EURO J Transp Logist (2019) 8:561–595
https://doi.org/10.1007/s13676-019-00143-8

RESEARCH PAPER

Tackling a VRP challenge to redistribute scarce equipment within time windows using metaheuristic algorithms

Ahmed Kheiri[1] · Alina G. Dragomir[2] · David Mueller[3] · Joaquim Gromicho[4,5] · Caroline Jagtenberg[4] · Jelke J. van Hoorn[4]

Received: 2 June 2018 / Accepted: 15 April 2019 / Published online: 25 April 2019

Abstract

This paper reports on the results of the VeRoLog Solver Challenge 2016–2017: the third solver challenge facilitated by VeRoLog, the EURO Working Group on Vehicle Routing and Logistics Optimization. The authors are the winners of second and third places, combined with members of the challenge organizing committee. The problem central to the challenge was a rich VRP: expensive and, therefore, scarce equipment was to be redistributed over customer locations within time windows. The difficulty was in creating combinations of pickups and deliveries that reduce the amount of equipment needed to execute the schedule, as well as the lengths of the routes and the number of vehicles used. This paper gives a description of the solution methods of the above-mentioned participants. The second place method involves sequences of 22 low level heuristics: each of these heuristics is associated with a transition probability to move to another low level heuristic. A randomly drawn sequence of these heuristics is applied to an initial solution, after which the probabilities are updated depending on whether or not this sequence improved the objective value, hence increasing the chance of selecting the sequences that generate improved solutions. The third place method decomposes the problem into two independent parts: first, it schedules the delivery days for all requests using a genetic algorithm. Each schedule in the genetic algorithm is evaluated by estimating its cost using a deterministic routing algorithm that constructs feasible routes for each day. After spending 80 percent of time in this phase, the last 20 percent of the computation time is spent on Variable Neighborhood Descent to further improve the routes found by the deterministic routing algorithm. This article finishes with an in-depth comparison of the results of the two approaches.

Keywords Routing · Evolutionary computations · Metaheuristics · Inventory

✉ Ahmed Kheiri
a.kheiri@lancaster.ac.uk

Extended author information available on the last page of the article

Table 1 The participants' institutions. If known, the names of the participants are also mentioned

Team	Institution	Country	Participants
–	CIRRELT	Canada	
–	Erasmus University Rotterdam	The Netherlands	
–	Ho Chi Minh City International University	Vietnam	
–	Koc University	Turkey	
akhe	Lancaster University	United Kingdom	Ahmed Kheiri
–	Los Andes University	Colombia	
–	Maastricht University	The Netherlands	
Success	Shiraz University	Iran	Morteza Keshtkaran
–	Tages s.c.	Italy	
–	Tata Consultancy Services	Unknown	
–	Universidad de Buenos Aires	Argentina	
Tau17	Tel Aviv University	Israel	Yael Arbel Dafna Piotro Alona Raucher Tal Raviv
MLS	Universidad de Los Andes	Colombia	María Ángel Lucia Paris
–	Universidad Torcuato Di Tella	Argentina	
–	University College Dublin	Ireland	
–	University of Groningen	The Netherlands	
–	University of Laguna	Spain	
–	University of Pavia	Italy	
–	University of Pisa	Italy	
mjg	University of the Federal Armed Forces Hamburg	Germany	Martin Geiger
–	University of Twente	The Netherlands	
ADDM	University of Vienna and Vienna University of Technology	Austria	Alina Dragomir David Müller
–	Vrije Universiteit Amsterdam	The Netherlands	
–	Warwick University	United Kingdom	
SunBeams	Zaporizhzhya National University	Ukraine	Igor Kozin Sergey Borue Olena Kryvtsun

1 Introduction

The VeRoLog Solver Challenge 2016–2017 was facilitated by VeRoLog, the EURO Working Group on Vehicle Routing and Logistics Optimization and organized in cooperation with ORTEC. This challenge inspired a total of 28 teams, worldwide, to participate. Table 1 lists the institutions to which the participants were affiliated. The

first, second, and third prize were awarded during the VeRoLog conference 2017 in Amsterdam[1].

ORTEC designed and ran the third solver challenge organized by the VeRoLog. The previous two editions took place in 2014[2] and 2015[3] and were designed and run by PTV. This paper concerns the third challenge[4,5].

The routing problem central to the VeRoLog Solver Challenge 2016–2017 was based on a real-life problem for one of ORTEC's clients. From this real-life problem, a few aspects were selected that lead to a new research topic in vehicle routing optimization.

The problem concerns a large cattle improvement company, that must regularly measure the milk quality at a number of farms (customers). This requires special measuring tools, which have to be delivered to the customers at their request. After the measurement, typically a few days later, the tools have to be picked up again. The scheduling of these deliveries to days and the routing for the planned deliveries and pickups are the issues to address in this challenge.

The problem of this challenge is deterministic and revolves over a long horizon, meaning that the scheduling of the individual delivery dates has a large impact on solution quality. The problem was first introduced by Gromicho et al. (2015) and in the context of the challenge by Dullaert et al. (2017). It combines the following decisions:

- On which day each delivery request should be served. This leads to a second automatic decision: on which day it should be picked up again.
- For each day in the planning horizon, which deliveries and which pickups to combine in each route and in what sequence.

The main objective is to serve all requests at a minimum cost (see Sect. 2.1), subject to the following constraints:

- The number of items (equipment or tools) available per type is limitative, making them scarce and forcing reuse.
- Items can be loaded at the depot or at a customer, but the items on board must always satisfy the capacity constraints.

We may recognize some of the aspects of the problem in the available literature. The routing part of our problem consists of unpaired pickups and deliveries which is central to the Pickup and Delivery VRP (PDVRP) as defined in Parragh et al. (2008), where they argued that this problem had received the least attention of all problems which they have surveyed. Only one paper was known to the authors of

[1] https://verolog2017.sciencesconf.org/.
[2] www.euro-online.org/websites/verolog/verolog-solver-challenge-2014/.
[3] www.euro-online.org/websites/verolog/news/verolog-solver-challenge-2015/.
[4] www.euro-online.org/websites/verolog/news/verolog-solver-challenge-2016-2017/.
[5] https://verolog2017.ortec.com/.

Parragh et al. (2008) addressing unpaired pickups and deliveries being repositioned by multiple vehicles, namely Dror et al. (1998). The problem being addressed there resembles in many aspects the routing of a specific day in our challenge, including the scarcity of the goods and the possibility of them being repositioned (shared electric cars). Dror et al. (1998) developed an exact methodology based on a mixed integer programming model, but the applicability was limited to very small instances.

Close to the time of writing Parragh et al. (2008), Montané and Galvão (2006) designed the first metaheuristic for the routing of unpaired pickups and deliveries by multiple vehicles, but no repositioning was considered: each vehicle departs from a single depot with the loads to deliver and returns to the same depot with the picked up loads. This seems to be the setting generally followed by subsequent research.

Battarra et al. (2014) consider the VRP with simultaneous pickup and delivery demands and attribute its origin to the work of Min (1989) which considered simultaneous pickup and delivery in the context of a public library. The study of Min (1989), which seems to be overlooked by Parragh et al. (2008), also assumes that the deliveries come from a depot and the pickups return to the same depot, hence disallows relocation. It is worthy to mention that their proposed mathematical model includes a parameter to model the traffic congestion, which makes sense since the setting of their study was a large urban area. This is simply done by adjusting the travel times on the arcs. However, just as in the case of our challenge, no detailed consideration is given to the time aspect and the travel times appear only in the objective function.

Another aspect that received substantial attention is that of pickups and deliveries of individual items, which is central to so-called dial-a-ride models (Cordeau and Laporte 2007). These models deal with the transportation of people and tend to focus on the journey of each item (person), which starts at pickup and finishes at delivery. Some modifications to this theme include the usage of transfer points as in (Masson et al. 2014) and anticipation on expected return transports as in (Schilde et al. 2011). The latter adds a coordination aspect that relates to the subject of our challenge in the sense that the items being delivered need to be picked up again; however, their items (people) are not ‘reusable’ nor ‘exchangeable’.

Simultaneous pickups and deliveries are also central in the literature on routing within reverse logistics, see (Dethloff 2001). Researchers in this field tend to focus on the transportation of reusable packaging, which seemingly relates to our relocation of scarce items. However, the relocation— and hence reuse—is out of scope of these models: packaging is brought back to depots to be reused in subsequent, not yet planned, routes. Typically, no careful inventory of packages is kept since they are not perceived as scarce.

In general, routing papers assume the pickup and delivery orders to be a priori defined and they should be served on the day of planning.

One area where we do find the multiple day aspect and the choice of delivery day is in inventory routing problems as surveyed by Coelho et al. (2014). The main difference is that in inventory routing models the inventory is managed at the delivery location and typically estimated by the routing operator. Goods are not relocated, nor they are scarce. Inventory inside the vehicles is simple to manage since loading takes place at the depot only, whereas unloading takes place during the route. The

intrinsic difficulty of such models comes from the combination of demand forecasting (leading to inventory estimation) with routing and scheduling decisions. These scheduling and routing aspects are present in our problem; however, classical inventory routing models lack the relocation and reuse of goods.

Finally, we mention the research on relocation planning for bike sharing systems [see (Fishman 2016) for a survey]. In this problem, the goal is to transport bikes from locations where they tend to accumulate to those where they are sought. The decision on what level of bike inventory to maintain at every location greatly depends on estimations of flows between locations, which makes this problem highly stochastic and complex. This complexity is mentioned in, for example, (Schuijbroek et al. 2017), which notes that finding provably optimal solutions is practically intractable. Perhaps that is why this domain is quite rich in approaches, including variable neighborhood search heuristics in Rainer-Harbach et al. (2013), branch-and-cut algorithm in Raviv and Kolka (2013), cluster-first route-second heuristic in Schuijbroek et al. (2017), and simulation optimization in Jian et al. (2016). Furthermore, these models are sometimes extended with different aspects of the problem, such as combining staff-based vehicle redistribution and real-time price incentives for customers (Pfrommer et al. 2014).

It is interesting to mention that the repositioning by multiple vehicles addressed by Dror et al. (1998) was also in a context of vehicle sharing, in their case electric cars, which are indeed much more scarce than the bikes being shared.

We believe that the combination of decisions found in the problem of this challenge, which is a practical problem faced by some of ORTEC's customers, is quite unique and may lead to subsequent research. Additionally, the richness of the objective function contributes to the versatility and difficulty of this problem: emphasizing tool minimization leads to different methodologies being effective than emphasizing total distance. To support future research with benchmarking, the challenge site remains alive after the challenge has ended.

The problem, including the format of the instance and solution files, and the challenge rules are described by the challenge team (Gerhard Post, Daan Mocking, Jelke van Hoorn, Caroline Jagtenberg and Joaquim Gromicho) which can be found on the competition website. Note that this paper contains a recap of the problem description and the challenge rules.

The challenge problem is a simplification of a richer version found by ORTEC's clients, which includes among other features multiple resource capabilities, heterogeneous fleet, multiple depots, route synchronization, tight time-windows and adherence to working and driving time directives. Furthermore, the real problem as solved by ORTEC for its clients includes an additional phase which is not part of this challenge: the scheduling and routing of inspectors, who should visit the farms while the equipment is present. Each inspector has his or her own home base, skills and periods of availability, which makes the whole problem an even greater challenge. The problem instances used during the VeRoLog challenge can be downloaded from the competition website.

As a curiosity, we mention that a group of undergraduate Business Analytics students at the Vrije Universiteit in Amsterdam ran a preliminary version of the challenge composed of smaller and less restrictive instances, as a case study

during a course taught by Joaquim Gromicho. During this case study it became evident that there are at least two main ways to tackle the problem: route first, schedule second or schedule first, route second. Those that focus first on routing day by day and just schedule to meet restrictions were the first to obtain solutions to all instances, while those that develop a sophisticated scheduling of the visits prior to routing took longer to design and implement their algorithms, but reached higher solution quality.

The remainder of this paper is structured as follows. Section 2 provides a description of the tackled problem. Section 3 describes the algorithms that ended up second and third in the ranking of the challenge. Section 4 presents the results, and Sect. 5 describes the conclusions.

2 Problem description

The problem discussed in this paper consists in planning of deliveries and pickups of tools to customers at their requests to achieve objectives under the presence of several constraints.

The problem consists of a set C of customers, a set T of tool kinds, and a set R of tool requests. A request $r(n, t, c, d, w)$ asks for $n \in \mathbb{N}$ tools of one kind $t \in T$, that need to be present at customer $c \in C$ for a given number of consecutive days d. The delivery of the tools has to fall within a certain time window w given in full days. If a customer requires several kinds of tools, this means separate requests are made. Note that all requests are known at the moment the planning is made. The tools of the request have to be picked up by one vehicle the day after the request is completed, i.e., precisely $d + 1$ days after the tools were delivered.

Each problem instance has one depot location where all tools are located at the beginning and end of the planning horizon. A vehicle can load a tool at the depot and unload it at a customer. Alternatively, after the first day, a vehicle can also pick up a tool at customer c_1 and deliver it to customer c_2 without visiting the depot in between. Vehicles can also visit the depot multiple times per day, leaving tools and picking them up later for redistribution. To avoid the need for synchronization between the vehicles on the same day, only the vehicle that left the tool at the depot may pick it up again. Relaxing this constraint would force detailed arrival moments at the depot to be modeled to enable checking that tools already brought by a vehicle are available to be taken by another during the same day. If the tools cannot be exchanged between vehicles on the same day, the arrival and departure times of vehicles at the depot do not need to be synchronized among the vehicles. This restriction does not apply if the tool is being picked up on a later day. All vehicles must start and end their day at the depot. If a vehicle visits the depot during the day, the vehicle route consists of multiple tours.

Each tool kind has a certain size, and the available vehicles all have the same capacity with respect to the tool sizes. During any part of a route, the total size on board of a vehicle may not exceed its capacity. There is no maximum amount on the number of vehicles one can use (although vehicles are not free).

Every problem instance provides coordinates for each customer as well as a depot, allowing the participants to compute the Euclidean distance between any two locations. There is an upper bound on the distance that a vehicle can travel in 1 day.

2.1 Objectives

All requests must be satisfied, and the objective is to minimize a cost function that consists of four parts: (1) costs per distance traveled, (2) costs for using a vehicle for a day, (3) costs for using a vehicle at all, and (4) a cost per tool, depending on the tool kind. The latter was inspired by the real-life problem that this challenge originated from: the tools involved are in fact rather expensive, and hence it is worthwhile to investigate whether routes can be created which allow for fewer tools to be purchased. Each problem instance includes a definition for costs (1)–(4), which means that different problem instances emphasize different aspects of this problem. This makes it more challenging for the participants to come up with one algorithm that tackles all problem instances.

For the challenge rules, including how algorithms are evaluated, we refer the reader to Appendix A.

3 Competitors' algorithms

Search methodologies are at the core of decision support systems, particularly while dealing with computationally difficult optimization problems. The cutting-edge methods are often tailored for a specific problem domain by the experts in the area. Such systems are custom-made and, often, costly to build. When exact methods cannot be applied, practitioners and researchers resort to heuristics, which are 'rule of thumb' methods for solving a given optimization problem. There is a growing interest towards more general, cheaper and intelligent methods. Metaheuristics (Sörensen and Glover 2013) and hyper-heuristics (Burke et al. 2013) are such methodologies that automate the search process. This section presents the methods that won the runner-up and the second runner-up prizes in the VeRoLog Solver Challenge 2016–2017. The former method employs a hyper-heuristic technique and the latter applies an improved genetic algorithm metaheuristic.

3.1 A sequence-based selection hyper-heuristic (Team: akhe)

The main components of selection hyper-heuristics as identified in (Burke et al. 2013) are (1) *heuristic selection* which selects a low level heuristic from a predefined set of low level heuristics and applies it to a candidate solution at each decision point; and (2) *move acceptance* which decides whether to continue with the newly generated solution or the previous solution. A new field of hyper-heuristic methods embedding data science techniques has recently been developed (Asta and Özcan 2015). Experiments on a hyper-heuristic benchmark framework (Kheiri and Keedwell 2015), urban transit route design problem (Ahmed et al. 2019), wind farm

Transition

	llh_0	llh_1	llh_2	llh_3
llh_0	1	1	1	1
llh_1	1	1	1	1
llh_2	1	1	1	1
llh_3	1	1	1	1

Status

	add	*end*
llh_0	1	1
llh_1	1	1
llh_2	1	1
llh_3	1	1

Fig. 1 Initial score values of the two matrices for $n = 4$ low level heuristics

layout optimization problem (Wilson et al. 2018), high school timetabling problem (Kheiri and Keedwell 2017) and on water distribution optimization problem (Kheiri et al. 2015) have shown that applying a sequence of low level heuristics can potentially improve the quality of solutions more than those that simply select and apply a single low level heuristic.

3.1.1 Overall model

The competing method that took the second place uses a method that applies sequences of heuristics. To achieve this, each low level heuristic is associated with two probabilities: a transition probability to move to another low level heuristic including itself, and another to determine whether to terminate the sequence of low level heuristics at this point.

Let $[llh_0, llh_1, \ldots, llh_{n-1}]$ be the set of low level heuristics. A transition matrix (*Transition*) of size $n \times n$ stores scores for each of the n low level heuristics, from which we calculate the probabilities of moving from one low level heuristic to another (by normalizing the scores given in the matrix). We also define another matrix referred to as sequence status matrix (*Status*) of size $n \times 2$ which specifies scores for each of the n low level heuristics in one of two options: *add* and *end*.

Initially, elements in both matrices (*Transition* and *Status*) are assigned the value 1. Figure 1 shows the initial score values of the two matrices for $n = 4$ low level heuristics.

At first, a randomly selected low level heuristic (assume llh_2 is selected) is added to the sequence of low level heuristics. [SEQUENCE: llh_2]

The *Status* matrix is used to decide whether another low level heuristic will be selected and added to the sequence or the sequence will end at this point. To make one of these two choices, a roulette wheel selection method is applied. For llh_2, the probability of adding another low level heuristic is 1/2. Assume that the chosen status is *add*. [SEQUENCE: llh_2,]

The decision now is to add another low level heuristic to the sequence. This will be chosen by a selection procedure based on the roulette wheel selection strategy. In our example, the probability of selecting any low level heuristic, given that the recently added low level heuristic was llh_2, is 1/4. Assume that the chosen low level heuristic is llh_1. [SEQUENCE: llh_2, llh_1]

Transition

	llh_0	llh_1	llh_2	llh_3
llh_0	1	1	1	1
llh_1	1	1	1	1
llh_2	1	2	1	1
llh_3	1	1	1	1

Status

	add	*end*
llh_0	1	1
llh_1	1	2
llh_2	2	1
llh_3	1	1

Fig. 2 Updated score values of the two matrices

The *Status* matrix is used again to decide whether another low level heuristic will be selected and added to the sequence or the sequence will end at this point. For llh_1, which is the recently added low level heuristic, the probability of adding another low level heuristic to the sequence is 1/2. Assume that the chosen status is *end*. [SEQUENCE: llh_2, llh_1].

In this case the current sequence of low level heuristics ([llh_2, llh_1]) will be applied to the candidate solution in this given order to generate a new solution.

If the new solution improved over the best solution, the scores in both matrices for the relevant low level heuristics are increased by 1 as a reward. This is illustrated in Fig. 2. If the new solution does not improve the quality of the best solution in hand, then the scores in the matrices will not be updated. This way we only increase the chance of selecting the sequences that generate improved solutions.

We are now at llh_1, and we continue with the same strategy to construct and apply the next sequence of heuristics using the updated scores.

The move acceptance method used in this work is Record-to-Record Travel (RRT) move acceptance criterion (Dueck 1993). The idea of RRT is based on the simple notion that any new solution, which is not much worse than the best solution recorded, is accepted. A candidate solution is in the form of a three-dimensional array (days × routes × visits).

Note that the quality of a given solution is evaluated using the main objective to be minimized and an estimated (secondary) objective depending on which cost type (described in Sect. 2.1) of a given problem instance is set highest. As an example, if the main objective is to minimize the number of vehicles, then the algorithm will locate the day that has the most number of vehicles (routes) running and the secondary objective becomes the trip distance of the route that has the least total distance on that day. Similarly for the number of used vehicles per day, the algorithm attempts to minimize the number of vehicles used per day as the main objective, and the trip distance of the route that has the least total distance as the secondary objective.

The organizers of the challenge provided a set of feasible instances, and confirmed that a feasible solution to the problem can be achieved by selecting for each visit (delivery or pickup) one vehicle to carry out only this visit. Following this, we developed a greedy algorithm to construct an initial feasible solution.

However, the implemented simple greedy algorithm, which runs in milliseconds, often yields a poor quality solution requiring further enhancement.

3.1.2 Low level heuristics

The sequence-based selection hyper-heuristic approach in this work controls a set of 22 low level heuristics to improve the quality of an initially generated solution. The low level heuristics are grouped into the following 6 categories: move, swap, reverse, add, delete and ruin and recreate.

3.1.3 Move low level heuristics

- LLH0: Moves a visit (delivery, pickup or depot) into a new location inside a route (Fig. 3a).
- LLH1: Selects two random routes, same day, and a random position on each route. The visit in the first position is moved into the second position on the second route (Fig. 3c).
- LLH2: Selects two random routes from different days and a random position on each route. The visit in the first position is moved into the second position on the second route. Corresponding visits (pickup or delivery) will be moved to satisfy the time window constraint.
- LLH3: Moves a block of visits, that is a set of consecutive visits, into a new location inside a route.
- LLH4: Moves a block of visits into a randomly selected location from another route in the same day (Fig. 3e).
- LLH5: Moves a block of visits into a randomly selected location from another route in different day. Corresponding visits will be moved to satisfy the time window constraint.
- LLH6: Moves a tour, that is visited between two depots, from a route into another route in the same day.
- LLH7: Moves a tour from a route into another route in different day. Corresponding visits will be moved to satisfy the time window constraint.

3.1.4 Swap low level heuristics

- LLH8: Selects a random route and two random positions and swaps the two visits in these positions (Fig. 3b).
- LLH9: Selects two random routes from a randomly selected day, and a random position on each route and swaps the visits in these positions (Fig. 3d).
- LLH10: Selects two random routes from two different days, and a random position on each route and swaps the visits in these positions. Corresponding visits will be moved to satisfy the time window constraint.
- LLH11: Exchanges block of visits inside a route.
- LLH12: Exchanges block of visits between two routes both from the same day (Fig. 3f).

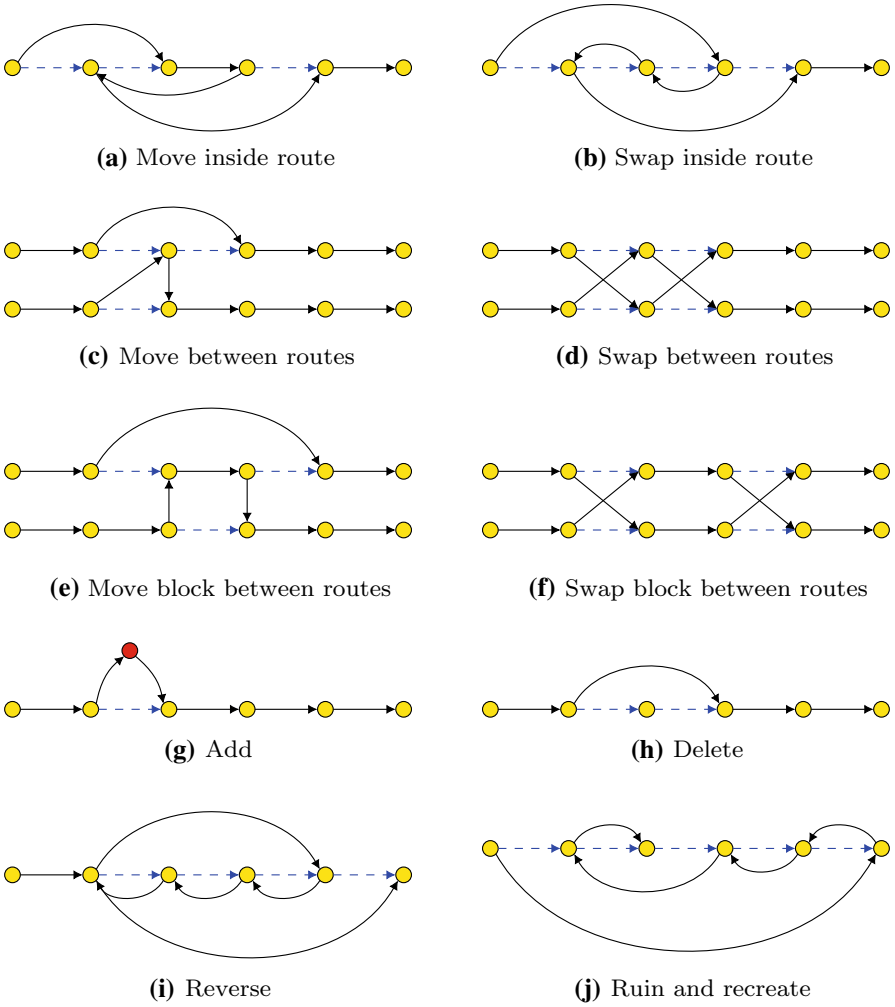


Fig. 3 Straight arcs are visits in the route, dashed arcs are visits removed after applying the heuristic, curved arcs are visits added after applying the heuristic

- LLH13: Exchanges block of visits between two routes from different days. Corresponding visits will be moved to satisfy the time window constraint.
- LLH14: Exchanges two tours in a randomly selected route.
- LLH15: Exchanges two tours in two different routes in a randomly selected day.
- LLH16: Exchanges two tours in two different routes from different days. Corresponding visits will be moved to satisfy the time window constraint.

3.1.5 Reverse low level heuristics

– LLH17: Consists in a chronological reversal of a block of visits in a randomly selected route (Fig. 3i).

3.1.6 Add low level heuristics

– LLH18: Selects a random route and a random position in this route and adds a depot visit into this position (Fig. 3g).

3.1.7 Delete low level heuristics

– LLH19: Deletes a depot visit (Fig. 3h).

3.1.8 Ruin and recreate low level heuristics

– LLH20: Destructs a randomly selected rout generating a partial solution and then reconstructs a complete solution at random (Fig. 3j).
– LLH21: Same as LLH20 but destructs/reconstructs several routes from a randomly selected day.

3.1.9 Additional remarks and conclusions

Although, the ultimate goal of the development of hyper-heuristic methods is to increase the level of generality, by offering methods that have the ability to work on a wide range of optimization problems, still it would be interesting to know the position of hyper-heuristics with respect to other problem-specific solution methods in a particular optimization problem while still being general. In this work, a sequence-based selection hyper-heuristic has been developed which aims to intelligently and effectively control the application of sequences of heuristics as opposed to simple selection of single heuristic. The method effectively exploits the features of the problems on the fly as indicated in Kheiri and Keedwell (2015). This is a viable approach considering that at different points during the search, different sequences of heuristics may be performing well.

Preliminary experiments did indicate that large low level heuristics at tour (or block of visits) level that tend to move tours around could lead to better results. Of course better understanding of this effect requires further work and much more exploration. Final results of the challenge suggest that the low level heuristics would need significant adjustment to handle the problem more effectively. Interesting future work might well try to explore features of instances that are correlated with the different objectives defined in Sect. 2.1.

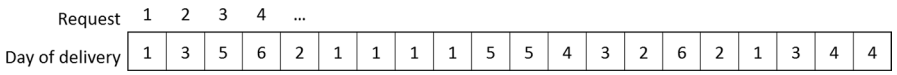


Fig. 4 An example of a genome sequence for the genetic algorithm

3.2 A genetic algorithm metaheuristic (Team: ADDM)

The proposed algorithm decomposes the problem into two independent parts. The first part finds a schedule for the delivery day of each request. The second part finds the routes of the vehicles for each day according to the predetermined schedule.

3.2.1 Scheduling of delivery days

For the scheduling, a genetic algorithm (GA) is used. For a pedagogical introduction on GA see Wall (1996). Hart et al. (2005) provide a review on evolutionary scheduling literature. The genome sequence represents the day of delivery for each request. Therefore, the sequence has a length equal to the number of requests. The day specified for each genome has to be within the delivery time window for the request. Since every request needs the tools for a specific amount of days, the corresponding pickup days can be easily determined. As far as the scheduling is concerned, all necessary information is stored in this sequence. Figure 4 shows an example of a genome sequence.

3.2.2 Building a population

For building an initial population, two different approaches are used: a greedy heuristic and a random schedule generator.

The greedy heuristic picks a request and assigns a delivery day that increases costs the least for the overall schedule. Here, a cost estimation is used based on the number of required vehicles and tools. If the resulting sequence is feasible, it is added to the initial population, otherwise the heuristic restarted. Feasibility is determined by checking if the required resources are not exceeding the available limit. To avoid a deterministic heuristic, the order of requests is chosen randomly.

The random schedule generator assigns each request a random day for delivery as long as it is within the allotted time windows. Therefore, the day for the sequence is selected between the first and last possible day for delivery. If the sequence is determined to be feasible, it is added to the initial population.

The size of the population is adapted to the size of the instance, i.e., the number of requests. Depending on the problem instance, we choose between 70 and 300 individuals. Both the greedy heuristic and the random schedule generator initially generate a pool of individuals two times the population size. The better half is kept, the worse half discarded.

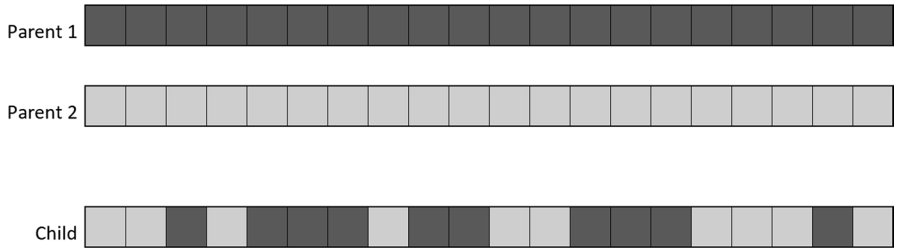


Fig. 5 An example of a uniform crossover

3.2.3 Selection, genetic operators and mutation

The parents for reproduction are chosen out of the whole population with decreasing probability the higher the cost of an individual: first, we sort the population by the score of each individual. The population is then separated into three parts: the top half and the next two quarters. A parent is then chosen with 1/2 probability out of the top half, 1/3 probability out of the less-than-average quarter and 1/6 probability out of the worst quarter. The same procedure is used for the second parent. This way of selecting parents guarantees that individuals with low cost are chosen most of the time, but also allows for less than average candidates to take part in the reproduction step. As a consequence, it takes longer for populations to converge to local optima. Clearly, constructing a selection method is not a rigorous task and there is a lot of freedom in the exact details of how to choose individuals. After some testing we settled on the above described selection method due to its simplicity and because it achieved acceptable results.

For reproduction, a uniform crossover operator is used. Each gene of the child sequence has an equal probability of being selected from one of the parents. Figure 5 shows an example of a uniform crossover.

After the crossover, some mutation might occur: we mutate up to 10 random requests, each having a probability of 1/2 to mutate. The mutation shifts a request to a different delivery day within the available time windows.

The genetic algorithm runs up to 400 generations or until the population 'converges'. We define a population to be converged, if the new generation contains more than 80 per cent of identical individuals compared to the last generation.

3.2.4 Routing and schedule evaluation

To get the exact score of a schedule, the routing for the whole planning horizon has to be computed. The routing algorithm has to run very often, because our genetic approach relies on evaluating the score of a large number of individual schedules in a short amount of time. In particular, the genetic algorithm might encounter the same individual more than once during its run. Therefore, the score evaluation of individual schedules represents the main performance bottleneck of the solver.

These problems can be solved by choosing a deterministic routing algorithm, which provides each schedule with a unique set of routes. As a result, two identical schedules have the same routing and the same score. Once the routes for a schedule are calculated, the score and a hash value of the schedule can be cached in a lookup table. The lookup table provides an easy and fast way of checking if a particular schedule has been encountered before and obtaining its score without running the routing algorithm again. Furthermore, memory use is also greatly improved because it is not necessary to store the routes of each schedule in the population. In the rare event that the exact routes are needed (for example when writing the best known solution to a file), we simply run the deterministic routing algorithm again and obtain the same result as before.

The routing algorithm takes a schedule as input and tries to solve the routing problem for each day individually. Since each vehicle can return to the depot multiple times within a day, each vehicle route can consist of multiple tours. The algorithm is comprised of four stages:

1. A modified parallel savings algorithm (Clarke and Wright 1964; Rand 2009) with $s_{ij} = k(s_{i0} + s_{j0} - s_{ij})$ where $k > 1$, if i is a pickup and j a delivery of the same tool. Otherwise $k = 1$. Since the problem has the quite unique characteristic of allowing tools to be passed on from one customer to the next, we favor savings where this is the case to minimize tool use. The algorithm is initialized with single requests tours. Each initial tour starts at the depot, fulfills a single pickup or delivery request and then returns to the depot. The savings algorithm is based on merging pairs of tours into increasingly longer tours. For every merge we determine if the capacity constraint is fulfilled by iterating through the arcs where each pickup reduces the remaining capacity and each delivery increases the remaining capacity of the vehicle. Additionally, maximum distance constraints are checked by summing over arc lengths. If constraints are violated, the savings pair is skipped and the next pair is considered.
2. A 2-opt heuristic (Croes 1958) using best improvement where all combinations of arc pairs are selected and the sub-route between the arcs is reversed. Every time tours are modified we check for capacity and distance constraints.
3. A 3-opt heuristic (Lin and Kernighan 1973) using best improvement where all combinations of arc triplets are selected. As with the 2-opt heuristic, we check for constraint violations.
4. A best fit decreasing bin packing heuristic (Johnson 1973; Martello and Toth 1990) to combine tours into vehicles routes with the goal to reduce the amount of vehicles in use. When packing tours into routes the capacity constraint is always fulfilled, as each tour starts and ends at the depot. However, the distance constraint must still be checked.

Stages 2–4 use simple cost estimations with different weight parameters for tool use, vehicle number and route length. Tools of the same type are assumed to always be passed on from a customer to the next to minimize the number of tools required from the depot. Depending on the parameter k and the weights, the

results of the routing algorithm can vary dramatically for a given schedule. The final version of our solver uses 5 different parameter sets, which put emphasis on particular aspects of the cost function. During the initial stage of the solver, we choose the parameter set which gives the best average score of the initial population.

3.2.5 Post-optimization for the routing

Once the genetic algorithm finishes (either due to convergence of the population or due to reaching the time limit) and a candidate schedule has been found, more computationally intensive improvement heuristics can be used to further improve the routes that were determined by the deterministic routing algorithm. Our solver sets aside 20 per cent of the available runtime for this post optimization stage. We use variable neighborhood descent (VND) (Hansen and Mladenović 1999) and the following neighborhoods:

1. *Move* Moves a random node (delivery or pickup) to another position and/or tour and/or vehicle route.
2. *Swap* Swaps two random nodes from randomly selected tours and vehicles.
3. *Tour move* Moves a random tour to another vehicle.
4. *Tour swap* Swaps two random tours from randomly selected vehicles.

The VND terminates after n iterations without improvement in each neighborhood or until available runtime is reached.

3.2.6 Additional remarks and conclusions

The decomposition approach (i.e., 'scheduling first, routing second') was chosen to simplify the problem and accommodate the genetic algorithm. In particular, it naturally leads to an appropriate solution representation of individuals in terms of schedules. The genetic algorithm proves to be a powerful metaheuristic for a problem like this, where sub-optimal, but feasible solutions can easily be found. The deterministic routing and schedule evaluation ensures that routing calculations are not needlessly repeated for identical schedules. This results in a great acceleration of the evaluation of the later generations. Such optimization is especially important for the (time) resource restricted challenge. The parameter values that account for the vastly different costs of each instance and the general parameters of the genetic algorithm were set in a trial-and-error fashion. In the end, participation in the all-time-best challenge with strong competition from other teams helped us choose the particular values used in the solver.

Table 2 The characteristics of the all-time-best instances and the cost of best obtained solutions

Instance	Customers	Tools	Capacity	Max distance	Best solution	Obtained by
VeRoLog_r100d5_1	100	3	50	20,000	1,552,435,049	mjg
VeRoLog_r100d5_2	100	2	40	20,000	996,709,544	mjg
VeRoLog_r100d5_3	100	4	40	20,000	119,957,689	mjg
VeRoLog_r100d5_4	99	5	30	15,000	1,359,088,350	Success
VeRoLog_r100d5_5	100	2	45	16,000	300,125,049,016	mjg
VeRoLog_r100d10_1	100	3	50	16,000	1,313,786,538	mjg
VeRoLog_r100d10_2	100	5	35	20,000	1,555,438,898	mjg
VeRoLog_r100d10_3	100	2	40	17,000	155,316,178	mjg
VeRoLog_r100d10_4	100	4	45	15,000	1,114,888,217	Success
VeRoLog_r100d10_5	98	3	35	16,000	43,871,262,004	Success
VeRoLog_r500d15_1	494	4	35	15,000	3,256,089,143	mjg
VeRoLog_r500d15_2	491	3	35	20,000	3,800,352,751	mjg
VeRoLog_r500d15_3	490	2	45	15,000	402,839,699	mjg
VeRoLog_r500d15_4	487	3	35	17,000	2,807,990,462	mjg
VeRoLog_r500d15_5	488	3	45	15,000	251,378,880,010	mjg
VeRoLog_r1000d25_1	950	2	45	15,000	7,004,087,706	mjg
VeRoLog_r1000d25_2	943	4	35	16,000	6,486,405,100	mjg
VeRoLog_r1000d25_3	944	3	50	17,000	207,087,083	mjg
VeRoLog_r1000d25_4	949	4	30	17000	5,598,405,178	mjg
VeRoLog_r1000d25_5	951	3	50	16,000	161,446,120,006	mjg
VeRoLog_r1000d30_1	940	5	40	15,000	5,220,068,560	mjg
VeRoLog_r1000d30_2	942	4	35	15,000	5,181,409,255	mjg
VeRoLog_r1000d30_3	945	4	40	20,000	187,843,389	mjg
VeRoLog_r1000d30_4	930	4	40	16,000	4,562,837,156	mjg
VeRoLog_r1000d30_5	948	3	35	16,000	257,497,762,007	mjg

4 Competition results

4.1 All-time-best challenge

Table 2 shows the characteristics of the all-time-best instances and the cost of best obtained solutions. Instances are named using two numbers with the prefixes r for requests and d for days. The instances range from 100 to 1000 customers and from 5 to 30 days. The last number indicates which cost type is set highest, with 1: tool cost followed by vehicle cost, 2: tool cost followed by vehicle day cost, 3: vehicle cost, 4: vehicle day cost and 5: distance cost. For example, the Instance VeRoLog_r100d5_1 has 100 requests over 5 days, with tools having the highest cost, and vehicle has the second highest cost. The instances also have between 2 and 5 different tool kinds that have to be distributed.

Table 3 The costs of the best five solutions achieved by the competitors and the date the solutions were submitted during the all-time-best challenge

Instance	First team	Second team	Third team	Fourth team	Fifth team
VeRoLog_r100d5_1					
Team	mjg	Success	akhe	ADDM	Sunbeams
Cost	1,552,435,049	1,552,437,090	1,552,472,997	1,552,667,702	1,552,674,956
Date	10/04/2017	12/06/2017	22/03/2017	24/03/2017	13/05/2017
VeRoLog_r100d5_2					
Team	mjg	akhe	Sunbeams	ADDM	Success
Cost	996,709,544	997,125,464	997,131,853	997,536,075	997,975,026
Date	17/04/2017	22/03/2017	19/05/2017	24/03/2017	13/03/2017
VeRoLog_r100d5_3					
Team	mjg	Success	ADDM	Sunbeams	EquipoMLS
Cost	119,957,689	119,999,604	120,174,619	125,484,960	141,027,538
Date	17/04/2017	15/06/2017	08/03/2017	13/05/2017	30/05/2017
VeRoLog_r100d5_4					
Team	Success	ADDM	mjg	Sunbeams	EquipoMLS
Cost	1,359,088,350	1,359,171,004	1,388,155,986	1,506,904,422	1,599,316,230
Date	12/03/2017	10/03/2017	13/04/2017	03/05/2017	30/05/2017
VeRoLog_r100d5_5					
Team	mjg	Success	ADDM	Sunbeams	EquipoMLS
Cost	300,125,049,016	302,122,548,017	302,145,047,015	312,987,048,016	324,679,054,017
Date	17/04/2017	15/06/2017	02/03/2017	03/05/2017	26/05/2017
VeRoLog_r100d10_1					
Team	mjg	Success	Sunbeams	ADDM	VeRoLog050
Cost	1,313,786,538	1,313,800,238	1,313,843,042	1,383,818,456	1,404,313,470
Date	05/03/2017	17/06/2017	19/05/2017	09/03/2017	03/03/2017

Table 3 (continued)

Instance	First team	Second team	Third team	Fourth team	Fifth team
VeRoLog_r100d10_2					
Team	mjg	Success	Sunbeams	VeRoLog050	ADDM
Cost	1,555,438,898	1,555,866,637	1,556,076,294	1,608,781,502	1,616,279,307
Date	14/02/2017	15/04/2017	19/05/2017	02/03/2017	09/03/2017
VeRoLog_r100d10_3					
Team	mjg	Success	ADDM	Sunbeams	TeamTau2017
Cost	155,316,178	155,451,452	155,606,071	155,859,870	156,154,622
Date	13/04/2017	30/03/2017	09/03/2017	19/04/2017	02/06/2017
VeRoLog_r100d10_4					
Team	Success	mjg	ADDM	TeamTau2017	goc-ar
Cost	1,114,888,217	1,152,790,895	1,192,096,909	1,350,637,725	1,351,320,617
Date	09/05/2017	28/04/2017	06/03/2017	02/06/2017	09/12/2016
VeRoLog_r100d10_5					
Team	Success	mjg	ADDM	Sunbeams	goc-ar
Cost	43,871,262,004	43,901,664,004	45,036,866,004	52,393,174,005	52,474,576,006
Date	23/04/2017	13/04/2017	10/03/2017	28/04/2017	05/12/2016
VeRoLog_r500d15_1					
Team	mjg	Sunbeams	Success	VeRoLog050	ADDM
Cost	3,256,089,143	3,288,333,346	3,346,228,685	3,487,816,461	3,506,242,192
Date	03/05/2017	09/06/2017	20/04/2017	03/03/2017	06/03/2017
VeRoLog_r500d15_2					
Team	mjg	Sunbeams	Success	ADDM	VeRoLog050
Cost	3,800,352,751	3,811,482,643	3,888,199,515	3,951,378,786	4,052,471,132
Date	03/05/2017	14/06/2017	13/04/2017	07/03/2017	03/03/2017

Table 3 (continued)

Instance	First team	Second team	Third team	Fourth team	Fifth team
VeRoLog_r500d15_3					
Team	mjg	ADDM	goc-ar	Success	TeamTau2017
Cost	402,839,699	454,318,339	511,857,243	558,690,479	607,429,601
Date	03/05/2017	07/03/2017	29/12/2016	02/04/2017	02/06/2017
VeRoLog_r500d15_4					
Team	mjg	ADDM	goc-ar	TeamTau2017	EquipoMLS
Cost	2,807,990,462	2,892,747,784	3,652,534,665	3,732,841,802	4,051,526,171
Date	03/05/2017	02/03/2017	31/12/2016	02/06/2017	01/06/2017
VeRoLog_r500d15_5					
Team	mjg	ADDM	goc-ar	TeamTau2017	VeRoLog050
Cost	251,378,880,010	259,677,305,009	306,490,210,012	312,221,625,011	330,726,465,011
Date	03/05/2017	02/03/2017	17/12/2016	02/06/2017	03/03/2017
VeRoLog_r1000d25_1					
Team	mjg	Sunbeams	akhe	ADDM	VeRoLog050
Cost	7,004,087,706	7,166,663,786	7,469,954,246	7,494,227,661	7,546,340,291
Date	03/05/2017	05/06/2017	14/04/2017	12/03/2017	02/03/2017
VeRoLog_r1000d25_2					
Team	mjg	Sunbeams	NSA	ADDM	VeRoLog050
Cost	6,486,405,100	6,811,349,274	7,177,503,250	7,229,613,701	7,232,820,503
Date	03/05/2017	05/06/2017	12/05/2017	06/03/2017	02/03/2017
VeRoLog_r1000d25_3					
Team	mjg	ADDM	goc-ar	EquipoMLS	TeamTau2017
Cost	207,087,083	239,642,618	246,934,382	298,214,337	308,695,312
Date	03/05/2017	10/03/2017	09/12/2016	03/06/2017	02/06/2017

Table 3 (continued)

Instance	First team	Second team	Third team	Fourth team	Fifth team
VeRoLog_r1000d25_4					
Team	mjg	ADDM	goc-ar	TeamTau2017	EquipoMLS
Cost	5,598,405,178	6,205,511,061	7,553,650,631	8,204,868,671	8,857,593,182
Date	03/05/2017	03/03/2017	09/12/2016	02/06/2017	08/06/2017
VeRoLog_r1000d25_5					
Team	mjg	ADDM	goc-ar	TeamTau2017	EquipoMLS
Cost	161,446,120,006	174,254,946,006	196,592,076,007	218,038,116,009	234,970,338,008
Date	03/05/2017	03/03/2017	09/12/2016	02/06/2017	08/06/2017
VeRoLog_r1000d30_1					
Team	mjg	Sunbeams	Success	ADDM	Eva
Cost	5,220,068,560	5,545,670,163	5,572,741,083	5,919,953,818	6,029,750,153
Date	03/05/2017	05/06/2017	14/04/2017	06/03/2017	16/05/2017
VeRoLog_r1000d30_2					
Team	mjg	Sunbeams	VeRoLog050	Success	ADDM
Cost	5,181,409,255	5,363,167,525	5,696,835,520	5,703,651,327	5,736,716,754
Date	03/05/2017	10/06/2017	02/03/2017	13/04/2017	06/03/2017
VeRoLog_r1000d30_3					
Team	mjg	ADDM	EquipoMLS	TeamTau2017	goc-ar
Cost	187,843,389	219,104,500	282,173,474	284,664,999	285,062,660
Date	03/05/2017	12/03/2017	29/04/2017	02/06/2017	05/12/2016
VeRoLog_r1000d30_4					
Team	mjg	ADDM	goc-ar	TeamTau2017	Success
Cost	4,562,837,156	5,139,734,143	6,157,957,481	6,367,255,071	6,644,215,518
Date	03/05/2017	02/03/2017	09/12/2016	02/06/2017	12/06/2017

Table 3 (continued)

Instance	First team	Second team	Third team	Fourth team	Fifth team
VeRoLog_r1000d30_5					
Team	mjg	ADDM	goc-ar	Success	TeamTau2017
Cost	257,497,762,007	287,103,606,008	353,163,918,012	367,894,412,012	380,128,952,013
Date	03/05/2017	03/03/2017	31/12/2016	12/06/2017	02/06/2017

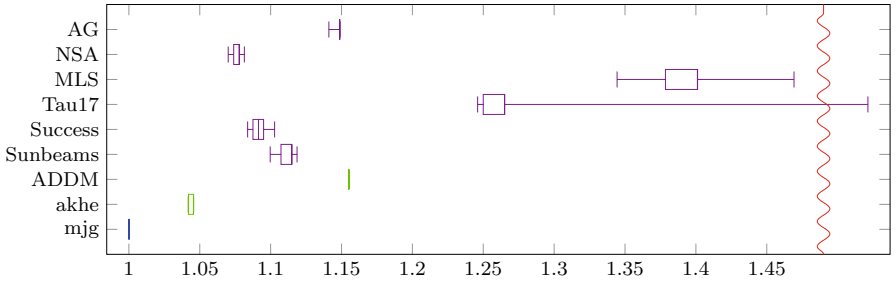


Fig. 6 VeRoLog_late_r1000d25_1

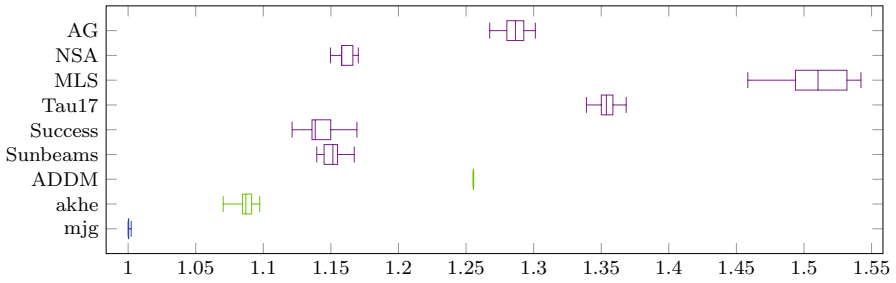


Fig. 7 VeRoLog_late_r1000d25_2

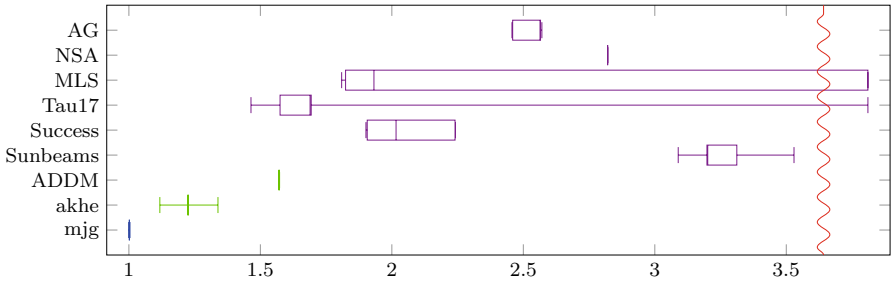


Fig. 8 VeRoLog_late_r1000d25_3

While team *mjg*–who won the finals—is frequently at the top of the ranking, there is an example where this team is outperformed[6] by team *ADDM*, who won third prize in the finals. This happens, for the problem instance VeRoLog_r100d5_4 (see Table 3), consisting of 100 requests and a 5 day planning horizon, where *ADDM* and *mjg* provided the second and third best solution respectively. For the other instances where *mjg* did not provide the best solution it provided the second best solution. It is still possible to upload solutions to the all-time-best challenge.

[6] At the time of writing this article, after the all-time-best challenge has ended.

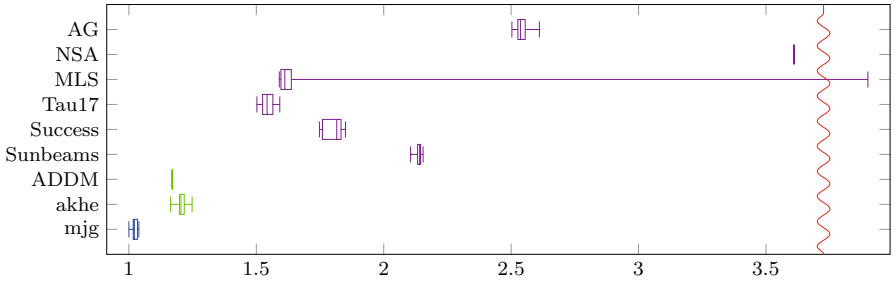


Fig. 9 VeRoLog_late_r1000d25_4

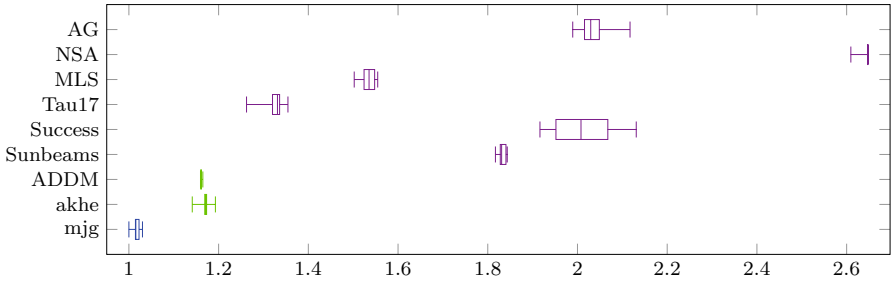


Fig. 10 VeRoLog_late_r1000d25_5

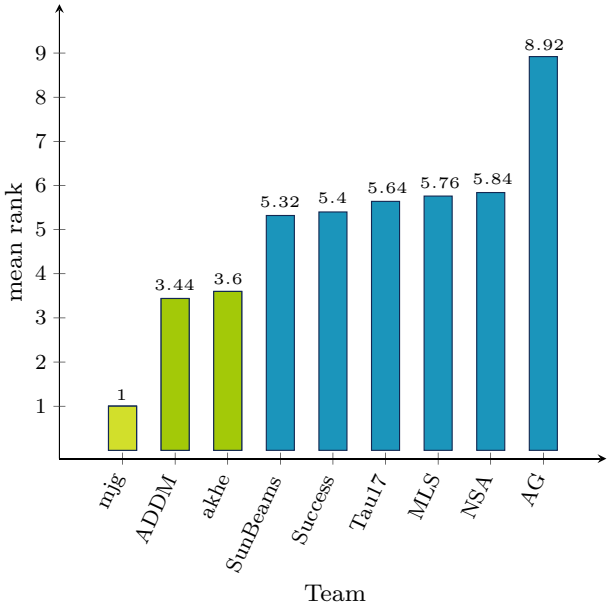


Fig. 11 Mean rank per team, as computed according to the challenge rules. These results are based on the submissions of the restricted resources challenge (late instances). The leftmost three bars correspond to the teams that reached the finale

4.2 Restricted resources challenge

Based on the challenge ranking system, the organizers selected potential finalists, and verified that their reported results could have realistically been produced by their submitted algorithms. This was done by running the algorithms on the same ORTEC-late instances and random seeds. Eventually, three participants were selected as finalists for the second part of the challenge.

Figures 6, 7, 8, 9, 10 show the performance variation of all competing methods on all five versions of VeRoLog_late_r1000d25 dataset. mjg achieved the best results in all instances. akhe performs better than ADDM on the first three versions, but ADDM found slightly better results compared to akhe on the last two versions (i.e., when vehicle day cost and distance cost are highly penalized, respectively). The same can be observed with the other instances.

Figure 11 shows the mean rank of the teams that submitted in the restricted resources challenge (on the so-called "ORTEC late instances"). The figure shows a large gap between the third and fourth place, which lead to the organizer's decision to allow precisely three participants in the finale.

Table 4 shows the characteristics of the hidden instances and the cost of the current best-known solutions. As before, the last integer in the instance name denotes the type of most penalized cost as described in Sect. 4.1.

Table 5 summarizes the results. We performed Mann–Whitney–Wilcoxon test (Kruskal 1957; Fagerland and Sandvik 2009) with a 95% confidence level to compare pairwise performance variations of two given competing methods statistically. The following notations are used: Given competing method A_1 versus competing method A_2, (1) $A_1 > (<) A_2$ denotes that A_1 (A_2) is better than A_2 (A_1) and this performance variance is statistically significant, (2) $A_1 \simeq A_2$ indicates that there is no statistically significant difference between A_1 and A_2.

Overall, it is clear that there is an unambiguous hierarchy: mjg (mean rank 1.02) performs better than akhe (rank 2.16) and akhe performs better than ADDM (rank 2.82). However, there are some exceptions to this and we focus on some differences between the second and third place. For the instances of type 4 (highest cost are of the type vehicle day) and type 5 (highest cost are of the type distance) the difference in performance between akhe and ADDM is less significant. This can also be observed in the late instances submitted by the participants. This suggests that the submitted solvers performed in a stable and consistent manner. With regards to instances with high tool cost (represented by type 1 and 2), akhe performed better than ADDM. One particular characteristic of the challenge problem is that tool cost can be avoided if tools are not directly delivered to a customer from the depot but transferred between customers instead. The results of the late and hidden instances indicate that ADDMs solver prioritizes distance cost over tool cost and, therefore, generally performs worse in such cases.

Table 4 The characteristics of the hidden instances and the cost of best obtained solutions

Instance	Customers	Tools	Capacity	Max distance	Best known
VeRoLog_hidden_r500d10_1	485	4	50	17,000	8365884102
VeRoLog_hidden_r500d10_2	492	2	35	16,000	5770640038
VeRoLog_hidden_r500d10_3	491	3	40	20,000	744495928
VeRoLog_hidden_r500d10_4	489	5	40	20,000	3774764460
VeRoLog_hidden_r500d10_5	492	3	40	20,000	166281961014
VeRoLog_hidden_r500d15_1	490	4	35	16,000	2696765455
VeRoLog_hidden_r500d15_2	491	3	45	15,000	5295158689
VeRoLog_hidden_r500d15_3	486	5	50	15,000	216780767
VeRoLog_hidden_r500d15_4	492	3	35	17,000	2019397633
VeRoLog_hidden_r500d15_5	493	2	50	16,000	153864525012
VeRoLog_hidden_r1000d20_1	950	4	30	17,000	8176085650
VeRoLog_hidden_r1000d20_2	950	3	50	20,000	4487557411
VeRoLog_hidden_r1000d20_3	942	5	50	16,000	452935897
VeRoLog_hidden_r1000d20_4	946	5	35	16,000	1816133429
VeRoLog_hidden_r1000d20_5	950	3	35	17,000	673881241010
VeRoLog_hidden_r1000d25_1	947	2	40	15,000	2476429490
VeRoLog_hidden_r1000d25_2	961	4	40	16,000	5953632945
VeRoLog_hidden_r1000d25_3	952	4	40	16,000	176042639
VeRoLog_hidden_r1000d25_4	952	2	35	16,000	7253854432
VeRoLog_hidden_r1000d25_5	952	2	45	20,000	445754474005
VeRoLog_hidden_r1500d30_1	1379	4	40	17,000	6446878020
VeRoLog_hidden_r1500d30_2	1372	5	40	16,000	6483597038
VeRoLog_hidden_r1500d30_3	1373	2	45	20,000	115134924
VeRoLog_hidden_r1500d30_4	1374	3	40	16,000	10599076534
VeRoLog_hidden_r1500d30_5	1382	4	50	20,000	1138250720007
VeRoLog_hidden_r1500d40_1	1375	3	40	15,000	3598635808
VeRoLog_hidden_r1500d40_2	1370	5	40	20,000	6035110304
VeRoLog_hidden_r1500d40_3	1367	3	50	16,000	253836767
VeRoLog_hidden_r1500d40_4	1373	4	35	15,000	2670813271
VeRoLog_hidden_r1500d40_5	1385	2	40	15,000	1610263788013
VeRoLog_hidden_r2000d50_1	1785	2	45	16,000	5029970838
VeRoLog_hidden_r2000d50_2	1785	5	30	16,000	4290045748
VeRoLog_hidden_r2000d50_3	1791	5	40	16,000	323926260
VeRoLog_hidden_r2000d50_4	1792	4	35	16,000	18321969466
VeRoLog_hidden_r2000d50_5	1777	4	40	16,000	1047956765006
VeRoLog_hidden_r2000d65_1	1776	2	40	16,000	3989738288
VeRoLog_hidden_r2000d65_2	1773	4	50	16,000	2457293730
VeRoLog_hidden_r2000d65_3	1782	5	40	16,000	195384322
VeRoLog_hidden_r2000d65_4	1767	3	35	17,000	15891623281
VeRoLog_hidden_r2000d65_5	1799	5	45	16,000	1370590324007
VeRoLog_hidden_r2500d70_1	2166	3	45	16,000	4751728213
VeRoLog_hidden_r2500d70_2	2164	5	35	16,000	5599542429

Table 4 (continued)

Instance	Customers	Tools	Capacity	Max distance	Best known
VeRoLog_hidden_r2500d70_3	2181	2	35	15,000	354579316
VeRoLog_hidden_r2500d70_4	2170	4	50	20,000	14479189673
VeRoLog_hidden_r2500d70_5	2140	5	40	17,000	1177525804008
VeRoLog_hidden_r2500d75_1	2160	4	40	15,000	5591086150
VeRoLog_hidden_r2500d75_2	2160	3	35	16,000	3520404346
VeRoLog_hidden_r2500d75_3	2147	5	50	15,000	300634318
VeRoLog_hidden_r2500d75_4	2185	3	45	17,000	15283897582
VeRoLog_hidden_r2500d75_5	2192	2	50	17,000	1625596820006

4.3 Convergence comparison

In this section, we compare akhe and ADDM algorithms on all five versions of VeRoLog_late_r1000d25 dataset. Our experiments are performed on Intel(R) Core(TM) i7-6500U CPU with a 2.50 GHz, 2.60 GHz and 8.00 GB of RAM. The convergence curves of the two algorithms on the selected instances is illustrated in Fig. 12. In all the cases, the hyper-heuristic method (akhe) improves the quality of the candidate solutions at the beginning of the search process rapidly, and then the process slows down when reaching the local optimum. The employment of the sequence-based strategy seems to lead the search to jump from local optima in some cases (e.g., VeRoLog_late_r1000d25_5), allowing further improvement to the quality of the solutions. Note that the plotted objective values for ADDM start slightly later than akhe. This is due to the initial phase of the solver: before starting the genetic algorithm, the initial population is evaluated using the routing algorithm with different parameter sets. Each parameter set is geared towards different cost priorities. The parameter set with the best average score is then chosen for the rest of the evolution. As this takes up some time, the objective value curves start only once this phase is completed.

5 Conclusion

In this paper, two heuristic algorithms were proposed to solve a vehicle routing problem with inter-route and intra-route challenges. This problem was the topic of a recent competition, referred to as VeRoLog Solver Challenge 2016–2017. It is based on a real-life problem of a cattle improvement company that combines routing, scheduling and inventory aspects. Instances differed, apart from size, in cost penalties, making the problem potentially relevant from a multi-objective point of view. 28 teams participated worldwide in the all-time-best challenge that ran for 8 months and 9 teams participated in a restricted resources challenge.

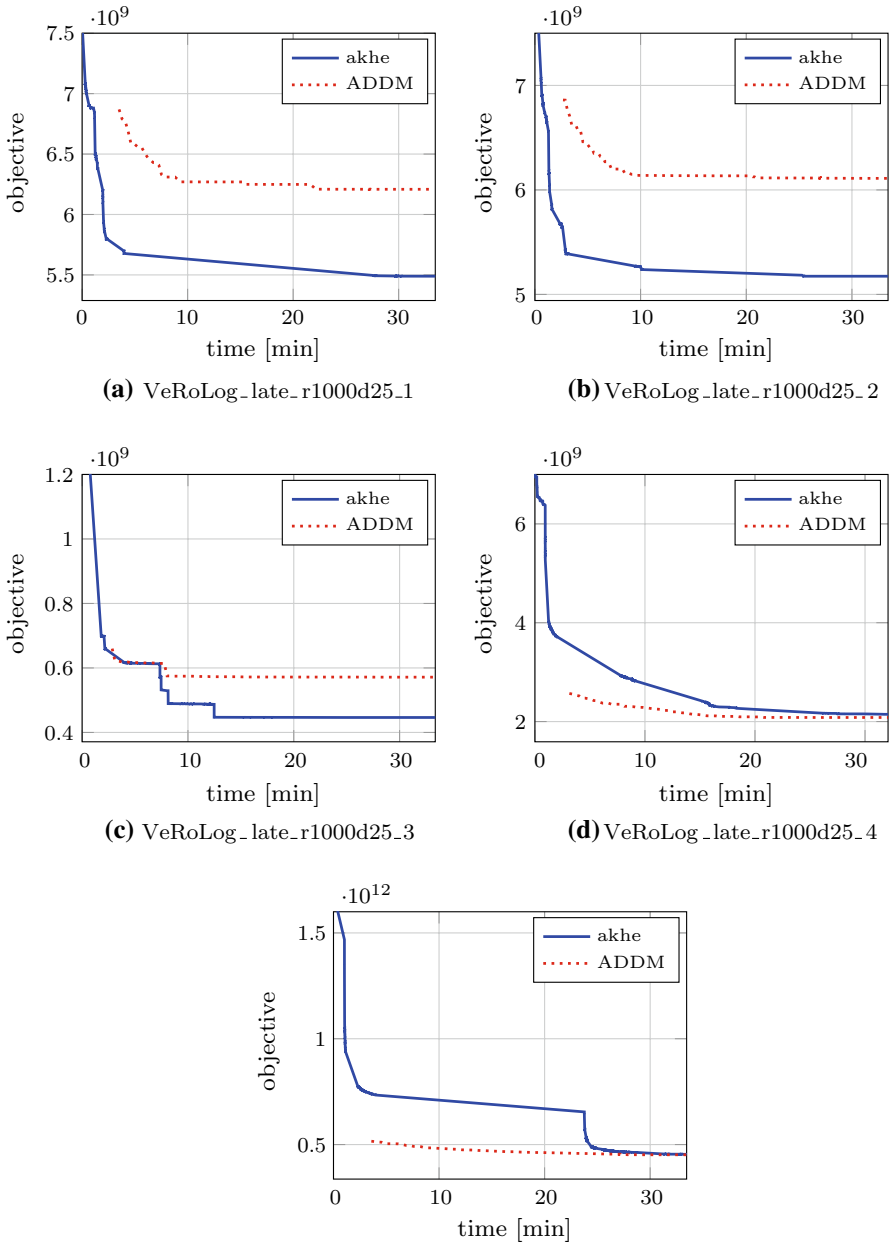
Academic challenges, such as the one described in this paper, have the pleasant property that algorithms can be compared objectively. Since it is ensured that:

Table 5 Summary of the competition results (hidden instances)

Instance	ADDM (A)		akhe (B)		mjg (C)		A vs B	A vs C	B vs C
	Avg. cost	Rank	Avg. cost	Rank	Avg. cost	Rank			
VeRoLog_hidden_r500d10_1	Infeasible	3	8543852756	2	8365899289	1	<	<	<
VeRoLog_hidden_r500d10_2	7301404930	3	5903916399	2	5771493723	1	<	<	<
VeRoLog_hidden_r500d10_3	863711081	3	814843760	1	819673854	2	<	<	≃
VeRoLog_hidden_r500d10_4	4561034407	2	4587216376	3	3799479893	1	≃	<	<
VeRoLog_hidden_r500d10_5	1.95044E+11	2	2.20579E+11	3	1.70088E+11	1	≃	<	<
VeRoLog_hidden_r500d15_1	3267429112	3	2838813151	2	2697367524	1	<	<	<
VeRoLog_hidden_r500d15_2	6264518370	3	5518848425	2	5311310871	1	<	<	<
VeRoLog_hidden_r500d15_3	300157427	3	258486750	2	217665073	1	<	<	<
VeRoLog_hidden_r500d15_4	2333125167	2	2603742796	3	2071346776	1	>	<	<
VeRoLog_hidden_r500d15_5	1.79542E+11	2	1.79855E+11	3	1.55013E+11	1	≃	<	<
VeRoLog_hidden_r1500d30_1	7848014014	3	6881717534	2	6467120269	1	<	<	<
VeRoLog_hidden_r1500d30_2	7788925107	3	7127091955	2	6489993680	1	<	<	<
VeRoLog_hidden_r1500d30_3	131376619	3	127751602	2	115315364	1	≃	<	<
VeRoLog_hidden_r1500d30_4	13173341132	3	13166641854	2	10703447930	1	≃	<	<
VeRoLog_hidden_r1500d30_5	1.31728E+12	2	1.34828E+12	3	1.15184E+12	1	>	<	<
VeRoLog_hidden_r1500d40_1	4320649929	3	3865210740	2	3599106687	1	<	<	<
VeRoLog_hidden_r1500d40_2	7316145898	3	6535928682	2	6081597171	1	<	<	<
VeRoLog_hidden_r1500d40_3	378641946	3	316000036	2	266249047	1	<	<	<
VeRoLog_hidden_r1500d40_4	3568036772	3	3269104825	2	2688601902	1	<	<	<
VeRoLog_hidden_r1500d40_5	1.92543E+12	2	2.03423E+12	3	1.62762E+12	1	>	<	<
VeRoLog_hidden_r2000d50_1	5810784719	3	5298648594	2	5030224679	1	<	<	<
VeRoLog_hidden_r2000d50_2	5328466675	3	4866236414	2	4308682151	1	<	<	<
VeRoLog_hidden_r2000d50_3	486474150	3	418201601	2	324847201	1	<	<	<

Table 5 (continued)

Instance	ADDM (A)		akhe (B)		mjg (C)		A vs B	A vs C	B vs C
	Avg. cost	Rank	Avg. cost	Rank	Avg. cost	Rank			
VeRoLog_hidden_r2000d50_4	25001890955	3	23252981136	2	18510562245	1	<	<	<
VeRoLog_hidden_r2000d50_5	1.28428E+12	3	1.25324E+12	2	1.05629E+12	1	<	<	<
VeRoLog_hidden_r2000d65_1	4810734214	3	4294615936	2	3990007875	1	<	<	<
VeRoLog_hidden_r2000d65_2	3083403319	3	2702857405	2	2465266227	1	<	<	<
VeRoLog_hidden_r2000d65_3	261437817	3	238093078	2	195919946	1	<	<	<
VeRoLog_hidden_r2000d65_4	20897565143	2	21949270729	3	16051327503	1	>	<	<
VeRoLog_hidden_r2000d65_5	1.64544E+12	3	1.61752E+12	2	1.38209E+12	1	>	<	<
VeRoLog_hidden_r2500d70_1	5923449374	3	5114335854	2	4751953813	1	<	<	<
VeRoLog_hidden_r2500d70_2	7171256068	3	6556981185	2	5643936333	1	<	<	<
VeRoLog_hidden_r2500d70_3	519502106	3	469897674	2	354745805	1	<	<	<
VeRoLog_hidden_r2500d70_4	18895390762	3	18836590052	2	14619186238	1	≃	<	<
VeRoLog_hidden_r2500d70_5	1.51888E+12	3	1.4329E+12	2	1.18559E+12	1	<	<	<
VeRoLog_hidden_r2500d75_1	6952945996	3	6156231838	2	5597113528	1	<	<	<
VeRoLog_hidden_r2500d75_2	4335142495	3	4135070761	2	3556430390	1	<	<	<
VeRoLog_hidden_r2500d75_3	428374927	3	383930752	2	301433094	1	<	<	<
VeRoLog_hidden_r2500d75_4	20347372369	3	19519419317	2	15493912178	1	<	<	<
VeRoLog_hidden_r2500d75_5	2.01467E+12	3	1.9737E+12	2	1.63357E+12	1	<	<	<
Average ranking	2.82		2.16		1.02				



(a) VeRoLog_late_r1000d25_1 **(b)** VeRoLog_late_r1000d25_2

(c) VeRoLog_late_r1000d25_3 **(d)** VeRoLog_late_r1000d25_4

(e) VeRoLog_late_r1000d25_5

Fig. 12 Comparison of the convergence profile of akhe and ADDM algorithms on VeRoLog_late_r1000d25 dataset

(1) all researchers are working on exactly the same, well-defined problem, (2) there is compensation for run times on different machines, and (3) each algorithm is applied to many different problem instances.

We described two different solution methods for the problem central to the challenge: the first method, by team akhe, is based on finding promising combinations of low level heuristics. These heuristics, such as move, swap or reverse, are combined in sequences that are randomly drawn with probabilities that are updated in a tuning process that depends on the problem instance. This algorithm is rather generic, in the sense that applying it to different problems would require relatively few changes (as long as it is easy to find initial feasible solutions for the problem).

The second method, by team ADDM, focuses on decomposing the problem. This means that the algorithm is explicitly tackling the problem of assigning tasks to days. It spends the first 80% of the computation time on finding good day to day schedules using a genetic algorithm. The last 20% of time is spent on Variable Neighborhood Descent in order to improve the routing given a certain day to day schedule. One might say this approach is intuitive, because the decomposition explicitly deals with the scheduling and the routing aspects of the problem.

We can observe differences and similarities between the two approaches. Let us focus on the most obvious difference first: team ADDM decomposed the problem whereas team akhe did not. While the two approaches appear quite different altogether, we can still find several similarities. First of all, the 'low level heuristics' as mentioned in akhe's approach overlap with the heuristics used in team ADDM's neighborhood search. Furthermore, both teams made use of the fact that initial feasible solutions were easy to find. Finally, both approaches were randomized and allowed for trying moves that appeared to be unlikely to improve the solution - albeit with a smaller probability than those moves that appeared promising.

The ability to compare algorithms objectively makes a challenge a valuable opportunity to gain insights into state-of-the-art solution techniques. In this paper, we demonstrated that the two solution approaches—although altogether different—were both effective in solving the $\mathcal{NP}$-hard optimization problem that underlined the VeRoLog Solver Challenge 2016–2017.

Acknowledgements We express our gratitude to the VeRoLog board as well as the organizing committee for the VeRoLog Conference that was held in Amsterdam, Netherlands, July 10–12, 2017. The last three authors wish to thank Gerhard Post and Daniël Mocking for co-organizing the VeRoLog Solver Challenge 2017. Alina G. Dragmir and David Mueller (team *ADDM*) have achieved their results for the all-time-best challenge using the Vienna Scientific Cluster. Additionally, Alina G. Dragomir would like to gratefully acknowledge the financial support by FWF the Austrian Science Fund (Project number P 27858).

Appendix A: Challenge rules

We summarize the challenge rules, which were originally published in (Dullaert et al. 2017). The challenge consisted of three parts, and the first two ran partially parallel in time.

Appendix A1: All-time-best challenge

The organizers disclosed 25 instances in December 2016: the "all-time-best instances". Participants were invited to submit a solution to an instance if it was better than the best solution submitted so far for this instance. Progress, i.e., the cost of the best solution over time, was shown to the participants. This information is still visible on the website, and it shows that different instances are won by different participants.

The all-time-best challenge ran till July 2, 2017 and the participants were rewarded in two ways: for every week during the all-time-best period that their solution was the best, and additionally for having the best solution at the end of the challenge. In this part of the challenge, any means, resources and time, were allowed.

Appendix A2: Restricted resources challenge

This challenge had a more "traditional" form: the resources were restricted, especially the computing time. The time T_{limit} (seconds) that each algorithm was allowed to run on the organizers' single core machine is limited by the formula $T_{\text{limit}} = 10 + 2|R|$. Here $|R|$ is the number of (delivery) requests in the instance. The organizers provided a calibration tool, so that each participant could estimate the equivalent time on his or her local machine. In addition, it was not allowed to use any software that is not freely available for commercial use. In particular, this means that for example the use of commercial MILP solvers was forbidden. Each algorithm had to run on a new set of 25 instances (available since April 1 2017). Furthermore, each solver had to run on each instance, using nine different random seeds, to reduce the variance coming from randomized algorithms. Non-randomized algorithms could also profit from the random seeds: they were known to be between 10^8 and 10^9 with a different starting digit for each seed, and hence it was possible to detect this and run 9 different deterministic algorithms. The corresponding results and solver binaries were submitted on May 8, 2017.

The evaluation of algorithms in the restricted resources challenge was done as follows. A rank score was calculated per instance for each solver. First, per instance, the two best solutions and the two worst solutions found by the solver were removed. The remaining five solutions were used to compute the score of the solver. If these five solutions were all feasible, their average counted as the score of the solver. Alternatively, if there were infeasible solutions among the middle 5 solutions, that solver was first ranked with respect to the number of feasible solutions, and secondary by the average cost of the feasible ones. Finalists were announced on June 1.

Appendix A3: The finals

The finalists' solvers were run by the organizers on a set of 50 not previously disclosed (the so-called *hidden*) instances. Again, per solver per instance but equal for each finalist, nine runs with different random seeds were done, again with nine

different starting digits. A solver ranking per instance was made with the same rules as above, as well as a ranking of the solvers based on these scores. The winner of the challenge was the participant whose solver had the lowest mean of the ranks.

References

Ahmed L, Mumford C, Kheiri A (2019) Solving urban transit route design problem using selection hyper-heuristics. Eur J Oper Res 274(2):545–559

Asta S, Özcan E (2015) A tensor-based selection hyper-heuristic for cross-domain heuristic search. Inf Sci 299:412–432

Battarra M, Cordeau JF, Iori M (2014) Pickup-and-delivery problems for goods transportation. In: Toth P, Vigo D (eds) Vehicle Routing, chap 6, pp 161–191. doi:https://epubs.siam.org/doi/pdf/10.1137/1.9781611973594.ch6

Burke EK, Gendreau M, Hyde M, Kendall G, Ochoa G, Özcan E, Qu R (2013) Hyper-heuristics: a survey of the state of the art. J Oper Res Soc 64(12):1695–1724

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

Coelho LC, Cordeau JF, Laporte G (2014) Thirty years of inventory routing. Transp Sci 48(1):1–19

Cordeau JF, Laporte G (2007) The dial-a-ride problem: models and algorithms. Ann Oper Res 153(1):29–46

Croes GA (1958) A method for solving traveling-salesman problems. Oper Res 6(6):791–812

Dethloff J (2001) Vehicle routing and reverse logistics: the vehicle routing problem with simultaneous delivery and pick-up. OR-Spektrum 23(1):79–96

Dror M, Fortin D, Roucairol C (1998) Redistribution of self-service electric cars: a case of pickup and delivery. Tech. Rep. RR-3543, INRIA

Dueck G (1993) New optimization heuristics: the great deluge algorithm and the record-to-record travel. J Comput Phys 104(1):86–92

Dullaert W, Gromicho J, van Hoorn J, Post G, Vigo D (2017) The VeRoLog solver challenge 2016–2017. J Veh Rout Algorithms 1:1–3. https://doi.org/10.1007/s41604-016-0001-7

Fagerland MW, Sandvik L (2009) The Wilcoxon–Mann–Whitney test under scrutiny. Stat Med 28(10):1487–1497

Fishman E (2016) Bikeshare: a review of recent literature. Transp Rev 36(1):92–113

Gromicho JA, Haneyah S, Kok L (2015) Solving a real-life vrp with inter-route and intra-route challenges. https://doi.org/10.2139/ssrn.2610549

Hansen P, Mladenović N (1999) An introduction to variable neighborhood search. In: Meta-heuristics, Springer, pp 433–458

Hart E, Ross P, Corne D (2005) Evolutionary scheduling: a review. Genetic Progr Evol Mach 6(2):191–220

Jian N, Freund D, Wiberg HM, Henderson SG (2016) Simulation optimization for a large-scale bike-sharing system. In: Proceedings of the 2016 Winter Simulation Conference, IEEE Press, Piscataway, NJ, USA, WSC '16, pp 602–613

Johnson DS (1973) Near-optimal bin packing algorithms. PhD thesis, Massachusetts Institute of Technology

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. Evolut Comput 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

Kruskal WH (1957) Historical notes on the Wilcoxon unpaired two-sample test. J Am Stat Assoc 52(279):356–360

Lin S, Kernighan BW (1973) An effective heuristic algorithm for the traveling-salesman problem. Oper Res 21(2):498–516
Martello S, Toth P (1990) Lower bounds and reduction procedures for the bin packing problem. Discrete Appl Math 28(1):59–70
Masson R, Lehuede F, Peton O (2014) The dial-a-ride problem with transfers. Comput Oper Res 41:12–23
Min H (1989) The multiple vehicle routing problem with simultaneous delivery and pick-up points. Transp Res Part A: General 23(5):377–386. https://doi.org/10.1016/0191-2607(89)90085-X
Montané FAT, Galvão RD (2006) A tabu search algorithm for the vehicle routing problem with simultaneous pick-up and delivery service. Comput OR 33:595–619
Parragh S, Doerner K, Hartl R (2008) A survey on pickup and delivery problems: Part II: Transportation between pickup and delivery locations. Journal für Betriebswirtschaft 58:81–117. https://doi.org/10.1007/s11301-008-0036-4
Pfrommer J, Warrington J, Schildbach G, Morari M (2014) Dynamic vehicle redistribution and online price incentives in shared mobility systems. IEEE Trans Intell Transp Syst 15(4):1567–1578
Rainer-Harbach M, Papazek P, Hu B, Raidl GR (2013) Balancing bicycle sharing systems: a variable neighborhood search approach. In: Middendorf M, Blum C (eds) Evolutionary computation in combinatorial optimization. Springer, Berlin, pp 121–132
Rand GK (2009) The life and times of the savings method for vehicle routing problems. ORiON 25(2):125–145
Raviv T, Kolka O (2013) Optimal inventory management of a bike-sharing station. IIE Trans 45(10):1077–1093
Schilde M, Doerner K, Hartl R (2011) Metaheuristics for the dynamic stochastic dial-a-ride problem with expected return transports. Comput Oper Res 38(12):1719–1730
Schuijbroek J, Hampshire R, van Hoeve WJ (2017) Inventory rebalancing and vehicle routing in bike sharing systems. Eur J Oper Res 257(3):992–1004
Sörensen K, Glover FW (2013) Metaheuristics. In: Gass SI, Fu MC (eds) Encyclopedia of operations research and management science. Springer, Berlin, pp 960–970
Wall MB (1996) A genetic algorithm for resource-constrained scheduling. PhD thesis, Massachusetts Institute of Technology
Wilson D, Rodrigues S, Segura C, Loshchilov I, Hutter F, Buenfil GL, Kheiri A, Keedwell E, Ocampo-Pineda M, Özcan E, Pena SIV, Goldman B, Rionda SB, Hernandez-Aguirre A, Veeramachaneni K, Cussat-Blanc S (2018) Evolutionary computation for wind farm layout optimization. Renew Energy 126:681–691

Affiliations

Ahmed Kheiri[1] · Alina G. Dragomir[2] · David Mueller[3] · Joaquim Gromicho[4,5] · Caroline Jagtenberg[4] · Jelke J. van Hoorn[4]

Alina G. Dragomir
alina-gabriela.dragomir@univie.ac.at

David Mueller
david.mueller@tuwien.ac.at

Joaquim Gromicho
Joaquim.Gromicho@ortec.com

Caroline Jagtenberg
C.J.Jagtenberg@gmail.com

Jelke J. van Hoorn
Jelke.vanHoorn@ortec.com

[1] Department of Management Science, Lancaster University Management School, Lancaster LA1 4YX, UK

[2] Faculty of Business, Economics and Statistics, University of Vienna, Vienna, Austria

[3] Institute for Theoretical Physics, Vienna University of Technology, Vienna, Austria

[4] ORTEC, Houtsingel 5, 2719 EA Zoetermeer, The Netherlands

[5] School of Business and Economics, Vrije Universiteit, De Boelelaan 1105, Amsterdam, The Netherlands