_24	182,073,280	130,142,469	29	1,805	182,073,280	132,738,156	27	1,800	182,072,650	182,099,930	182,076,697	8,778	26
Small_25	3,499,190			785	3,499,190			10	3,499,190	3,499,190	3,499,190	0	16

Table 4: The performance comparison of the nalists and POHH, based on the average, and minimum of the objective values over 9 runs for each instance. The top six methods per each instance are reported and best values are highlighted in bold

Instance	Team	Min	Avg	Instance	Team	Min	Avg	Instance	Team	Min	Avg
Hidden_01	UOS	**67303070**	**67347510.0**	Hidden_10	UOS	**31425420**	**31615172.8**	Hidden_19	UOS	**4379062260**	**4379503211.1**
	MJG	67539160	67768841.7		MJG	32134970	32406070		MJG	4379599620	4384265171
	CokaCoders	67936410	68239128.3		TCSExplorer	35602350	36296800		CokaCoders	4385514720	4416094113
	AAVK	68049230	68497476.7		CokaCoders	41951125	44037471.7		orlab	4390908575	4397116298
	orlab	68059355	68531090.6		orlab	42557550	43110900.6		POHH	4400633465	4433562786
	TCSExplorer	68067705	68669280		POHH	44757390	46971126.1		justFall	4415787735	4460775575
Hidden_02	UOS	**866780485**	**884328838.3**	Hidden_11	UOS	**4052063633**	**4143464703.2**	Hidden_20	UOS	**126020890**	**127117758.9**
	MJG	878698335	887583730		MJG	4104065729	4150825522		MJG	128220285	130004861.7
	TCSExplorer	940755820	966644283.9		TCSExplorer	4490010535	4573692086		TCSExplorer	138750815	141489971.1
	CokaCoders	978877270	992970729.3		POHH	4792051226	4947603379		CokaCoders	143190180	146078606.1
	POHH	999703125	1070465586		justFall	4933454022	5124240633		orlab	164537160	167708532.2
	justFall	1178709770	1289905508		AAVK	5050880606	5176963402		POHH	171503820	177538151.1
Hidden_03	UOS	**1353070685**	**1356206683.3**	Hidden_12	UOS	**2985079895**	**2988919325.0**	Hidden_21	UOS	**33041255**	**33217240.6**
	MJG	1358875910	1363989363		MJG	2985608315	2989299754		MJG	33313460	33871058.9
	CokaCoders	1362273745	1374383989		CokaCoders	2998137785	3062709553		TCSExplorer	35525950	36048057.8
	orlab	1365243450	1373092383		orlab	3020441765	3030703193		AAVK	38119685	38845431.7
	AAVK	1388162220	1394947955		wanderer	3056711365	3059281502		POHH	38347205	38839638.3
	justFall	1389700390	1407156949		justFall	3072299875	3086846959		justFall	40036785	41366946.1
Hidden_04	MJG	**5354045**	**5382928.3**	Hidden_13	UOS	**5237955246**	**5239090235.3**	Hidden_22	MJG	**6642535**	**6677298.3**
	UOS	5407020	5470823.3		MJG	5243444173	5270924784		UOS	6700250	6750354.4
	TCSExplorer	5782480	5973581.1		CokaCoders	5251322598	5267097465		TCSExplorer	6986000	7020108.9
	AAVK	6053945	6122060		orlab	5273032383	5300095233		AAVK	7473115	7575487.8
	orlab	6258020	6409908.3		justFall	5284251162	5322430964		POHH	7671390	7831162.8
	POHH	6750245	6812897.2		POHH	5289271553	5312240754		orlab	7777350	7896450.6
Hidden_05	UOS	**2420668237**	**2438943861.4**	Hidden_14	UOS	**1378863050**	**1380531068.9**	Hidden_23	CokaCoders	**22341696590**	22374478803
	MJG	2461219726	2472452759		orlab	1385370725	1392172054		UOS	22342274615	**22368503380.6**
	CokaCoders	2793126594	2824641910		CokaCoders	1385514390	1387323213		MJG	22473152460	22489158018
	TCSExplorer	2873811043	2900849794		MJG	1386071905	1389987112		orlab	22564706605	22644343334
	wanderer	3463496082	3463496082		AAVK	1394884865	1403511603		justFall	22803650615	22946341903
	orlab	3489050112	3582195247		justFall	1399908710	1414358900		AAVK	22946775355	23021877311
Hidden_06	MJG	**32919590**	**32950587.2**	Hidden_15	UOS	**163646890**	**163861015.0**	Hidden_24	UOS	**31350467425**	**31373781566.1**
	UOS	33035255	33122941.7		orlab	164986370	165539965.6		CokaCoders	31386137725	31441905918
	orlab	33243100	33310187.2		MJG	165442970	165746732.2		orlab	31457461925	31554867014
	AAVK	33537475	33642695		AAVK	166559140	167109473.3		MJG	31502515080	31579529472
	TCSExplorer	33730430	33798883.1		wanderer	167271785	167271785		justFall	31915258830	32102001066
	wanderer	33799140	33799140		TCSExplorer	168523765	168889354.4		wanderer	31945701195	31947545553
Hidden_07	UOS	**102098250**	**102289614.4**	Hidden_16	MJG	**52730075**	**52860556.7**	Hidden_25	UOS	**549505255**	**549854756.1**
	MJG	102375745	103005780.6		UOS	53232615	53561470.6		MJG	552735110	563054914.4
	CokaCoders	107548420	110818258.9		TCSExplorer	58750440	59286328.3		CokaCoders	586771940	605819273.9
	POHH	108198340	109634304.4		POHH	59861645	60499887.8		TCSExplorer	611102855	621490488.9
	TCSExplorer	108308895	110359084.4		justFall	61176575	63036133.3		POHH	625293110	652332916.1
	justFall	121495535	132836498.3		AAVK	61626085	63624953.9		orlab	659488500	678896693.9
Hidden_08	UOS	**728784055**	**729462820.6**	Hidden_17	UOS	**27301086592**	**27322051463.0**				
	MJG	729588325	732733357.2		MJG	27387159376	27426339743				
	CokaCoders	734907040	737740395		CokaCoders	27396284273	27996989611				
	orlab	735203675	737068948.3		orlab	27486127713	27525238097				
	AAVK	743077790	750319370.6		justFall	27717856052	27822517277				
	justFall	746651030	757387192.2		POHH	27867305050	27914747560				
Hidden_09	UOS	**1692627537**	**1713909634.1**	Hidden_18	MJG	**52602450**	**52658013.3**				
	MJG	1732570744	1746683152		UOS	52921995	53040623.3				
	CokaCoders	1884282890	1939897718		AAVK	54778115	55313427.2				
	TCSExplorer	1964621385	2033485372		TCSExplorer	54906655	55136475				
	POHH	2375666512	2439427249		justFall	55108940	58268698.3				
	justFall	2509944603	2639808947		POHH	55585100	56052686.1				

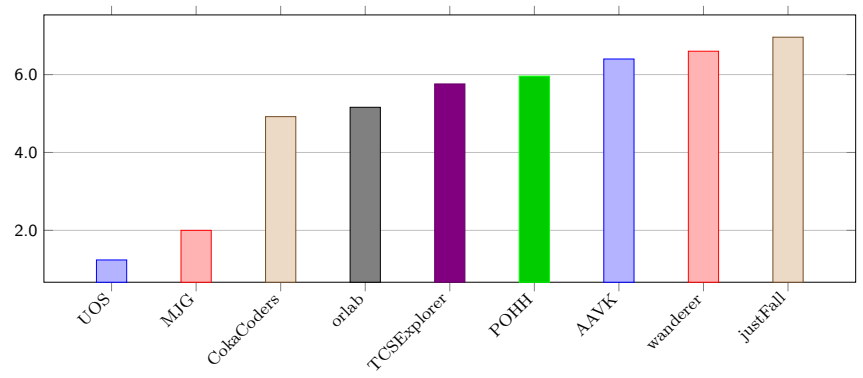


Fig. 5: Mean ranks of the nalists teams and the POHH algorithm computed according to the competition rules

by the challenge organisers (ORTEC and VeRoLog, 2019). The large light blue circle indicates the depot, and this may or may not have technicians on the premises at any given time. Yellow locations, indicated by medium circles, have technicians, and they may or may not have requests. Green locations have requests, but no technicians. The large diameter is used when a location (of any colour) is open, and the small diameter indicates that a location is not open for delivery. The depot (blue) is open throughout the planning horizon, and each request location is open on the days speci ed. The gure shows only the beginning of day 1 for each instance. Figure 6 also shows semi-transparent green circles centred on the locations with technicians. The radius of these circles is equal to the half of the maximum daily distance of the corresponding technician. Because these circles are semitransparent, overlap leads to colour intensi cation, making visually clear which customer locations are within reach of few or many technicians. One may notice from the clustering, for example, as for Hidden_06, that the locations are making the shape of the Netherlands. This implies that these are indeed based on real-world data o ered by ORTEC.

Although Figure 6 gives a rough picture of those instances (e.g. some instances are more limiting in terms of number and action radius of technicians), we must emphasise that it does not describe a given problem instance fully. For example, the importance of violating a given constraint is not depicted and, as we mentioned before, penalties for violating the di erent constraints can di er substantially as a part of the cost function. Table 1 is particularly useful in this case.

The pie charts in Figure 7 depict the utilisation rates of the di erent selection hyper-heuristics applied in our framework. The utilisation is calculated in terms of the ratio between the number of times a selection hyper-heuristic was successful in nding an improved solution over the best global solution to the total number of improvements made by all the selection hyper-heuristics in the duration of run time.

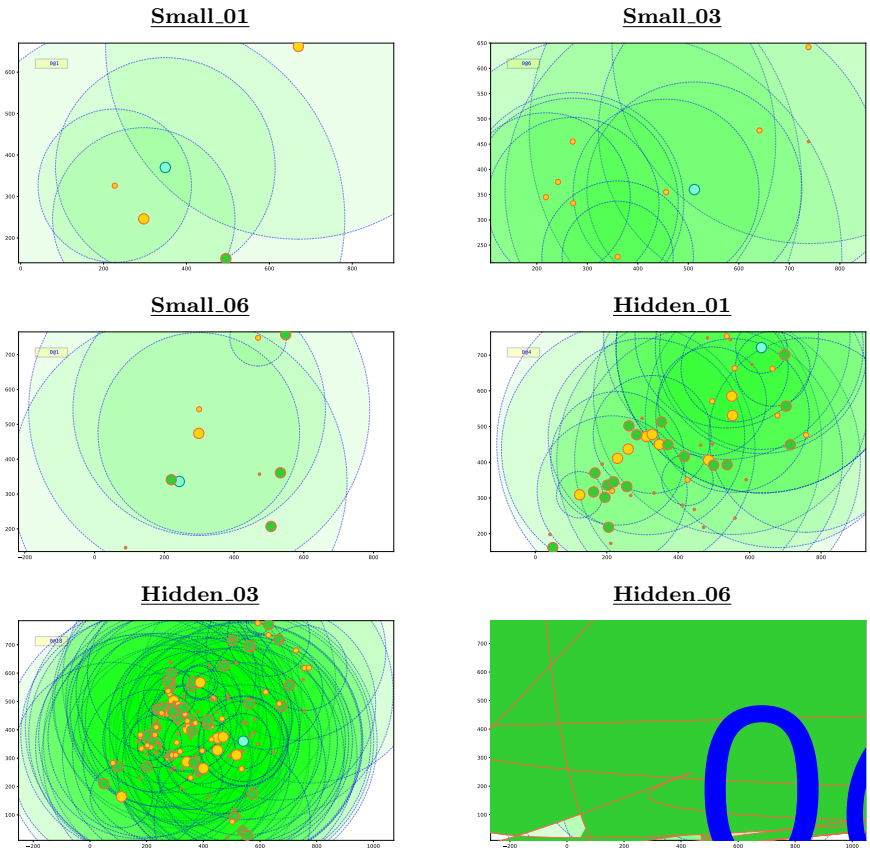


Fig. 6: Visualisation of sample instances

There are particular selection hyper-heuristic methods that clearly performed better than the rest in making improvements to the solutions during the search process. The incorporation of the RR-based selection hyper-heuristics in the POHH appears to play a key role of solving the problem in a relatively effective manner, in particular SS-RR which performed equally well in the small and hidden datasets. SR-GD was very successful in the larger size instances, where 50% of the improvement rate was achieved by SR-GD in Hidden_06 that has 900 requests. The least successful selection hyper-heuristics are the ones combined with the simulated annealing acceptance. The utilisation of SR-SA was down to 0% in all instances, except for an insigni cant improvement rate of 5% in Small_03. Also, SS-SA did make much contribution in terms of improvement for the hidden set. The na•ve acceptance is more successful for the small instances than the larger ones, but only when it is combined with the sequence based selection method.

There seems to be a variation in the performance between the selection hyper-heuristics in instances with di erent complexities, and we cannot generalise that a certain combination of a selection and a move acceptance methods would be successful in every instance in this problem. An interesting idea would be to embed a high-level intelligent control mechanism that can observe these variations, and apply the components of selection hyper-heuristics accordingly during the search time, similar to the online selection methods. The random selection criteria that we apply currently in our framework was able to nd `reasonable' results as reported, and we expect that the suggested improvements in the selection mechanism could yield even better results.

6.3.1 Performance Comparison to the Constituent Hyper-heuristics

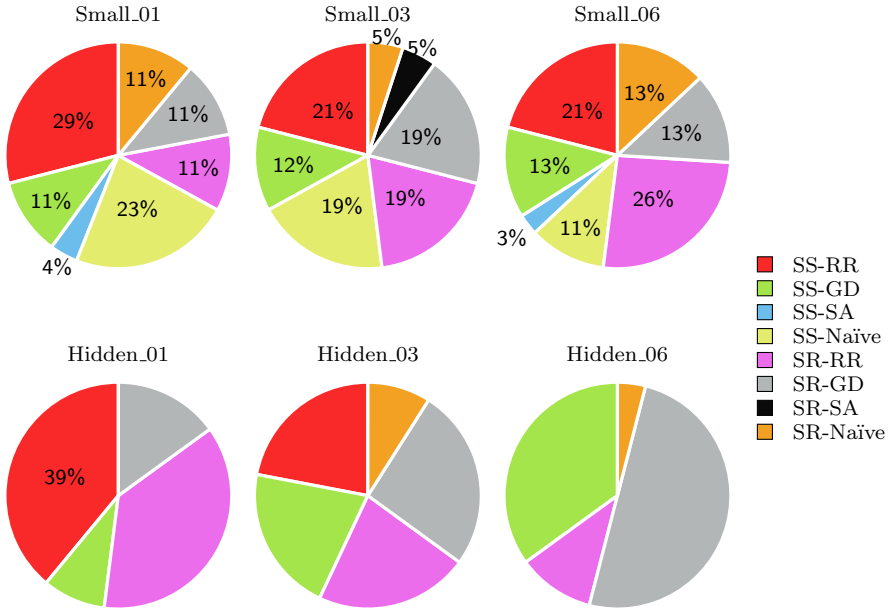
Another round of experiments have been conducted by applying the eight selection hyper-heuristics employed in our framework independently on the instances displayed in Figure 6. Each selection hyper-heuristic was run for nine times using the same rules to calculate the run time of an instance described in this section. The results are displayed in Table 5 using the best and average objective values from the nine runs, along with the associated standard deviation. The MannWhitneyWilcoxon test is performed with a 95% con - dence level in order to compare pairwise performance variations of two given algorithms statistically. The following notations are used: (i) `+' denotes that our algorithm (POHH) is better and this performance variance is statistically signi cant, (ii) `-' denotes that the performance of POHH is worse and this performance variance is statistically signi cant, (iii) `=' indicates that there is no statistical signi cant between the two methods.

The POHH algorithm performed statistically better than each of the constituent hyper-heuristic for all instances, except Hidden_03, where the two methods SR-RR and SS-RR found slightly better averages and performed statistically better. Other than those two cases, the POHH algorithm found the best averages and minimum values in all instances, performing exceptionally better in particular on the largest instance of Hidden_06. This provides evidence for the success of two proposals: 1) utilising multiple solutions allows better exploration and more possibilities for further improvement in new areas of the search space, instead of focusing on a single solution; 2) applying a sequence of selection hyper-heuristics to a solution might be useful in utilising the varying performances of these selection hyper-heuristics. This can lead the search to di erent directions that can yield further improvement.

Overall, POHH creates a synergy among the eight selection hyper-heuristics resulting in an improved performance for almost all instances. Although POHH does not utilise learning, various learning mechanisms embedded into each one of the low level selection hyper-heuristics potentially contributes to the overall success of the proposed approach when compared to the performance of each constituent selection hyper-heuristic.

Table 5: The performance comparison of POHH, SS-RR, SS-GD, SS-SA, SS-Na•ve, SR-RR, SR-GD, SR-SA and SR-Na•ve based on the average (Avg), associated standard deviation (Std), minimum (Min) of the objective values over 9 trials and the pairwise average performance comparison of POHH vs (SS-RR, SS-GD, SS-SA, SS-Na•ve, SR-RR, SR-GD, SR-SA and SR-Na•ve) based on Mann-Whitney-Wilcoxon for each instance produced by each approach. The hyper-heuristic producing the best value for Avg and Min per each instance are highlighted in bold

	Small_01				Small_03				Small_06			
Method	vs	Avg	Std	Min	vs	Avg	Std	Min	vs	Avg	Std	Min
POHH	=	**36016020**	0	**36016020**	=	**32085922**	83	**32085800**	=	**180524**	0	**180524**
SS-RR	=	**36016020**	0	**36016020**	+	32087251	858	**32085800**	+	247213	49513	**180524**
SS-GD	=	**36016020**	0	**36016020**	+	32094142	3088	32089068	+	200911	2055	198515
SS-SA	=	36016433	620	**36016020**	+	32114432	9290	32098239	+	211280	3317	206996
SS-Naïve	=	**36016020**	0	**36016020**	+	32087868	1110	32086696	+	183922	649	182888
SR-RR	+	36016847	620	**36016020**	+	32086500	533	**32085800**	+	280221	0	280221
SR-GD	=	36016219	395	**36016020**	+	32092212	1004	32090760	+	199464	2466	196666
SR-SA	+	36016732	554	**36016020**	+	32109815	5690	32104880	+	210099	2883	205800
SR-Naïve	=	**36016020**	0	**36016020**	+	32086480	462	32085900	+	183029	1056	182161
	Hidden_01				Hidden_03				Hidden_06			
Method	vs	Avg	Std	Min	vs	Avg	Std	Min	vs	Avg	Std	Min
POHH	=	**68683339**	290545	**68151265**	=	1410411798	5353659	1399418890	=	**34157264**	106666	**34008705**
SS-RR	+	69197055	311273	68917615	-	1389617007	3467539	**1381431185**	+	36409477	75941	36313420
SS-GD	+	85782696	778892	84218865	+	1815339912	13783473	1791026650	+	41742650	251435	41235155
SS-SA	+	86554467	657440	85333250	+	1802310946	18417576	1773264280	+	42003434	158875	41697785
SS-Naïve	+	73793282	565531	72767395	+	1537797289	6427024	1529124090	+	36526647	154146	36315960
SR-RR	+	69209684	153049	69060100	-	**1387767927**	2077527	1385115295	+	36291840	47266	36224495
SR-GD	+	86004059	466746	85462365	+	1811403039	8342226	1800385550	+	41963413	197981	41518920
SR-SA	+	86552397	462097	85483775	+	1812705973	4911482	1805213785	+	42044408	139089	41796340
SR-Naïve	+	74076952	437930	73144995	+	1527445028	5356630	1515517920	+	36719875	63682	36626675


Small_01
11%
11%
29%
11%
11%
23%
4%
Small_03
5%
5%
21%
19%
12%
19%
19%
Small_06
13%
21%
13%
13%
26%
3%
11%
Hidden_01
39%
Hidden_03
Hidden_06
SS-RR
SS-GD
SS-SA
SS-Naïve
SR-RR
SR-GD
SR-SA
SR-Naïve

the e ciency of the proposed hyper-heuristic algorithm compared to the results of the exact model, insofar as optimal solutions where found in shorter computational times. The hyper-heuristic results also compared well to the results of the eight nalists of the competition. The approach also performed better than the constituent hyper-heuristics performed on their own, for most of the instances. There is scope for further research into several aspects of POHH. One example is how to decide on which hyper-heuristic strategies to include into our population-based algorithm. Additionally, the algorithm does not have the ability to learn from history which strategy performs well. Thus, it may be bene cial to exclude certain strategies altogether to speed-up the algorithm. 1

ACKNOWLEDGMENTS

Acknowledgements We express our gratitude to the VeRoLog board as well as the organising committee for the VeRoLog Conference that was held in Seville, Spain, June 2019. We would like to thank the organising team of the VeRoLog Solver Challenge 2019 and for developing the visualiser presented in the results section.

References

Ahmed L, Mumford C, Kheiri A (2019) Solving urban transit route design problem using selection hyper-heuristics. European Journal of Operational Research 274(2):545{559

Alonso F, Alvarez MJ, Beasley JE (2008) A tabu search algorithm for the periodic vehicle routing problem with multiple vehicle trips and accessibility restrictions. Journal of the Operational Research Society 59(7):963{976

Archetti C, Speranza MG (2012) Vehicle routing problems with split deliveries. International transactions in operational research 19(1-2):3{22

Archetti C, Speranza MG, Savelsbergh MW (2008) An optimization-based heuristic for the split delivery vehicle routing problem. Transportation Science 42(1):22{31

Archetti C, Bianchessi N, Speranza MG (2011) A column generation approach for the split delivery vehicle routing problem. Networks 58(4):241{254

Archetti C, Jabali O, Speranza MG (2015) Multi-period vehicle routing problem with due dates. Computers & Operations Research 61:122{134

Augerat P, Belenguer JM, Benavent E, Corberan A, Naddef D, Rinaldi G (1995) Computational results with a branch and cut code for the capacitated vehicle routing problem. IMAG

Bae H, Moon I (2016) Multi-depot vehicle routing problem with time windows considering delivery and installation vehicles. Applied Mathematical Modelling 40(13-14):6536{6549

Balinski ML, Quandt RE (1964) On an integer program for a delivery problem. Operations research 12(2):300{304

Belhaiza S, Hansen P, Laporte G (2014) A hybrid variable neighborhood tabu search heuristic for the vehicle routing problem with multiple time windows. Computers & Operations Research 52:269{281

Bertels S, Fahle T (2006) A hybrid setup for a hybrid scenario: combining heuristics for the home health care problem. Computers & Operations Research 33(10):2866{2890

Bilgin B, Özcan E, Korkmaz EE (2006) An experimental study on hyper-heuristics and exam timetabling. In: International Conference on the Practice and Theory of Automated Timetabling, Springer, pp 394{412

Boudia M, Prins C, Reghioui M (2007) An e ective memetic algorithm with population management for the split delivery vehicle routing problem. In: International Workshop on Hybrid Metaheuristics, Springer, pp 16{30

Burke EK, Hyde M, Kendall G, Ochoa G, Özcan E, Woodward JR (2019) A classi cation of hyper-heuristic approaches: revisited. In: Handbook of metaheuristics, vol 272, Springer, pp 453{477

Campbell AM, Wilson JH (2014) Forty years of periodic vehicle routing. Networks 63(1):2{15

Cattaruzza D, Absi N, Feillet D, Vigo D (2014) An iterated local search for the multi-commodity multi-trip vehicle routing problem with time windows. Computers & Operations Research 51:257{267

Chen Y, Mourdjis P, Polack F, Cowling P, Remde S (2016) Evaluating hyper-heuristics and local search operators for periodic routing problems. In: Evolutionary Computation in Combinatorial Optimization, Springer, pp 104{120

Cheng CB, Wang KP (2009) Solving a vehicle routing problem with time windows by a decomposition technique and a genetic algorithm. Expert Systems with Applications 36(4):7758{7763

Clarke G, Wright JW (1964) Scheduling of vehicles from a central depot to a number of delivery points. Operations research 12(4):568{581

Cordeau JF, Laporte G, Pasin F, Ropke S (2010) Scheduling technicians and tasks in a telecommunications company. Journal of Scheduling 13(4):393{409

Cowling P, Kendall G, Soubeiga E (2000) A hyperheuristic approach to scheduling a sales summit. In: International Conference on the Practice and Theory of Automated Timetabling, Springer, pp 176{190

Cowling P, Kendall G, Han L (2002) An investigation of a hyperheuristic genetic algorithm applied to a trainer scheduling problem. In: Proceedings of the 2002 Congress on Evolutionary Computation. CEC'02 (Cat. No. 02TH8600), IEEE, vol 2, pp 1185{1190

Dantzig GB, Ramser JH (1959) The truck dispatching problem. Management science 6(1):80{91

Ding Q, Hu X, Sun L, Wang Y (2012) An improved ant colony optimization and its application to vehicle routing problem with time windows. Neurocomputing 98:101{107

Drake JH, Kheiri A, Özcan E, Burke EK (2020) Recent advances in selection hyper-heuristics. European Journal of Operational Research 285(2):405{428

Dror M, Trudeau P (1989) Savings by split delivery routing. Transportation Science 23(2):141{145

Dror M, Trudeau P (1990) Split delivery routing. Naval Research Logistics (NRL) 37(3):383{402

Fisher H (1963) Probabilistic learning combinations of local job-shop scheduling rules. Industrial scheduling pp 225{251

Fisher ML, Jaikumar R (1981) A generalized assignment heuristic for vehicle routing. Networks 11(2):109{124

Fukasawa R, Longo H, Lysgaard J, de Aragao MP, Reis M, Uchoa E, Werneck RF (2006) Robust branch-and-cut-and-price for the capacitated vehicle routing problem. Mathematical programming 106(3):491{511

Gromicho J, vant Hof P, Vigo D (2019) The verolog solver challenge 2019. Journal on Vehicle Routing Algorithms pp 1{3

Gu W, Cattaruzza D, Ogier M, Semet F (2019) Adaptive large neighborhood search for the commodity constrained split delivery vrp. Computers & Operations Research 112:104761

Hsiao PC, Chiang TC, Fu LC (2012) A vns-based hyper-heuristic with adaptive computational budget of local search. In: 2012 IEEE Congress on Evolutionary Computation, IEEE, pp 1{8

Kheiri A (2020) Heuristic sequence selection for inventory routing problem. Transportation Science 54(2):302{312

Kheiri A, Keedwell E (2015) A sequence-based selection hyper-heuristic utilising a hidden Markov model. In: Proceedings of the 2015 on Genetic and Evolutionary Computation Conference, GECCO '15, pp 417{424

Kheiri A, Keedwell E (2017) A hidden Markov model approach to the problem of heuristic selection in hyper-heuristics with a case study in high school timetabling problems. Evolutionary Computation 25(3):473{501

Kheiri A, Keedwell E, Gibson MJ, Savic D (2015) Sequence analysis-based hyper-heuristics for water distribution network optimisation. Procedia Engineering 119:1269{1277, Computing and Control for the Water Industry (CCWI2015) Sharing the best practice in water management

Kheiri A, Dragomir AG, Mueller D, Gromicho J, Jagtenberg C, van Hoorn JJ (2019) Tackling a vrp challenge to redistribute scarce equipment within time windows using metaheuristic algorithms. EURO Journal on Transportation and Logistics 8(5):561{595

Kolen AW, Rinnooy Kan A, Trienekens HW (1987) Vehicle routing with time windows. Operations Research 35(2):266{273

Kovacs AA, Parragh SN, Doerner KF, Hartl RF (2012) Adaptive large neighborhood search for service technician routing and scheduling problems. Journal of scheduling 15(5):579{600

Laporte G (2009) Fifty years of vehicle routing. Transportation Science 43(4):408{416

Lehrbaum A, Musliu N (2012) A new hyperheuristic algorithm for cross-domain search problems. In: International Conference on Learning and Intelligent Optimization, Springer, pp 437{442

Sontrop H, Van Der Horn P, Uetz M (2005) Fast ejection chain algorithms for vehicle routing with time windows. In: International Workshop on Hybrid Metaheuristics, Springer, pp 78{89

Tatarakis A, Minis I (2009) Stochastic single vehicle routing with a predened customer sequence and multiple depot returns. European Journal of Operational Research 197(2):557{571

Tavakkoli-Moghaddam R, Gazanfari M, Alinaghian M, Salamatbakhsh A, Norouzi N (2011) A new mathematical model for a competitive vehicle routing problem with time windows solved by simulated annealing. Journal of manufacturing systems 30(2):83{92

Tsirimpas P, Tatarakis A, Minis I, Kyriakidis E (2008) Single vehicle routing with a predened customer sequence and multiple depot returns. European Journal of Operational Research 187(2):483{495

Urra E, Cubillos C, Cabrera-Paniagua D (2015) A hyperheuristic for the dial-a-ride problem with time windows. Mathematical Problems in Engineering 2015, DOI https://doi.org/10.1155/2015/707056

Walker JD, Ochoa G, Gendreau M, Burke EK (2012) Vehicle routing and adaptive iterated local search within the hyflex hyper-heuristic framework. In: International conference on learning and intelligent optimization, Springer, pp 265{276

Wilson D, Rodrigues S, Segura C, Loshchilov I, Hutter F, Buenfil GL, Kheiri A, Keedwell E, Ocampo-Pineda M, zcan E, Pea SIV, Goldman B, Rionda SB, Hernndez-Aguirre A, Veeramachaneni K, Cussat-Blanc S (2018) Evolutionary computation for wind farm layout optimization. Renewable Energy 126:681{691

Xie F, Potts CN, Bektas T (2017) Iterated local search for workforce scheduling and routing problems. Journal of Heuristics 23(6):471{500

Xu J, Chiu SY (2001) Effective heuristic procedures for a field technician scheduling problem. Journal of Heuristics 7(5):495{509

Zhang Y, Chen X (2014) An optimization model for the vehicle routing problem in multi-product frozen food delivery. Journal of applied research and technology 12(2):239{250