Journal of Heuristics (2023) 29:349–382
https://doi.org/10.1007/s10732-023-09514-x

The ALNS metaheuristic for the transmission maintenance scheduling

David Woller[1,2] · **Jakub Rada**[1] · **Miroslav Kulich**[1]

Received: 31 March 2022 / Revised: 10 January 2023 / Accepted: 10 May 2023 /
Published online: 27 May 2023

Abstract

ROADEF Challenge is an established international competition addressing challenging industrial problems of combinatorial optimization. It is organized by the French Operations Research and Decision Support Society (ROADEF) every 2 years since 1999. The most recent ROADEF challenge 2020 was co-organized by the French electricity transmission network operator, the RTE company. The competition problem addressed a novel variant of the transmission maintenance scheduling problem, distinctive in that it has multiple time-dependent properties, constraints, and a risk-based aggregate objective function. Therefore, the problem is more complex than the previous formulations, and the existing methods are not directly applicable. This paper presents a metaheuristic algorithm based on the adaptive large neighborhood search. The algorithm's performance is based on a large bank of newly proposed problem-specific destroy and repair heuristics, an efficient local search engine, and a penalization mechanism for avoiding invalid solutions. The algorithm is compared with the best-known solutions from all competition phases and other methods submitted to the final phase. The result shows that the method yields consistent performance in all available datasets. The proposed algorithm finished 6th in the semifinal phase of the competition and 8–9th in the final phase. Finally, the effect of individual components and the algorithm's behaviour are analyzed in detail.

✉ David Woller
wolledav@cvut.cz

Jakub Rada
radajak5@fel.cvut.cz

Miroslav Kulich
kulich@cvut.cz

[1] Czech Institute of Informatics, Robotics, and Cybernetics, Czech Technical University in Prague, Jugoslávských Partyzánů 1580/3, Praha 6 160 00, Czech Republic

[2] Department of Cybernetics, Faculty of Electrical Engineering, Czech Technical University in Prague, Karlovo Náměstí 13, Praha 2 121 35, Czech Republic

Keywords Adaptive large neighborhood search · Metaheuristics · Combinatorial optimization · Transmission maintenance scheduling · ROADEF challenge 2020

1 Introduction

The power transmission and distribution industry provide a large number of combinatorial optimization problems, some of which can be solved by classical graph theory algorithms (Than Kyi et al. 2019; Borůvka 1926), while others require the design of specialized algorithms. This paper addresses a novel variant of the Transmission Maintenance Scheduling (TMS) problem. The goal of TMS is to schedule the maintenance of a high-voltage electricity network, which requires the disconnecting of individual power lines. These disconnections are called interventions and correspond to the tasks to be scheduled. The addressed variant is more complex than various existing variants of TMS (Froger et al. 2016), as it has multiple time-dependent properties, constraints, and a nonlinear objective function. The objective function to be minimized expresses the risk-induced cost of a schedule, where the financial risk evaluation is based on historical data. The addressed TMS variant is proposed by the operator of the French power network (the largest in Europe), the RTE company.

As the TMS is a challenging problem with direct industrial application, it was announced as a competition problem in the ROADEF Challenge 2020. ROADEF Challenge is an established international competition held every two years since 1999. The competition's goal is to identify a previously unsolved real-world industrial problem and present it to the operations research community. The duration of the 2020 challenge was 16 months, and 74 teams participated. Previous ROADEF challenges addressed various problems, such as the Glass Sheets Cutting Optimization problem (proposed by Saint-Gobain Glass France, 2018), the Liquid Oxygen Inventory Routing problem (Air Liquide, 2016), the Rolling Stock Unit Management at Railway Sites problem (Société Nationale des Chemins de Fer, 2014), the Machine Reassignment problem (Google, 2012), or the Large-scale Production Management problem (Électricité de France, 2010). The successful techniques come from all areas of combinatorial optimization. For example, the methods from the final phase in 2014 included custom graph algorithms, local search, metaheuristics (Tabu Search, Simulated Annealing, Large Neighborhood Search), as well as Constraint Programming and Integer Programming techniques (Double Column Generation, Benders Decomposition) (Artigues et al. 2018).

In this paper, a metaheuristic algorithm based on the Adaptive Large Neighborhood Search (ALNS) metaheuristic (Pisinger and Ropke 2010) is proposed for the TMS. The basic principle of the algorithm is to repeatedly partially destroy and repair the current solution, which allows the exploration of various solution neighbourhoods. In these neighbourhoods, a local search is applied to refine the current solution and reach a local optimum. A heuristic approach was selected, as the competition instances contain up to 1000 interventions and are likely to be computationally intractable for exact methods. The addressed TMS variant is rich in properties and constraints, and thus, methods for other variants could not be easily reapplied. The ALNS metaheuristic

in particular was selected because a large number of destroy and repair heuristics can be designed, each mimicking a different partial property of the problem.

The initial version of the proposed algorithm was described in Woller and Kulich (2021), with experimental results on the qualification and semifinal dataset. The qualification version consisted of the standard ALNS metaheuristic adapted for the TMS problem, extended by a local search phase. An augmented objective function was formulated to handle both soft and hard constraint violations, and a bank of problem-specific destroy and repair heuristics was newly proposed, together with a set of local search operators.

This paper describes the algorithm submitted to the final phase of the challenge. Several extensions are newly introduced in this paper, together with an exhaustive experimental evaluation on all four competition datasets. The following contributions are newly proposed in this paper, as they were not implemented in the qualification method.

- The proposed repair heuristics are systematically hybridized. Hybridization adds randomness to otherwise deterministic intervention selection and start time selection and enables to further diversify the search process.
- The evaluation of some repair heuristics is sped by estimates based on precomputing intervention static properties, such as average resource demand, cost, and length.
- An automated algorithm design and tuning setup is presented. In the latter stages of the competition, both the structure of the algorithm and the parameters were determined by Iterated Racing for Automatic Algorithm Configuration (López-Ibáñez et al. 2016).

The presented method yields consistent performance on all 4 competition datasets. The method finished 6th in the semifinal phase of the competition (1st in the Junior category) and 8–9th in the final phase (2nd–3rd in the Junior category).

The rest of the paper is structured as follows. Section 2 details the related works. Section 3 provides a formal description of the Transmission Maintenance Scheduling problem. The method submitted to the final competition phase is described in Sect. 4. The experimental results are presented and discussed in Sects. 5, and 6 gives the conclusions.

2 Related works

The ROADEF Challenge 2020 addresses a variant of the Transmission Maintenance Scheduling (TMS) problem. The goal of TMS is to schedule interventions in a transmission network to perform the necessary maintenance. The existing literature distinguishes between long-term TMS and short-term TMS (Shahidehpour and Marwali 2000). In the case of long-term TMS, the time horizon is typically a year, the granularity is days or even weeks, and the maintenance to be scheduled is preventive. The competition problem falls into this category. As for short-term TMS, the time horizon ranges in days or weeks, the granularity of the schedule can be an hour or less, and maintenance is typically preplanned and reactively rescheduled depending on the

current state of the network. An example of a short-term TMS variant can be found in Lv et al. (2015).

A closely related problem to TMS is the Generator Maintenance Scheduling problem (GMS). The goal of the GMS is to schedule the shutdown of power generation units rather than disconnecting power lines, but otherwise, the GMS has objectives and constraints similar to the TMS (Froger et al. 2016). These two problems are often addressed simultaneously (Wang et al. 2016).

A wide range of TMS problems was proposed in the literature, but the formulation used in the ROADEF Challenge 2020 competition problem does not seem to fit any variant previously addressed. In existing TMS variants, the structure of the network is often modelled by a transportation model (Abirami et al. 2014) or a DC power flow model (Da Silva et al. 2000). The competition problem does not employ an explicit network model. The TMS objective function can be reliability-based (Schlünz and Van Vuuren 2013), cost-based (El-Sharkh 2014) or a combination of both objectives (Moro and Ramos 1999). The competition problem uses the last variant, as its objective function is a nonlinear aggregation of risk-induced cost values. There can be numerous constraints related to different attributes of the TMS. Interventions can have restricted time windows and be subject to precedence or concurrency constraints. Maintenance can be limited by the availability of workforce or resources. To ensure incident-free network operation, there may also be constraints on the maximum capacity of the transmission lines or the minimum customer demand (Froger et al. 2016). From these, the competition problem considers time windows, mutual exclusivity of interventions, and limited workforce. TMS problems often need to reflect uncertainty, which can be caused by unpredictable fluctuations in supply and demand or extreme weather conditions. This uncertainty can be modelled by the loss of load probability (Reihani et al. 2012) or the expected energy not served (Lu et al. 2012). The competition problem does not utilize an explicit model. Instead, uncertainty is implicitly incorporated into the input data, which contain risk-based costs for each possible start time of every intervention and all relevant crisis scenarios. Finally, most constraints and problem properties are time-dependent, which is not common in existing formulations.

Due to the intractability of the TMS, there has been a cumulative effort of researchers to develop both specialized exact methods and heuristic methods. TMS variants are often formulated as a Mixed Integer Programming problem and addressed with a commercial solver, although this approach is reported to be feasible only for small instances (Mollahassani-Pour et al. 2014). Its scalability can be improved using the Benders decomposition (Geetha and Swarup 2009), as many TMS variants possess the necessary block structure. Some custom problem-specific exact methods were also proposed, such as a dynamic programming approach (Huang 1997) or the branch-and-cut technique (Pandžić et al. 2012). Various metaheuristic algorithms were successfully applied to approximately solve large instances. According to Froger et al. (2016), the most commonly adapted metaheuristics are population-based, such as Genetic Algorithm (Volkanovski et al. 2008) or Particle Swarm optimization (Suresh and Kumarappan 2013). Metaheuristics based on local search are also applied, for example, Simulated Annealing (Saraiva et al. 2011) or Tabu Search (Burke and Smith 2000). Some authors combine exact and heuristic methods in a hybrid solver, e.g. Genetic Algorithm and Mixed Integer Programming (Feng et al. 2009) or Non-Dominated

Sorting Genetic Algorithm III with a custom Dual Simplex method (Salinas San Martin et al. 2022).

The Adaptive Large Neighborhood Search (ALNS) metaheuristic was first applied to the Pickup and Delivery Problem with Time Windows (Ropke and Pisinger 2006) and it is an extension of the Large Neighborhood Search (LNS) metaheuristic (Shaw 1998). LNS introduced the concept of large neighbourhoods, defined by pairs of destroy and repair heuristics, as a counterpart to small neighbourhoods typically used in local search operators. The ALNS then added a learning mechanism that adjusts the use of individual heuristics according to their previous performance, thus adapting the method to the currently solved instance. The main strength of the LNS is providing a robust diversification mechanism that prevents premature convergence, whereas the ALNS is suitable for managing large banks of available heuristics or addressing highly heterogeneous datasets. The ALNS was previously applied to a related problem focused on Maintenance Scheduling for Offshore Oil and Gas Platforms (Khalid et al. 2021). Regarding scheduling problems in general, the existing literature documents many successful applications, such as Service Technician Routing and Scheduling (Kovacs et al. 2012), Multiple Agile Satellites Scheduling (He et al. 2018), Electric Vehicle Scheduling (Wen et al. 2016) or Distributed Reentrant Permutation Flow Shop Scheduling (Rifai et al. 2016). These problems have multiple properties that can be exploited by problem-specific destroy and repair heuristics, which is true for the addressed TMS problem as well. The hybridization mechanisms used in this paper to further expand the portfolio of heuristics and adding a local search phase to the standard ALNS are inspired by Smith and Imeson (2017), which proposed a state of the art ALNS-based algorithm for the Generalized Traveling Salesman Problem. Finally, the proposed method obtains valid solutions using a static penalization of constraint violations with tunable weights, frequently used in Evolutionary Optimization (Back et al. 1997). More robust alternatives to this approach are dynamic penalization (Joines et al. 1994) or adaptive penalization (Ben Hamida and Schoenauer 2000). To the best of our knowledge, the ALNS is applied to TMS for the first time in this paper, respectively, in the preceding conference paper (Woller and Kulich 2021).

3 Problem definition

This section formally defines the TMS problem addressed in the ROADEF Challenge 2020. It is structured as follows. The necessary notation is defined in Sect. 3.1. The problem constraints are described in Sect. 3.2 and the objective function is provided in Sect. 3.3. More details about the problem can be found in Ruiz et al. (2020).

3.1 Notations

This section defines the notation necessary for the formal definition of the problem and the exact description of the proposed method. The notation is adapted from Ruiz et al. (2020) and slightly expanded. As mentioned in Sect. 2, the TMS variant addressed

employs an explicit model of the transmission network. Therefore, the formulation is more similar to a highly constrained scheduling problem.

Planning horizon is finite and discrete. Let $T \in \mathbb{N}$ denote the number of time steps in an instance. The discrete-time horizon is defined as $H = \{1, .., T\}$. The value of T determines the granularity of the instance; for example, $T = 53$ corresponds to weeks and $T = 365$ to days.

Interventions correspond to the disconnections of individual power lines from the network and represent the tasks to be scheduled. Let I be the set of all interventions and $i \in I$ a single intervention. Then $\Delta_{i,t} \in \mathbb{N}$ is the duration of an intervention i, given that it started at time t.

Exclusions describe the mutual exclusivity of some interventions. Let Exc be the set of all exclusions. An exclusion is a triplet $(i_1, i_2, t) \in Exc$, where i_1, i_2 are two interventions that cannot be scheduled simultaneously at time $t \in H$.

Resources, such as workforce or equipment, are needed to carry out an intervention. Let C be the set of all resources. The demand for the resource $c \in C$ at time $t \in H$ by an intervention $i \in I$, which started at time t', is given by $r_{i,t'}^{c,t} \in \mathbb{R}$. Each resource $c \in C$ has a lower usage bound $l_t^c \in \mathbb{R}$ and an upper usage bound $u_c^t \in \mathbb{R}$ for each time $t \in H$.

Scenarios are external factors, such as seasonal weather conditions, that can disrupt power delivery and cause financial losses to the network operator. Each time $t \in H$ is assigned a set of possible scenarios S_t.

Risk expresses the possible financial loss due to the scheduling of an intervention $i \in I$ to begin at time $t' \in H$. The risk value at time $t \in H$ is then expressed as $risk_{i,t'}^{s,t} \in \mathbb{R}$, where $s \in S_t$ is an imminent scenario. The risk values are based on historical data collected by the network operator.

Solution of the TMS problem is a list x of pairs $(i, t') \in I \times H$, where t' is the scheduled start time of an intervention i. Finally, let I_t be the set of interventions ongoing at time $t \in H$ and let I_x be the set of interventions scheduled in a solution x. An example of a valid solution is shown in Fig. 1.

3.2 Constraints

Any solution x must meet the following constraints to represent a valid schedule.

Non-preemptive scheduling: an intervention must proceed without interruption once it has started. If an intervention $i \in I$ starts at time $t \in H$, it must end at time $t + \Delta_{i,t}$.

Everything scheduled on time: all interventions must end within the planning horizon. It must hold that $t + \Delta_{i,t} \leq T + 1, \forall i \in I$.

Resource constraints: the total workload of each resource $c \in C$ at time $t \in H$ must be within its lower and upper bounds. The total workload of resource c at time t is given by

$$r^{c,t} = \sum_{i \in I_t} r_{i,t'}^{c,t}.$$

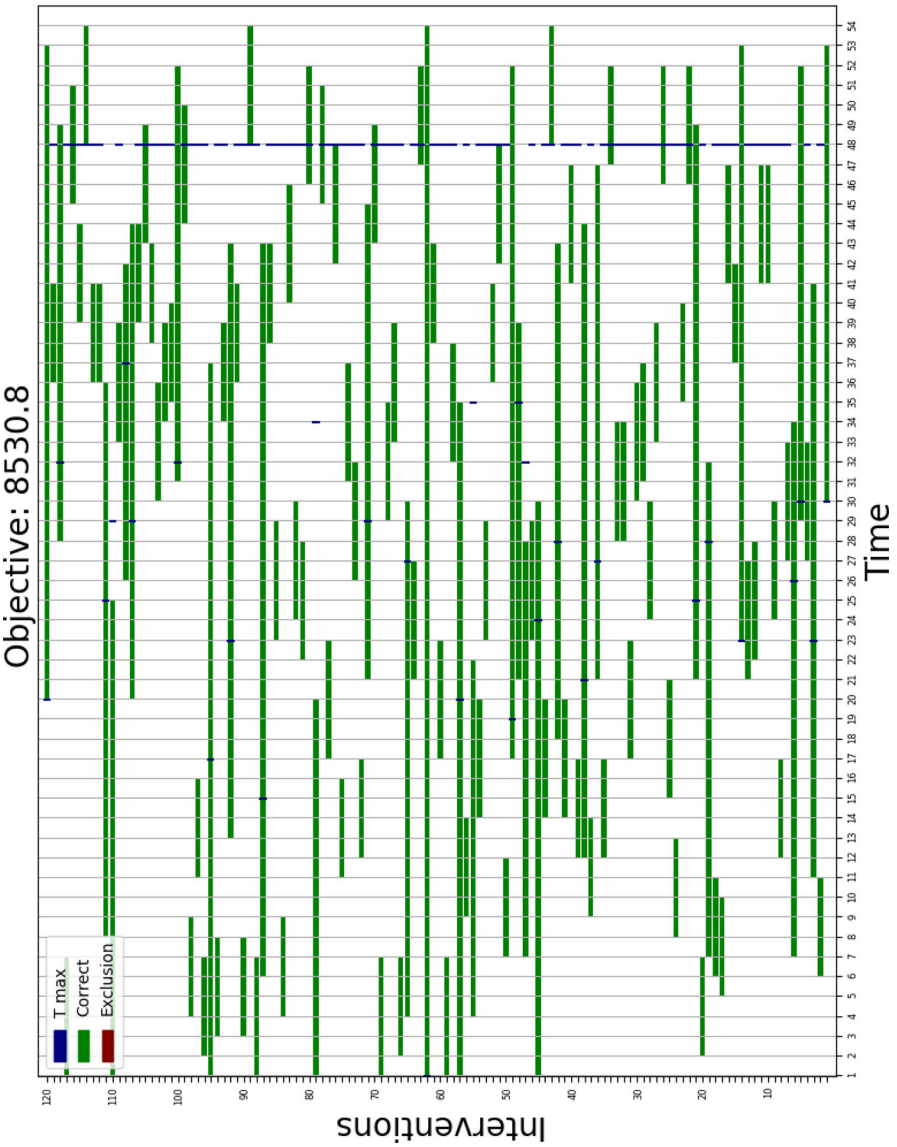


Fig. 1 Solved instance C01

It must hold that $l_t^c \leq r^{c,t} \leq u_t^c, \forall c \in C, t \in H$.

Exclusivity constraints: all the exclusions $(i_1, i_2, t) \in Exc$ has to be respected at all times. It must hold that $i_1 \in I_t \implies i_2 \notin I_t, \forall (i_1, i_2, t) \in Exc$.

3.3 Objective

The objective value obj of a schedule is defined as a weighted aggregation of two values: mean cost obj_1 and expected excess obj_2.

Mean cost evaluates the total planning risk averaged for all scenarios and time steps. The cumulative planning risk at $t \in H$ for a scenario $s \in S_t$ is defined as

$$risk^{s,t} = \sum_{i \in I_t} risk^{s,t}_{i,t'},$$

where t' is the start time of the intervention i. The mean cumulative planning risk at $t \in H$ is then defined as

$$\overline{risk^t} = \frac{1}{|S_t|} \sum_{s \in S_t} risk^{s,t}.$$

Finally, the mean cost is defined as

$$obj_1 = \frac{1}{T} \sum_{t \in H} \overline{risk^t}.$$

Expected excess is intended to suppress extreme cost variability in different scenarios. The expected excess at time $t \in H$ is defined as

$$Excess_\tau(t) = max(0, Q^t_\tau - \overline{risk^t}).$$

Here, Q^t_τ is the τ quantile of cumulative planning risks of all scenarios $s \in S_t$ at time $t \in H$, defined as

$$Q^t_\tau = Q_\tau(\{risk^{s,t}\}_{s \in S_t}),$$

. The value of τ is part of a TMS problem instance. The expected excess of the whole schedule is then defined as

$$obj_2 = \frac{1}{T} \sum_{t \in H} Excess_\tau(t).$$

Final objective obj is given by

$$obj = \alpha \times obj_1 + (1 - \alpha) \times obj_2,$$

where $\alpha \in [0, 1]$ is a provided scaling factor.

4 ALNS metaheuristic for the TMS problem

This section describes the algorithm proposed for the final phase of the challenge. The high-level ALNS metaheuristic is described in Sect. 4.1. Section 4.2 introduces the augmented objective function used to penalize invalid solutions. The key element of the ALNS is a bank of destroy and repair heuristics, designed specifically for the TMS problem. These are described in Sect. 4.3. Hybridization, which systematically adds randomness to the proposed repair heuristics, is introduced in Sect. 4.4. Section 4.5 then presents an approach to reducing the time requirements of some repair heuristics. Finally, the local search is described in Sect. 4.6.

4.1 Overall solution approach

The Adaptive Large Neighborhood Search metaheuristic, first introduced in Ropke and Pisinger (2006), is based on repeated partial destruction and reconstruction of a current solution. For this purpose, a large bank of heuristics for destroying and repairing was proposed specifically for the TMS problem. The usage of individual heuristics is adaptive, meaning that their selection weights in each iteration are adjusted on the basis of their previous performance. In addition, a local search phase was added to the standard ALNS, together with newly proposed local search operators. Local search enables efficient refinement of a current solution and reaching local optima.

Algorithm 1 ALNS metaheuristic

$i \leftarrow 0,\ x^* \leftarrow \emptyset$
while !`stop()` **do**
  if $i = 0$ **then**
    $x' \leftarrow$ `construction()`
    $\rho^-,\ \rho^+ \leftarrow (1, ..., 1)$
  $x \leftarrow x'$
  $N_r \leftarrow \mathcal{U}(0, \texttt{DEPTH} * \texttt{size}(x))$
  select `d`, `r` according to ρ^- and ρ^+
  $x \leftarrow \texttt{r}(\texttt{d}(x, N_r), N_r)$
  $x \leftarrow$ `local_search`(x)
  if $c(x) < c(x')$ **then**
    $i \leftarrow 0,\ x' \leftarrow x$
  else
    $i \leftarrow i + 1$
  if $c(x) < c(x^*)$ **then**
    $x^* \leftarrow x$
  update ρ^-(`d`) and ρ^+(`r`)
  if $i = i_{max}$ **then**
    $i \leftarrow 0$
return x'

The resulting extended variant of the ALNS is described in Algorithm 1. Here, the variables x, x', x^* represent the current, current best and overall best solution, respectively. The counter i controls the restarting of the main ALNS loop and resets after $i_{max} =$ `ITERS_MAX` not improving iterations. Each restart begins with creating

a new solution x' and resetting the selection weights ρ^-, ρ^+ of destroy and repair heuristics, respectively (lines 3–5). Each iteration performs the following process. The rate of destruction N_r of a solution x is randomly sampled uniformly from the interval $(0, \texttt{DEPTH} * \texttt{size}(x))$ (line 7). Then, the destroy heuristic `d` and the repair heuristic `r` are selected according to the selection weights ρ^-, ρ^+, using a roulette wheel selection mechanism (line 8). After that, N_r interventions are removed from x using `d` and reinserted using `r` and subsequently x is subjected to a local search (lines 9–10). Finally, the acceptance and restart criteria are checked and the selection weights are updated (lines 11–18). The main ALNS loop is controlled by an arbitrary stop condition (line 2), which was given by a fixed computation time according to the rules of the challenge. Alternatively, a maximum number of fitness evaluations or the number of iterations could be used.

The mechanism of updating the selection weights of the individual heuristics is intended to reflect their previous performance and is adapted from Pisinger and Ropke (2010). The weight $\rho^-(\texttt{d})$ of a destroy heuristic `d` is updated using the following formula:

$$\rho^-(\texttt{d}) = \lambda\rho^-(\texttt{d}) + (1-\lambda)\omega,$$

where

$$\omega = \begin{cases} \Omega_1, & \text{if new overall best solution } x^* \text{ was found,} \\ \Omega_2, & \text{if new current best solution } x' \text{ was found,} \\ \Omega_3, & \text{otherwise.} \end{cases}$$

It must hold that $\Omega_1 \geq \Omega_2 \geq \Omega_3 \geq 0$. Then $\lambda \in [0, 1]$ is a decay parameter. Regarding the remaining solver parameters, $\texttt{ITERS\_MAX} \in \mathbb{N}$ and $\texttt{DEPTH} \in [0, 1]$. The values of `DEPTH`, `ITERS_MAX`, λ, Ω_1, Ω_2, and Ω_3 are all fixed solver parameters. Their tuning is described in Sect. 5.3.

4.2 Augmented objective function

The addressed variant of the TMS problem considers four kinds of constraints, described in Sect. 3.2. The scheduling problem is non-preemptive and limited by a finite planning horizon, both of which requirements are easy to satisfy and incorporate into the implementation. However, the resource and exclusivity constraints are variable in time and thus significantly harder to meet. No polynomial-time constructive or repair procedure is known. Therefore, the proposed algorithm also considers invalid solutions during the search process and treats the latter two constraints as soft constraints. The invalidity rate is penalized and combined with the actual objective function of the TMS problem obj in the so-called augmented objective function obj_{aug}. This mechanism is inspired by Michel and Hentenryck (2018).

Individual measures of violating the resource and disjunctive constraints are defined as follows. The excessive usage of the resource c at time t is given by $r^{c,t}_{over} = \max(r^{c,t} - u^c_t, 0)$ and the insufficient usage by $r^{c,t}_{under} = \max(l^c_t - r^{c,t}, 0)$.

Then, the total resource usage of all resources beyond their bounds can be defined as

$$r_{under} = \sum_{c \in C} \sum_{t \in H} r_{under}^{c,t},$$
$$r_{over} = \sum_{c \in C} \sum_{t \in H} r_{over}^{c,t}.$$

As for the exclusivity constraints, the penalty can be defined as

$$e_{pen} = \sum_{(i_1, i_2, t) \in Exc} \texttt{both\_scheduled}(i_1, i_2, t),$$

where $\texttt{both\_scheduled}(i_1, i_2, t) = 1$ if $i_1, i_2 \in I_t$, and 0 otherwise. Finally, the augmented objective function obj_{aug} can be defined as a weighted aggregation of obj and the introduced penalties:

$$obj_{aug} = obj + \beta_1 * r_{under} + \beta_2 * r_{over} + \gamma * e_{pen}.$$

The values of $\beta_1, \beta_2, \gamma \in [10,000, 100,000]$ are tunable parameters. Their ranges are empirically set so that all valid solutions have values of obj_{aug} lower than those of the invalid ones.

4.3 Destroy and repair heuristics

The basic principle of ALNS is to repeatedly ruin and recreate part of a current solution x. In each iteration, N_r interventions are removed by applying a randomly selected destroy heuristic (`d`) N_r times (Algorithm 1, line 9). Then, these interventions are scheduled again, using a repair heuristic `r`. The triplet (`d`, `r`, N_r) defines the "large neighborhood" in the ALNS metaheuristic. Various destroy heuristics designed specifically for the TMS problem are introduced in Sect. 4.3.1 and repair heuristics in Sect. 4.3.2. Section 4.4 describes two hybridization mechanisms that allow a further generalization of the proposed repair heuristics. Finally, an approach to reduce the computational complexity of repair heuristics is presented in Sect. 4.5.

4.3.1 Destroy heuristics

All destroy heuristics operate as follows. Let x be a current solution and I_x be the set of interventions currently scheduled in x. A destroy heuristic selects a single intervention $i \in I_x$ and removes the pair (i, t) from x. Intervention i can be randomly selected according to some of its individual properties (length, number of exclusions) or according to its influence on some properties of the current schedule (decrease in cost, decrease in resource demand). The resulting partial solution is marked as $x \backslash i$. Regarding complexity, most of the destroy heuristics are $\mathcal{O}(|I|)$. The proposed destroy heuristics are formally defined in Table 1.

Table 1 Destroy heuristics

Heuristic	Intervention selection rule
Random (RND)	$i = \underset{i \in I_x}{\text{rand}()}$
Cheapest (CH)	$i = \arg\min_{i \in I_x} (obj_{aug}(x) - obj_{aug}(x \backslash i))$
Most expensive (ME)	$i = \arg\max_{i \in I_x} (obj_{aug}(x) - obj_{aug}(x \backslash i))$
Lowest resource demand (LRD)	$i = \arg\min_{i \in I_x} (r_{total}(x) - r_{total}(x \backslash i))$
Highest resource demand (HRD)	$i = \arg\max_{i \in I_x} (r_{total}(x) - r_{total}(x \backslash i))$
Shortest (SH)	$i = \arg\min_{i \in I_x} \Delta_{i,t'}$
Longest (LN)	$i = \arg\max_{i \in I_x} \Delta_{i,t'}$
Least exclusions (LEX)	$i = \arg\min_{i \in I_x} \lvert Exc_i \rvert$
Most exclusions (MEX)	$i = \arg\max_{i \in I_x} \lvert Exc_i \rvert$
Least used (LU)	$i = \arg\min_{i \in I_x} removed_i$
Most used (MU)	$i = \arg\max_{i \in I_x} removed_i$

Here, t' is the start time currently scheduled for $i \in I_x$, $Exc_i \subset Exc$ is a set of exclusions involving i, $removed_i$ is a counter incremented at each removal of i, and r_{total} is the total resource usage of a current solution, defined as

$$r_{total} = \sum_{c \in C} \sum_{t \in H} r^{c,t}.$$

For example, the HRD destroy heuristic removes the intervention i, which is the most resource-intensive in a current solution x.

4.3.2 Repair heuristics

All repair heuristics perform the following operation. A single unscheduled intervention $i \in I \backslash I_x$ is selected and its new start time t' is determined. The pair (i, t') is then added to a partial solution x, which is marked as $x \cup (i, t')$. The complexity of most repair heuristics is $\mathcal{O}(|I| \times |H|)$ due to the need to determine the start time $t' \in H$. As most of the properties of the intervention are time-dependent, the start time t' influences the duration of the intervention, the total cost and the demand for resources of $x \cup (i, t')$. Therefore, four different start time selection mechanisms are proposed in Table 2.

Individual repair heuristics are formally defined in Table 3. Most heuristics employ the cheapest start time $t'_{i,cheap}$, including those based on a time-independent property (RND, LEX, MEX, LU, MU). Only the LRD2, SH2 and LN2 heuristics employ an alternative start time selection mechanism consistent with their intervention selection

Table 2 Start time selection mechanisms

Mechanism	Start time selection rule
Cheapest	$t'_{i,cheap} = \arg\min_{t \in H}(obj_{aug}(x \cup (i,t)) - obj_{aug}(x))$
Lowest resource demand	$t'_{i,lrd} = \arg\min_{t \in H}(r_{total}(x \cup (i,t)) - r_{total}(x))$
Shortest	$t'_{i,short} = \arg\min_{t \in H} \Delta_{i,t}$
Longest	$t'_{i,long} = \arg\max_{t \in H} \Delta_{i,t}$

Table 3 Repair heuristics

Heuristic	Intervention selection rule
Random (RND)	$i = \underset{i \in I \setminus I_x}{\text{rand}()}$
Cheapest (CH)	$i = \arg\min_{i \in I \setminus I_x} (obj_{aug}(x \cup (i, t'_{i,cheap})) - obj_{aug}(x))$
Most expensive (ME)	$i = \arg\max_{i \in I \setminus I_x} (obj_{aug}(x \cup (i, t'_{i,cheap})) - obj_{aug}(x))$
Lowest res. demand 1 (LRD1)	$i = \arg\min_{i \in I \setminus I_x} (r_{total}(x \cup (i, t'_{i,cheap})) - r_{total}(x))$
Lowest res. demand 2 (LRD2)	$i = \arg\min_{i \in I \setminus I_x} (r_{total}(x \cup (i, t'_{i,lrd})) - r_{total}(x))$
Highest resource demand (HRD)	$i = \arg\max_{i \in I \setminus I_x} (r_{total}(x \cup (i, t'_{i,cheap})) - r_{total}(x))$
Shortest 1 (SH1)	$i = \arg\min_{i \in I \setminus I_x} \Delta_{i,t'_{i,cheap}}$
Shortest 2 (SH2)	$i = \arg\min_{i \in I \setminus I_x} \Delta_{i,t'_{i,short}}$
Longest 1 (LN1)	$i = \arg\max_{i \in I \setminus I_x} \Delta_{i,t'_{i,cheap}}$
Longest 2 (LN2)	$i = \arg\max_{i \in I \setminus I_x} \Delta_{i,t'_{i,long}}$
Least exclusions (LEX)	$i = \arg\min_{i \in I \setminus I_x} \lvert Exc_i \rvert$
Most exclusions (MEX)	$i = \arg\max_{i \in I \setminus I_x} \lvert Exc_i \rvert$
Least used (LU)	$i = \arg\min_{i \in I \setminus I_x} removed_i$
Most used (MU)	$i = \arg\max_{i \in I \setminus I_x} removed_i$

rule. For example, the LRD2 repair heuristic schedules the intervention i to $t'_{i,lrd}$, which causes the lowest increase in total resource demand, regardless of increase in obj_{aug}.

All repair heuristics can also be used as an initial solution construction procedure by simply calling the heuristic $|I|$ times, starting with an empty solution x. This is used in each restart of the ALNS metaheuristic (line 4, Algorithm 1). The repair heuristic selected to serve as a construction procedure is the longest 1 (LN1) heuristic. The tuning process that results in this choice is described in Sect. 5.3.

4.4 Hybridization of repair heuristics

The repair heuristics introduced so far are all deterministic, with the only exception of the trivial RND repair heuristic. The same holds for the start time selection mechanisms. Therefore, applying the same repair heuristic to the same partial solution will, in most cases, produce the same complete solution, which may cause stagnation of the ALNS search process. To prevent this, the selection of the intervention and the selection of the start time is further randomized through the following mechanisms, adopted from Smith and Imeson (2017).

4.4.1 Randomization of start time selection

When selecting an intervention $i \in I \backslash I_x$ to schedule, most repair heuristics consider the insertion costs at the cheapest start time $t'_{i,cheap}$, defined in Table 2. The start time selection mechanism can be easily randomized by adding noise to the insertion cost using the following formula:

$$t'_{i,cheap} = \arg\min_{t \in H}((1 + rand)(obj'_{aug})),$$

where $obj'_{aug} = obj_{aug}(x \cup (i,t)) - obj_{aug}(x)$ is the insertion cost of scheduling an intervention i at time t in a partial solution x, $rand$ is a number generated uniformly randomly from $[0, \eta]$ each time the insertion cost is evaluated, and η is a solver parameter. If $\eta = 0$, the described start time selection mechanism becomes deterministic. Otherwise, $\eta = H$, where $H \in (0, 0.5]$ is a tunable solver parameter. The three remaining start time selection mechanisms could be randomized analogically. However, each is used only in a single repair heuristic; thus, their randomization was omitted.

4.4.2 Randomization of intervention selection

The selection of interventions in some repair heuristics is randomized similarly to the start time. Instead of always selecting an intervention $i \in I \backslash I_x$ that minimizes or maximizes a certain property (e.g., minimizes resource demand in the case of the LRD1 heuristic), the following mechanism is used. All interventions from $I \backslash I_x$ are sorted according to the property currently considered (e.g., lowest resource demand to the highest resource demand for LRD1). Then, an index $k \in \{1, 2..., l\}$ is randomly selected according to the unnormalized probability mass function $[\mu^0, \mu^1, ...\mu^{l-1}]$, where $l = |I \backslash I_x|$ and μ is a solver parameter. After that, the intervention with the $k - th$ lowest value of the currently considered property is selected.

The value of μ determines the behaviour of the repair heuristic, as it controls whether an intervention with the smallest or largest possible property value is most likely to be selected. If $\mu = 0$, the selection process is deterministic, and the intervention with the lowest property value is always selected. When $\mu \in (0, 1)$, the selection strongly favours the few interventions with the lowest property values. The $\mu = 1$ corresponds

Table 4 Hybridized repair heuristics

	$\eta = 0$	$\eta = H$
$\mu = 0$	CH, LRD1, SH1, LEX, LU, ME, HRD, LM1, MEX, MU	H1-RND, H1-CH, H1-LRD1, H1-SH1, H1-LEX, H1-LU, H1-ME, H1-HRD, H1-LM1, H1-MEX, H1-MU
$\mu = M_1$	H2-CH, H2-LRD1, H2-SH1, H2-LEX, H2-LU	H3-CH, H3-LRD1, H3-SH1, H3-LEX, H3-LU
$\mu = M_2$	H2-ME, H2-HRD, H2-LN1, H2-MEX, H2-MU	H3-ME, H3-HRD, H3-LN1, H3-MEX, H3-MU

to a uniform random selection. Finally, $\mu > 1$ favours interventions with the highest property values.

In the solver, two separate values of μ are used to parameterize some of the repair heuristics: $M_1 \in (0, 1)$ and $M_2 > 1$ for the selection of interventions with some minimal, respectively, maximal properties. The value M_1 is used in the heuristics intended to assign a high selection probability to interventions with low property values (e.g., LRD1), while M_2 is used in the heuristics that assign a high selection probability to interventions with high property values (e.g., HRD).

4.4.3 Introduced hybrid heuristics

Two previously described mechanisms were combined to create a large bank of randomized repair heuristics. In addition to the deterministic heuristics described in Sect. 4.3.2, the following randomized hybrid variants were added.

The first set of hybrid heuristics uses only the randomized start time selection mechanism, determined by the parameter $\eta = H$. The intervention selection mechanism remains deterministic, therefore $\mu = 0$. These heuristics are marked with the H1 prefix. The second set uses only the randomization of the intervention selection, determined by the parameter M_1 or M_2. These heuristics are marked with the H2 prefix. The third set, marked with the H3 prefix, uses both randomization mechanisms at the same time. Note that M_1 is used to hybridize the minimizing heuristics (CH, LRD1, SH1, LEX, LU) and M_2 the maximizing (ME, HRD, LM1, MEX, MU). The tuning of the actual values of the parameters H, M_1, and M_2 is described in Sect. 5.3. All introduced heuristics, including the original deterministic variants, are classified in Table 4.

4.5 Repair heuristics speed up

Some repair heuristics select an intervention to schedule based on some time-independent properties of the intervention, such as the number of exclusions (LEX, MEX, and their variants), their previous usage (LU, MU, and their variants), or completely randomly (RND). Most heuristics, however, evaluate some intervention property at all possible intervention start times and select the best start time possible. This was introduced in Sect. 4.3.2 as start time selection mechanisms. As the best

possible start time must be determined for each unscheduled intervention in $i \in I \setminus I_x$, these heuristics have a higher time complexity than those based on time-independent properties.

The properties considered (length, cost, or resource demand) tend to be relatively similar across all possible start times: a very long intervention will probably be more expensive than a very short one, regardless of their start times. Using this assumption, the set of currently unscheduled interventions $I \setminus I_x$ considered by a repair heuristic is reduced before applying the heuristic. For this purpose, the average resource demand, cost, and length are precomputed for all interventions. Then, the repair heuristics consider only a candidate subset of fixed size $I_c \subset I \setminus I_x$, where $|I_c| = l_c$. The interventions in I_c are preselected according to their average values of the property currently considered (e.g., I_c for the CH heuristic contains only l_c interventions from $I \setminus I_x$ with the lowest average cost, while I_c for the HRD heuristic contains only l_c interventions with the highest resource demand).

The value of l_c is defined as

$$l_c = \min(batch\_size * |I|, |I \setminus I_x|),$$

where $batch\_size \in (0, 1]$. The value of $batch\_size$ is parametrized separately for each property: $batch\_size =$ `LENGTH_BATCH` is used for those heuristics that use the duration of the intervention, $batch\_size =$ `COST_BATCH` for those that use cost, and $batch\_size =$ `RD_BATCH` for those that use resource demand.

4.6 Local search

The purpose of the local search is to reach a local optima with respect to multiple neighborhoods, after the current solution x has been partially destroyed and recreated (line 10 of Algorithm 1). It enables further refinement of a current solution, which could not be achieved solely by the main ALNS ruin-and-recreate operation. The local search is controlled by the Randomized Variable Neighborhood Descent heuristic (RVND, Duarte et al. 2018), which is described in Algorithm 2.

Algorithm 2 RVND heuristic

Input: solution x
1: $improved \leftarrow$ **true**
2: **while** $improved$ **do**
3: `shuffle`(N)
4: **for** $k \leftarrow 1$ to k_{max} **do**
5: $improved \leftarrow$ **false**
6: $x' \leftarrow$ `best_improve`$(N_k(x))$
7: **if** $obj_{aug}(x') < obj_{aug}(x)$ **then**
8: $x \leftarrow x'$
9: $improved \leftarrow$ **true**
10: **break**
return x

The RVND repeatedly performs the following process until no improvement is achieved in any of the neighborhoods N. First, the k neighborhoods in N are randomly shuffled (line 3). Then, the neighborhoods in N are sequentially searched for the most improving move (lines 4–6). If a better solution x' is found, the process is restarted (lines 7–10). Otherwise, a locally optimal solution is reached, and the local search terminates. Three local search operators, corresponding to different neighborhoods in N, are specifically proposed for the TMS problem.

1-shift (1SH) reschedules a single intervention i' to a new start time t'. The pair with the best improvement (i', t') is selected according to the following formula:

$$i', t' = \arg \max_{i \in I_c, t \in H} (obj_{aug}(x) - obj_{aug}(x \backslash i \cup (i, t))).$$

The improved solution x' is then obtained as $x' = x \backslash i' \cup (i', t')$. In each iteration of the RVND, the 1SH operator considers a randomly sampled subset $I_c \subset I_x$. The size of I_c is given by $|I_c| = \texttt{ONE\_SHIFT\_LIMIT} * |I_x|$, where $\texttt{ONE\_SHIFT\_LIMIT} \in [0, 1]$ is a tunable solver parameter introduced to control the intensity of the operator.

Random 2-shift (2SH-R) reschedules two randomly selected interventions $i_1, i_2 \in I_x$. The most improving start times $t'_1, t'_2 \in H$ are selected as follows:

$$t'_1, t'_2 = \arg \max_{t_1, t_2 \in H} (obj_{aug}(x) - obj_{aug}(x')).$$

The improved solution is then $x' = x \backslash \{i_1, i_2\} \cup \{(i_1, t'_1), (i_2, t'_2)\}$). The 2SH-R operator is applied $\texttt{2\_SHIFT\_LIMIT}$ times in each RVND iteration, where $\texttt{2\_SHIFT\_LIMIT} \in \mathbb{N}$ is a tunable solver parameter.

Exclusion 2-shift (2SH-E) is a variant of the 2SH-R operator, which reschedules two randomly selected interventions $i_1, i_2 \in Exc$. The 2SH-E operator is applied only if the exclusion penalty e_{pen} of a current solution x is nonzero, and thus some exclusivity constraints are violated. Again, the parameter $\texttt{2\_SHIFT\_LIMIT}$ controls the intensity of this operator.

5 Results and discussion

This section describes the testing setup in Sect. 5.1, the competition datasets provided in Sect. 5.2, and the tuning process in Sect. 5.3. The performance of the proposed ALNS metaheuristic on all competition datasets is documented in Sect. 5, together with a summary of the competition results and a comparison with the qualification method. Finally, Sect. 5.5 provides an in-depth analysis of the behavior of ALNS and evaluates the benefit of its individual components.

5.1 Testing setup

The algorithm is implemented in C++. All results are obtained on a Linux computer with an Intel Core i7-7700 3.60 GHz processor. According to the competition rules, the instances are solved in a short 15-minute run and a long 90-min run. In this paper,

Table 5 Datasets

Dataset A id: $\|I\|, \|C\|, T, \|Exc\|$	Dataset B id: $\|I\|, \|C\|, T, \|Exc\|$	Dataset C id: $\|I\|, \|C\|, T, \|Exc\|$	Dataset X id: $\|I\|, \|C\|, T, \|Exc\|$
A01: 181, 9, 90, 48	B01: 100, 9, 53, 19	C01: 120, 9, 53, 36	X01: 120, 9, 53, 30
A02: 89, 9, 90, 25	B02: 100, 9, 53, 14	C02: 120, 9, 53, 22	X02: 706, 9, 53, 792
A03: 91, 10, 90, 8	B03: 706, 9, 53, 764	C03: 706, 9, 53, 787	X03: 280, 9, 53, 111
A04: 706, 9, 365, 846	B04: 706, 9, 53, 764	C04: 706, 9, 53, 771	X04: 426, 9, 25, 327
A05: 180, 9, 182, 53	B05: 706, 9, 53, 846	C05: 706, 9, 53, 846	X05: 467, 9, 220, 390
A06: 180, 10, 182, 53	B06: 100, 9, 53, 14	C06: 280, 9, 53, 102	X06: 528, 9, 300, 472
A07: 36, 9, 17, 2	B07: 250, 9, 53, 124	C07: 120, 9, 42, 26	X07: 209, 9, 300, 51
A08: 18, 9, 17, 2	B08: 119, 9, 42, 18	C08: 426, 9, 25, 211	X08: 209, 9, 300, 32
A09: 18, 10, 17, 0	B09: 120, 9, 42, 30	C09: 110, 9, 53, 19	X09: 548, 9, 30, 539
A10: 108, 9, 53, 30	B10: 398, 9, 25, 249	C10: 522, 9, 102, 480	X10: 460, 9, 35, 338
A11: 54, 9, 53, 4	B11: 100, 9, 53, 21	C11: 89, 9, 102, 28	X11: 521, 9, 131, 462
A12: 54, 10, 53, 0	B12: 495, 9, 102, 394	C12: 298, 9, 191, 124	X12: 522, 9, 131, 479
A13: 179, 9, 90, 83	B13: 99, 9, 102, 2	C13: 505, 9, 230, 356	X13: 336, 9, 212, 163
A14: 108, 10, 53, 13	B14: 297, 9, 191, 142	C14: 465, 9, 220, 414	X14: 613, 9, 180, 611
A15: 108, 10, 53, 13	B15: 495, 9, 250, 425	C15: 528, 9, 300, 409	X15: 613, 9, 180, 601

30 short runs and 5 long runs are carried out for each instance. The random number generator is seeded with consecutive integers, starting with $1 - \{1, 2, ..., 30\}$ for short runs and $\{1, 2, ..., 5\}$ for long runs. The Best Known Scores (BKS) are taken from the best solutions found in the competition in 90-minute runs. The BKSs were obtained by various methods from other competitors.

5.2 Datasets

Four datasets, each consisting of 15 instances, were provided throughout the competition. Each competition phase was evaluated on a different dataset: sprint phase on A, qualification on A, semifinal on B, and final on C and X. Dataset X was not available before submitting the final program. The following basic properties of each instance are listed in Table 5: number of interventions $|I|$, number of resources $|C|$, planning horizon T, and number of exclusions $|Exc|$. All datasets are publicly available at ROADEF (2020).

5.3 Parameters tuning

The algorithm has numerous parameters, which were tuned specifically for a particular dataset during the competition. Similarly, the usage of individual destroy and repair heuristics, local search operators, and initial construction was also parametrized and tuned. For this purpose, the iterated racing procedure, implemented within the irace package (López-Ibáñez et al. 2016), was used. The results presented in this paper

are generated using a common configuration, which was generated after the competition using all four competition datasets. Although tuning for a particular dataset or even an instance would provide better results, this approach is less prone to overfitting and allows a more objective assessment of the method's performance. The final configuration is provided in Table 6.

5.4 Competition results

This section presents detailed results obtained by the final proposed method on all four competition datasets. Performance in the short and long runs is evaluated separately. For each instance, the basic statistical measures (min score, mean score, and standard deviation) are shown. The results also contain gap values calculated relative to the BKS. The gap is given in percents (%) and calculated as $gap = 100 \times \frac{score - BKS}{BKS}$, where $score$ is either the best value (min gap) or the average value in all runs (mean gap). As the method does not guarantee the finding of a valid solution, a success rate measure is provided for datasets C and X, where the method did not succeed every time. The success rate is given as $100 \times \frac{\text{no. of successful runs}}{\text{total no. of runs}}$ (%). The final method is compared with the qualification method described in Woller and Kulich (2021) on datasets A and B, for which the qualification method was tuned, and with the results of the winner of the competition, team S34, on datasets C and X. Finally, a summary of the final results of the competition is presented, which contains 13 different methods. When different methods are compared, hypothesis testing is performed. For this purpose, the paired t-test is employed. Hypothesis H1 states that the average of gaps of the final ALNS method is better (thus, lower) than that of a reference method, and the confidence level CL (%) of H1 being true is provided.

Tables 7 and 8 show results on the qualification dataset A. ALNS reached BKS (or found a solution within the 0.01% gap) in the majority of instances both in the short and long runs. On average, the mean gap was 0.19% in short runs and 0.10% in long runs. Therefore, the ALNS can be considered highly efficient on this dataset. Both tables also show the results of the method submitted in the qualification phase, which was described in Woller and Kulich (2021). Here, the symbol ↑ indicates that the qualification method performed worse than the final method, whereas the symbol ↓ indicates the opposite and no symbol means achieving an identical result by both methods. It can be seen that the average values of the min and mean gaps obtained by the final method are better than those obtained by the qualification method, although the difference is close to 0.1%. The confidence level of the final method being better than the qualification method is 92.71% (15-minute run), respectively 97.76% (90-minute run).

Tables 9 and 10 show results on the semifinal dataset B. The ALNS generally remains within the 0.76% gap in short runs and 0.59% in long runs. Again, the results of the qualification method are provided. Although the qualification method found better solutions in some cases, the average mean gap of the final method is lower by 1.14% in the short runs and by 0.73% in the long runs, with confidence levels of 89.12%, respectively 86.03%. Although the average mean gap was reduced by half

Table 6 Final configuration

Parameters	Values
ALNS	DEPTH $= 0.134$, ITERS_MAX $= 881$, $\lambda = 0.791$, $\Omega_1 = 89.552$, $\Omega_2 = 26.452$, $\Omega_3 = 0.464$
obj_{aug}	$\beta_{lower} = 63614$, $\beta_{upper} = 35265$, $\gamma = 25881$
Destroy heur	RND, MEX, LRD, SH, MEX, LEX, LU
Repair heur	RND, CH, ME, LRD1, LRD2, LN1, LN2, MEX, MU, H1-ME, H1-RND, H1-MEX, H2-LRD1, H2-HRD, H2-LN1, H2-SH1, H2-MEX, H2-LEX, H2-LU, H3-CH, H3-MEX,H3-SH1,H3-MEX, H3-LEX, H3-MU, H3-LU
Construction	LN1
Hybridization	$H = 0.117$, $M_1 = 0.938$, $M_2 = 190.7$
rep. speed-up	LENGTH_BATCH $= 0.224$, COST_BATCH $= 0.585$, RD_BATCH $= 0.528$
LS operators	1SH, 2SH-E
LS params	ONE_SHIFT_LIMIT $= 0.433$, TWO_SHIFT_LIMIT $= 14$

Table 7 Dataset A—15-min results

Instances		ALNS—final				ALNS—qualification	
Name	BKS	Objective values		Gaps (%)		gaps (%)	
		Min	Mean ± stdev	Min	Mean	Min	Mean
A01	1767.82	1768.23	1768.96 ± 0.53	0.02	0.06	0.05↑	0.12↑
A02	4671.38	4671.38	4671.50 ± 0.28	0.00	0.00	0.00	0.00
A03	848.18	848.18	848.19 ± 0.01	0.00	0.00	0.00	0.00
A04	2085.88	2093.27	2096.01 ± 3.23	0.35	0.49	0.89↑	1.75↑
A05	635.22	635.55	635.85 ± 0.24	0.05	0.10	0.07↑	0.13↑
A06	590.62	591.47	593.08 ± 1.41	0.14	0.42	0.49↑	1.74↑
A07	2272.78	2272.78	2272.78 ± 0.00	0.00	0.00	0.00	0.00
A08	744.29	744.29	744.29 ± 0.00	0.00	0.00	0.00	0.00
A09	1507.28	1507.29	1507.29 ± 0.00	0.00	0.00	0.00	0.00
A10	2994.85	2994.87	2994.87 ± 0.00	0.00	0.00	0.00	0.00
A11	495.26	495.27	495.29 ± 0.04	0.00	0.01	0.00	0.01
A12	789.63	789.64	789.64 ± 0.00	0.00	0.00	0.00	0.00
A13	1998.66	1998.88	1999.21 ± 0.19	0.01	0.03	0.01	0.02↓
A14	2264.12	2264.91	2279.30 ± 12.90	0.03	0.67	0.05↑	0.90↑
A15	2268.57	2269.61	2292.59 ± 14.17	0.05	1.06	0.08↑	0.90↓
Mean				0.04	0.19	0.11↑	0.37↑

Table 8 Dataset A—90-min results

Instances		ALNS—final				ALNS—qualification	
Name	BKS	Objective values		Gaps (%)		gaps (%)	
		Min	Mean ± stdev	Min	Mean	Min	Mean
A01	1767.82	1768.68	1768.83 ± 0.10	0.05	0.06	0.03↓	0.06
A02	4671.38	4671.38	4671.38 ± 0.00	0.00	0.00	0.00	0.00
A03	848.18	848.18	848.18 ± 0.00	0.00	0.00	0.00	0.00
A04	2085.88	2093.68	2096.40 ± 3.36	0.37	0.50	0.40↑	0.81↑
A05	635.22	635.33	635.44 ± 0.07	0.02	0.03	0.05↑	0.07↑
A06	590.62	592.43	594.64 ± 1.87	0.31	0.68	0.45↑	0.80↑
A07	2272.78	2272.78	2272.78 ± 0.00	0.00	0.00	0.00	0.00
A08	744.29	744.29	744.29 ± 0.00	0.00	0.00	0.00	0.00
A09	1507.28	1507.28	1507.28 ± 0.00	0.00	0.00	0.00	0.00
A10	2994.85	2994.85	2994.85 ± 0.00	0.00	0.00	0.00	0.00
A11	495.26	495.32	495.32 ± 0.00	0.01	0.01	0.00↓	0.00↓
A12	789.63	789.63	789.63 ± 0.00	0.00	0.00	0.00	0.00
A13	1998.66	1998.88	1998.95 ± 0.04	0.01	0.01	0.01	0.01
A14	2264.12	2265.03	2265.90 ± 0.65	0.04	0.08	0.03↓	0.31↑
A15	2268.57	2269.80	2270.91 ± 1.32	0.05	0.10	0.04↓	0.30↑
Mean				0.06	0.10	0.07↑	0.16↑

Table 9 Dataset B—15-min results

Instances		ALNS—final				ALNS—qualification	
Name	BKS	Objective values		Gaps (%)		Gaps (%)	
		Min	Mean ± stdev	Min	Mean	Min	Mean
B01	3986.20	4043.93	4063.66 ± 10.53	1.45	1.94	1.58↑	2.42↑
B02	4301.66	4327.58	4340.46 ± 7.16	0.60	0.90	0.45↓	1.04↑
B03	35,277.23	35,657.12	35,719.85 ± 39.95	1.08	1.25	2.04↑	2.61↑
B04	34,826.95	34,831.06	34,833.79 ± 1.70	0.01	0.02	0.03↑	0.06↑
B05	2397.10	2427.13	2431.97 ± 8.84	1.25	1.45	1.10↓	1.16↓
B06	4284.67	4297.10	4308.43 ± 5.69	0.29	0.55	0.43↑	0.79↑
B07	7555.95	7570.17	7587.13 ± 7.51	0.19	0.41	0.22↑	0.49↑
B08	7435.72	7435.72	7435.72 ± 0.00	0.00	0.00	0.00	0.00
B09	7491.75	7599.15	7642.35 ± 20.52	1.43	2.01	1.21↓	2.10↑
B10	10,633.02	10,706.93	10,744.78 ± 14.49	0.70	1.05	1.25↑	1.84↑
B11	3626.03	3640.39	3649.16 ± 4.84	0.40	0.64	0.55↑	0.97↑
B12	37,601.38	37,820.38	37,870.71 ± 31.27	0.58	0.72	1.35↑	14.19↑
B13	5024.49	5025.54	5037.71 ± 3.63	0.02	0.26	0.31↑	0.49↑
B14	11,901.77	11,912.83	11,924.52 ± 7.47	0.09	0.19	0.27↑	0.34↑
B15	22,563.54	22,564.39	22,566.86 ± 1.72	0.00	0.01	0.04↑	0.06↑
Mean				0.54	0.76	0.72↑	1.90↑

Table 10 Dataset B—90-min results

Instances		ALNS—final				ALNS—qualification	
Name	BKS	Objective values		Gaps (%)		Gaps (%)	
		Min	Mean ± stdev	Min	Mean	Min	Mean
B01	3986.20	4044.04	4047.57 ± 4.07	1.45	1.54	1.13↓	1.71↑
B02	4301.66	4321.71	4333.18 ± 8.43	0.47	0.73	0.24↓	0.42↓
B03	35,277.23	35,585.40	35,628.56 ± 26.32	0.87	1.00	1.60↑	1.86↑
B04	34,826.95	34,830.40	34,831.86 ± 0.91	0.01	0.01	0.01	0.02↑
B05	2397.10	2420.33	2425.48 ± 3.01	0.97	1.18	1.03↑	1.08↓
B06	4284.67	4294.41	4300.43 ± 5.13	0.23	0.37	0.03↓	0.31↓
B07	7555.95	7569.85	7577.99 ± 7.90	0.18	0.29	0.02↓	0.34↑
B08	7435.72	7435.72	7435.72 ± 0.00	0.00	0.00	0.00	0.00
B09	7491.75	7610.53	7616.32 ± 6.35	1.59	1.66	0.95↓	1.44↓
B10	10,633.02	10,720.15	10,725.87 ± 4.76	0.82	0.87	1.24↑	1.40↑
B11	3626.03	3639.19	3642.11 ± 2.62	0.36	0.44	0.32↓	0.63↑
B12	37,601.38	37,694.28	37,739.04 ± 26.19	0.25	0.37	0.79↑	10.25↑
B13	5024.49	5033.33	5034.53 ± 1.13	0.18	0.20	0.07↓	0.21↑
B14	11,901.77	11,911.26	11,916.43 ± 6.07	0.08	0.12	0.11↑	0.18↑
B15	22,563.54	22,563.83	22,564.43 ± 0.58	0.00	0.00	0.01↑	0.02↑
Mean				0.50	0.59	0.50	1.32↑

Table 11 Dataset C—15-min results

Instances		ALNS - final					S34
Name	BKS	Objective values		Gaps (%)		Valid	Gap (%)
		Min	Mean ± stdev	Min	Mean	(%)	Min
C01	8515.90	8524.87	8565.86 ± 17.04	0.11	0.59	100	0.00
C02	3539.80	3561.53	3574.81 ± 6.32	0.61	0.99	100	0.06
C03	33,512.26	34,013.56	34,133.22 ± 60.39	1.50	1.85	100	0.00
C04	37,586.31	37,591.07	37,599.35 ± 4.99	0.01	0.03	100	0.00
C05	3166.18	3183.77	3188.11 ± 2.21	0.56	0.69	100	0.03
C06	8394.48	8485.49	8516.27 ± 15.83	1.08	1.45	100	0.02
C07	6083.04	6116.58	6146.59 ± 20.12	0.55	1.04	100	0.04
C08	11,155.64	11,329.41	11,374.24 ± 22.16	1.56	1.96	100	0.06
C09	5585.65	5619.92	5647.14 ± 14.25	0.61	1.10	100	0.28
C10	43,341.84	44,165.64	44,424.34 ± 123.76	1.90	2.50	100	0.00
C11	5749.96	5757.73	5776.58 ± 7.48	0.14	0.46	100	0.00
C12	12,718.79	12,778.96	12,804.79 ± 11.50	0.47	0.68	100	0.02
C13	42,484.56	43,504.11	43,576.51 ± 47.53	2.40	2.57	97	0.01
C14	26,457.11	26,925.26	26,951.30 ± 21.51	1.77	1.87	43	0.04
C15	39,757.55	40,345.14	40,435.46 ± 46.11	1.48	1.71	70	0.01
Mean				0.98	1.30	94	0.04

Table 12 Dataset C—90-min results

Instances		ALNS—final					S34
Name	BKS	Objective values		Gaps (%)		Valid	Gap (%)
		Min	Mean ± stdev	Min	Mean	(%)	Min
C01	8515.90	8528.53	8540.50 ± 8.05	0.15	0.29	100	0.00
C02	3539.80	3558.16	3566.61 ± 6.70	0.52	0.76	100	0.00
C03	33,512.26	33,864.63	33,897.94 ± 42.40	1.05	1.15	100	0.00
C04	37,586.31	37,589.96	37,592.46 ± 2.92	0.01	0.02	100	0.00
C05	3166.18	3182.76	3186.49 ± 3.25	0.52	0.64	100	0.00
C06	8394.48	8483.00	8496.60 ± 12.22	1.05	1.22	100	0.00
C07	6083.04	6112.08	6125.20 ± 7.65	0.48	0.69	100	0.00
C08	11,155.64	11,319.23	11,328.73 ± 5.50	1.47	1.55	100	0.00
C09	5585.65	5618.38	5628.46 ± 12.94	0.59	0.77	100	0.15
C10	43,341.84	43,879.90	43,964.73 ± 97.83	1.24	1.44	100	0.00
C11	5749.96	5755.15	5762.74 ± 5.37	0.09	0.22	100	0.00
C12	12,718.79	12,753.57	12,763.37 ± 9.85	0.27	0.35	100	0.00
C13	42,484.56	43,167.98	43,195.75 ± 26.98	1.61	1.67	100	0.00
C14	26,457.11	26,751.71	26,775.25 ± 20.22	1.11	1.20	100	0.00
C15	39,757.55	40,098.38	40,154.43 ± 50.32	0.86	1.00	100	0.00
Mean				0.73	0.86	100	0.01

when using the final method, the statistical significance decreased compared to dataset A.

Tables 11 and 12 show the results obtained for the final public dataset C. Here, the ALNS occasionally dropped below 100% success rate in short runs (instances C13, C14, and C15), but maintained 100% in long runs. The average minimal and mean gaps are still within 1% of the BKS in long runs and within 1.3% in short runs. Therefore, the quality of the solution can still be considered very good. Both tables also contain minimal gaps obtained by the winning team S34, which produced the majority of the BKSs in the long runs.

Finally, Tables 13 and 14 show the results obtained for the final dataset X. This dataset proves to be more demanding than the first three datasets, especially in terms of constraint satisfaction. The average success rate is 91% in the short runs and 95% in the long runs. Regarding solution quality, the average gaps are close to 2% in long runs and above 2% in short runs. Therefore, X instances seem rather challenging for the ALNS, although decent solutions can still be obtained within given time limits. Again, the winning team S34 found most of the BKSs.

The results of the final phase of the competition, evaluated on datasets C and X, are summarized in Table 15. Some implementations were not shared. Thus, the results are adopted from the competition website (ROADEF 2020). In total, 13 of 74 registered teams advanced to the final phase. For each team, the mean gap, confidence level CL and success rate across both datasets are presented, together with the competition score and the final ranking. The competition score is defined in ROADEF (2020). A team could obtain from 0 to 10 points per instance, 300 in total. A team loses a point for each competitor that achieves strictly better combined objective value in the short and long run on a given instance.

The confidence level CL indicates the probability that the overall mean gap of the proposed ALNS method is better than that of the reference method. Each instance was solved by each method only once in the final phase. Multiple methods, including ALNS, did not reach a 100% success rate on some instances. The results of the proposed ALNS, as submitted to the final phase, are shown under team ID J49 and ranked 8th-9th according to the competition scoring. Due to numerical issues in the submitted implementation, the final version produced a large number of invalid solutions on the hidden dataset X. The results shown under the ID J49* were generated after the competition using the parameter setup from this paper with the discovered issues fixed. CL values are calculated using the J49* results to provide a more accurate statistical comparison.

The CL values are consistent with the final competition ranking. Some of the methods are publicly available at ROADEF (2020), although not yet described in a publication. The winning team S34 combined Mixed Integer Programming (MIP) solver and the custom Local Search, where the MIP was used to find a feasible solution and the Local Search to refine it. The 3rd team S56 combined a custom GRASP metaheuristic with a MIP solver but used GRASP for initial construction and MIP for refining. The 4th team S19 applied purely heuristic A* Local Search, whereas the 5th J73 and 7th S58 focused on MIP relaxation and eventual problem decomposition. No details are available about the remaining submissions.

Table 13 Dataset X—15-min results

Instances		ALNS—final					S34
Name	BKS	Objective values		Gaps (%)		Valid	Gap (%)
		Min	Mean ± stdev	Min	Mean	(%)	Min
X01	4011.38	4079.98	4109.80 ± 16.91	1.71	2.45	100	0.07
X02	32,228.64	33,234.93	33,430.20 ± 79.38	3.12	3.73	100	0.01
X03	8102.59	8242.56	8283.36 ± 18.00	1.73	2.23	90	0.02
X04	11,303.40	11,638.54	11,695.23 ± 24.95	2.96	3.47	100	0.11
X05	22,837.42	23,124.34	23,195.01 ± 32.34	1.26	1.57	33	0.09
X06	47,032.16	47,388.20	47,435.27 ± 22.78	0.76	0.86	97	0.00
X07	13,221.36	13,355.15	13,375.78 ± 14.89	1.01	1.17	100	0.00
X08	13,707.28	13,877.78	13,924.88 ± 27.18	1.24	1.59	100	0.07
X09	20,180.45	21,159.56	21,258.07 ± 56.62	4.85	5.34	100	0.08
X10	17,267.82	17,850.72	17,974.29 ± 59.83	3.38	4.09	100	0.12
X11	39,115.27	39,692.69	39,782.24 ± 57.54	1.48	1.71	57	0.02
X12	47,441.37	48,501.56	48,728.53 ± 120.81	2.23	2.71	100	0.13
X13	15,784.17	15,901.61	15,925.55 ± 15.19	0.74	0.90	100	0.00
X14	79,416.87	80,579.54	80,717.74 ± 81.07	1.46	1.64	100	0.01
X15	45,422.29	46,237.44	46,328.70 ± 64.15	1.79	2.00	87	0.15
Mean				1.98	2.36	91	0.06

Table 14 Dataset X—90-min results

Instances		ALNS—inal					S34
Name	BKS	Objective values		Gaps (%)		Valid	Gap (%)
		Min	Mean ± stdev	Min	Mean	(%)	Min
X01	4011.38	4064.38	4070.37 ± 7.30	1.32	1.47	100	0.00
X02	32,228.64	33,030.34	33,088.62 ± 54.92	2.49	2.67	100	0.00
X03	8102.59	8249.80	8261.69 ± 9.23	1.82	1.96	80	0.00
X04	11,303.40	11,584.36	11,606.69 ± 25.86	2.49	2.68	100	0.00
X05	22,837.42	23148.55	23,173.44 ± 23.08	1.36	1.47	60	0.00
X06	47,032.16	47,370.69	47,418.99 ± 50.71	0.72	0.82	100	0.00
X07	13,221.36	13,317.45	13,329.05 ± 10.80	0.73	0.81	100	0.00
X08	13,707.28	13,811.97	13,850.28 ± 32.76	0.76	1.04	100	0.00
X09	20,180.45	20,856.26	20899.61 ± 38.01	3.35	3.56	100	0.00
, X10	17,267.82	17,751.09	17,787.52 ± 31.47	2.80	3.01	100	0.00
X11	39,115.27	39,652.85	39,664.42 ± 9.49	1.37	1.40	100	0.00
X12	47,441.37	48,203.23	48,611.43 ± 251.29	1.61	2.47	100	0.10
X13	15,784.17	15,904.22	15,919.43 ± 12.54	0.76	0.86	100	0.00
X14	79,416.87	80,533.26	80,669.00 ± 114.40	1.41	1.58	100	0.00
X15	45,422.29	46,297.63	46,380.03 ± 67.05	1.93	2.11	80	0.00
Mean				1.66	1.86	95	0.01

Table 15 Final results of the competition

Team	15-min results			90-min results			Final results	
	Mean ± stdev Gap (%)	CL (%)	Valid (%)	Mean ± stdev (%)	CL (%)	Valid (%)	Score (−)	Rank (−)
S34	0.05 ± 0.06	0	100	0.01 ± 0.03	0.00	100	291	1
S66	0.11 ± 0.13	0	90	0.07 ± 0.08	0.00	100	236	2
S56	0.14 ± 0.16	0	93	0.07 ± 0.08	0.00	93	225	3
S19	0.35 ± 0.28	0	100	0.25 ± 0.26	0.00	97	187	4
J73	0.95 ± 1.81	6.75	100	0.69 ± 0.93	0.64	100	139	5
S68	0.54 ± 0.55	0.01	90	0.54 ± 0.55	0.02	87	139	5
S58	0.80 ± 1.96	6.20	93	0.27 ± 0.25	0.00	83	138	7
J49	0.80 ± 0.49	45.97	67	0.66 ± 0.42	35.04	77	77	8
J49*	1.48 ± 1.07	–	100	1.20 ± 0.82	–	100	–	–
J43	1.34 ± 1.13	79.01	70	1.45 ± 1.49	82.53	97	77	8
J24	3.23 ± 2.06	100	100	2.71 ± 1.94	100.00	100	49	10
J3	3.09 ± 2.08	100	97	1.93 ± 1.52	99.96	100	48	11
S14	2.46 ± 1.95	99.48	83	2.14 ± 1.73	99.79	97	40	12
S28	2.54 ± 2.13	99.53	53	1.99 ± 1.66	99.68	57	13	13

5.5 ALNS analysis

The behaviour of the ALNS is further analyzed in a series of experiments. The goal of this section is to provide insight into the algorithm operation and assess the benefit of individual components.

The first experiment, presented in Table 16, documents the importance of local search, which is not a standard part of ALNS. The following two setups were tested on dataset X: standard ALNS without local search and isolated local search without the ALNS. The results show that the standard ALNS reaches a mean gap of 2.70%, but with a success rate of only 32% overall and 0% for 7 instances out of 15. The isolated local search achieves a slightly better mean gap of 2.23%, but its success rate on all instances is only 12% and it finds a valid solution only for 5 instances. Thus, both setups perform quite poorly in terms of finding valid solutions. Presumably, the standard ALNS provides a robust diversification mechanism but fails to reach valid local optima without the intensification element of the local search.

The second experiment investigates the benefit brought by individual components used in the final tuned setup on dataset X. The efficiency of a component is estimated as the percentage ratio between the number of improving calls and the total calls of the component. A call is considered improving if it leads to a new overall best solution after destroying, repairing, and local search. The efficiency of destroy heuristics on dataset X is summarized in Fig. 2a. One data point corresponds to a single run on a single instance, and each boxplot summarizes data across all instances and runs for a single heuristic. Individual heuristics are sorted from highest to lowest mean efficiency. The mean efficiency of all heuristics is around 10%, with the random destroy heuristic

Table 16 Dataset X—15-min results, separating components

Instances name	standard ALNS			Isolated local search		
	Gaps (%)		valid	Gaps (%)		Valid
	Min	Mean	(%)	Min	Mean	(%)
X01	2.06	2.61	100	3.02	3.58	20
X02	7.21	8.38	30	–	–	0
X03-X05	–	–	0	–	–	0
X06	1.22	1.41	47	–	–	0
X07	1.20	1.56	100	1.81	1.96	100
X08	1.41	1.73	97	1.51	1.70	30
X09	–	–	0	–	–	0
X10	5.22	5.79	17	–	–	0
X11-X12	–	–	0	–	–	0
X13	1.18	1.32	63	1.39	1.48	30
X14	2.12	2.57	27	2.28	2.42	7
X15	–	–	0	–	–	0
Mean	2.70	3.17	32	2.00	2.23	12

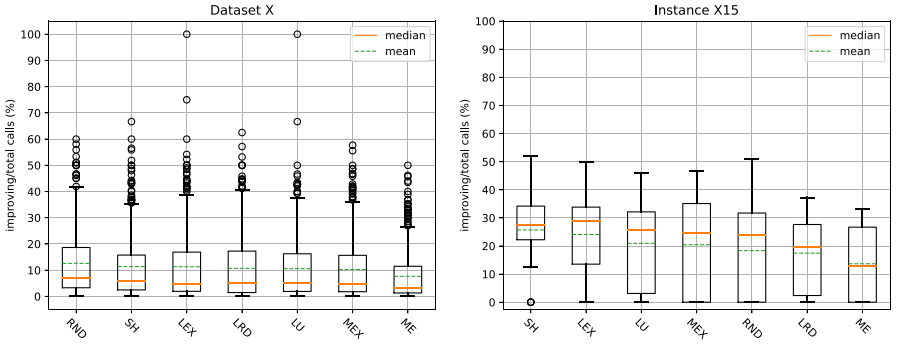


Fig. 2 Destroy heuristics efficiency

being the most successful. Each boxplot contains a large number of outliers, often reaching 50% efficiency. This is also documented in Fig. 2b, which displays the same statistics for instance X15. Here, both the heuristic ranking and their efficiency differ significantly from the averaged data in Fig. 2a. The same analysis was performed for the repair heuristics (Fig. 3) and the local search operators (Fig. 4a). The repair heuristics perform similarly to the destroy heuristics: their mean efficiency is also close to 10% with lots of outliers. In conclusion, all the heuristics not disabled by the automated tuning process have very similar mean efficiency. However, their performance greatly varies from instance to instance, as documented by the large number of outliers. Thanks

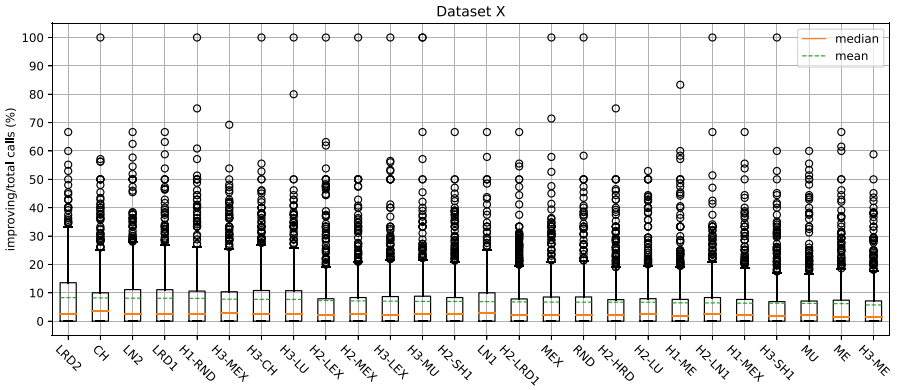
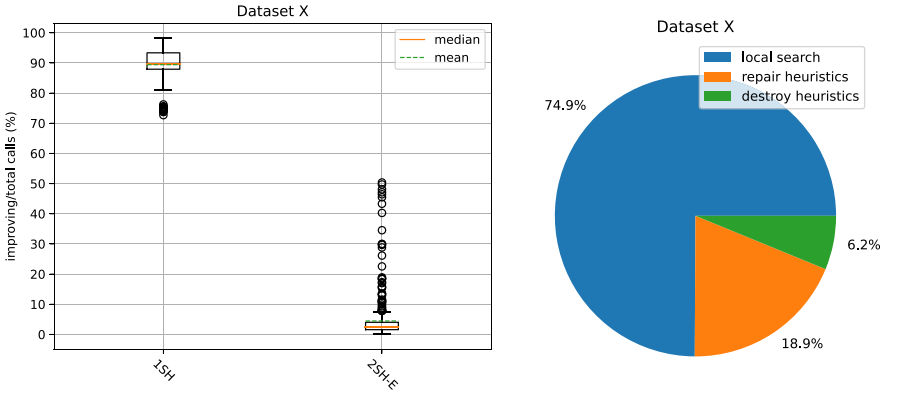


Fig. 3 Repair heuristics eficiency



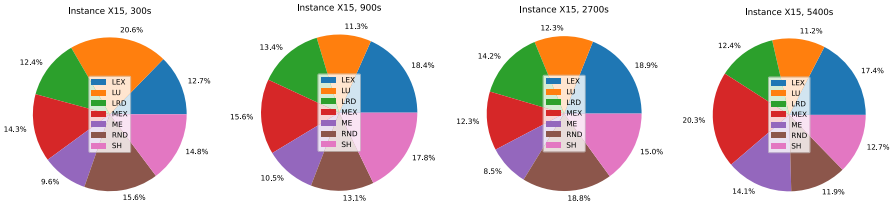
(a) Local search operators efficiency (b) Time spent by components

Fig. 4 Local search

to the ALNS weight-adjusting process, unsuccessful heuristics on a given instance are called less frequently and can be kept in the tuned configuration.

As for local search operators, efficiency was measured by counting improving calls within the RVND heuristic, not the high-level ALNS. The data show that the mean efficiency of the very simple 1-shift operator is close to 90%, whereas the Exclusion 2-shift's is below 5%, although with a large number of outliers.

The third experiment focuses on the CPU time requirements of individual components and is summarized in a single pie chart in Fig. 4b. It shows that 74.9% of the computational budget is consumed by the local search, 18.9% by repairing and only 6.2% by destroying. The ratio between destroying and repairing is given by the higher complexity of repair heuristics, which need to select both an intervention to schedule and its start time, whereas destroy heuristics select only an intervention to remove. The time spent by the local search is controlled by parameters `ONE_SHIFT_LIMIT`



(a) 5 min (b) 15-minute (c) 45 min (d) 90-minute

Fig. 5 Selection weights of destroy heuristics in time

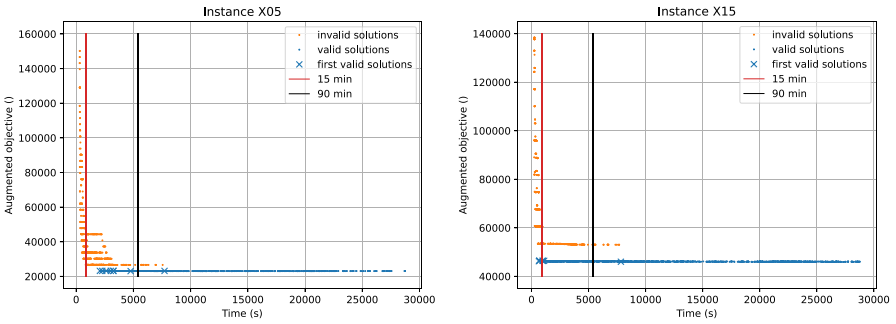
Table 17 Very long runs

Instances		ALNS - final				
Name	BKS	Objective values		Gaps (%)		Valid
		Min	Mean ± stdev	Min	Mean	(%)
X03	8102.59	8205.44	8211.85 ± 4.83	1.27	1.35	100
X05	22,837.42	23,064.99	23,082.82 ± 15.92	1.00	1.07	100
X15	45,422.29	45,916.02	45,992.48 ± 61.12	1.09	1.26	100
Average				1.12	1.23	100

and `TWO_SHIFT_LIMIT`, which were subject to tuning. Thus, the presented ratio is presumably close to an optimal setup.

The fourth experiment attempts to provide insight into the ALNS learning mechanism, which adjusts the selection weights of individual heuristics based on their previous performance. The normalized selection weights of destroy heuristics in different time steps, averaged across multiple runs on a single instance X15, are visualized by pie charts in Fig. 5. It can be seen that the selection weights do not change dramatically and generally oscillate between $10 - 20\%$. The selection weights of individual heuristics do not copy their relative efficiency, shown in Fig. 2b, as their benefit changes over time. The intensity of the learning mechanism is controlled by the parameter $\lambda \in [0, 1]$, where 0 corresponds to no adjusting and 1 to no memory. It is tuned to $\lambda = 0.791$; thus, the weight adjusting is rather aggressive and implements short-term memory.

The final fifth experiment addresses those instances for which a valid solution was not always found even in the long 90-minute runs: X03, X05, and X15. Each of these instances was solved 8 times with a random seed, given a time budget of 8 h. The results of this experiment are provided in Table 17. The solver achieved a 100% success rate in this experiment, thus indicating that it is capable of consistently finding valid solutions for all competition datasets, although not always within the competition-defined time budget. This is illustrated in Fig. 6, which shows the convergence of the solver for instances X05 and X15 over multiple runs. The values of the augmented objective after each improving ALNS iteration are plotted against time. Invalid solutions, valid solutions, and first valid solutions in a given run are distinguished. In both instances, the solver is shown to sometimes work with an invalid solution even after the 90-



(a) (b)

Fig. 6 ALNS convergence

minute time limit. The augmented objective function does not converge smoothly when improving invalid solutions, as it contains a penalty for unmet exclusivity constraints, which changes in discrete steps.

6 Conclusions

An adaptation of the ALNS metaheuristic for a variant of the transmission maintenance scheduling (TMS) problem assigned within the ROADEF Challenge 2020 is described in this paper. Although ALNS is not a commonly applied metaheuristic in TMS problems, the proposed method performed well in the competition of 74 teams, as it finished 6th in the semifinal phase (1st in the Junior category) and 8-9th in the final phase (2nd–3rd in the Junior category). In terms of solution quality, the method consistently finds solutions within the 2% gap on the most difficult competition dataset X and within 1% of the remaining datasets A, B, and C. The main contribution of the approach lies in proposing a large number of destroy and repair heuristics that exploit individual properties and constraints of the problem. These heuristics could easily be reapplied to other scheduling problems with a similar set of properties.

The paper also provides an experimental analysis of the proposed approach. Although the method is tuned, only 10% of the ALNS iterations result in an improved solution. The proposed method employs a penalization mechanism to satisfy multiple hard constraints of the problem. The method is shown to produce valid solutions with a high success rate (above 89% in 90-min runs). The approach could be improved by incorporating a complete procedure to ensure that a valid solution is returned every time, either custom or based on an exact MIP solver. The standard ALNS metaheuristic is extended by a local search, which is shown to be essential for its performance but also to be responsible for consuming most of the computational budget. The convergence rate of the proposed method could be further improved by speeding up the local search, for example, by restricting its operators only to neighborhoods consisting of valid solutions.

Acknowledgements The work on this paper was supported by the Czech Science Foundation grant 23 - 05104S. The work of David Woller was also supported by the Grant Agency of the Czech Technical University in Prague, grant No. SGS23/122/OHK3/2T/13. Computational resources were supplied by the project "e-Infrastruktura CZ" (e-INFRA CZ LM2018140) supported by the Ministry of Education, Youth and Sports of the Czech Republic.

References

Abirami, M., Ganesan, S., Subramanian, S., Anandhakumar, R.: Source and transmission line maintenance outage scheduling in a power system using teaching learning based optimization algorithm. Appl. Soft Comput. **21**, 72–83 (2014). https://doi.org/10.1016/j.asoc.2014.03.015

Artigues, C., Bourreau, E., Jost, V., Kedad-Sidhoum, S., Ramond, F.: Trains do not vanish: the ROADEF/EURO challenge 2014. Ann. Oper. Res. **271**(2), 1091–1105 (2018). https://doi.org/10.1007/s10479-018-3066-x

Back, T., Fogel, D.B., Michalewicz, Z.: Handbook of Evolutionary Computation, 1st edn. IOP Publishing Ltd. (1997)

Ben Hamida, S., Schoenauer, M.: An adaptive algorithm for constrained optimization problems. In: International Conference on Parallel Problem Solving from Nature, pp. 529–538 (2000). Springer

Borůvka, O.: O jistém problému minimálním. Práce Moravské přírodovědecké společnosti II **I**(3), 37–58 (1926)

Burke, E.K., Smith, A.J.: Hybrid evolutionary techniques for the maintenance scheduling problem. IEEE Trans. Power Syst. **15**(1), 122–128 (2000). https://doi.org/10.1109/59.852110

Da Silva, E.L., Schilling, M.T., Rafael, M.C.: Generation maintenance scheduling considering transmission constraints. IEEE Trans. Power Syst. **15**(2), 838–843 (2000). https://doi.org/10.1109/59.867182

Duarte, A., Sánchez-Oro, J., Mladenović, N., Todosijević, R.: In: Martí, R., Pardalos, P.M., Resende, M.G.C. (eds.) Variable Neighborhood Descent, pp. 341–367. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-07124-4_9

El-Sharkh, M.Y.: Clonal selection algorithm for power generators maintenance scheduling. Int. J. Electr. Power Energy Syst. **57**, 73–78 (2014). https://doi.org/10.1016/j.ijepes.2013.11.051

Feng, C., Wang, X., Li, F.: Optimal maintenance scheduling of power producers considering unexpected unit failure. IET Gener. Transm. Distrib. **3**(5), 460–471 (2009). https://doi.org/10.1049/iet-gtd.2008.0427

Froger, A., Gendreau, M., Mendoza, J.E., Pinson, É., Rousseau, L.M.: Maintenance scheduling in the electricity industry: a literature review. Eur. J. Oper. Res. **251**(3), 695–706 (2016). https://doi.org/10.1016/j.ejor.2015.08.045

Geetha, T., Swarup, K.S.: Coordinated preventive maintenance scheduling of GENCO and TRANSCO in restructured power systems. Int. J. Electr. Power Energy Syst. **31**(10), 626–638 (2009). https://doi.org/10.1016/j.ijepes.2009.06.006

He, L., Liu, X., Laporte, G., Chen, Y., Chen, Y.: An improved adaptive large neighborhood search algorithm for multiple agile satellites scheduling. Comput. Oper. Res. **100**, 12–25 (2018). https://doi.org/10.1016/j.cor.2018.06.020

Huang, S.J.: Generator maintenance scheduling: a fuzzy system approach with genetic enhancement. Electr. Power Syst. Res. **41**(3), 233–239 (1997). https://doi.org/10.1016/s0378-7796(96)01194-7

Joines, J.A., Houck, C.R., et al.: On the use of non-stationary penalty functions to solve nonlinear constrained optimization problems with Ga's. In: International Conference on Evolutionary Computation, pp. 579–584 (1994)

Khalid, W., Soleymani, I., Mortensen, N.H., Sigsgaard, K.V.: Ai-based maintenance scheduling for offshore oil and gas platforms. In: 2021 Annual Reliability and Maintainability Symposium (RAMS), pp. 1–6 (2021). https://doi.org/10.1109/RAMS48097.2021.9605794

Kovacs, A.A., Parragh, S.N., Doerner, K.F., Hartl, R.F.: Adaptive large neighborhood search for service technician routing and scheduling problems. J. Sched. **15**(5), 579–600 (2012). https://doi.org/10.1007/s10951-011-0246-9

López-Ibáñez, M., Dubois-Lacoste, J., Pérez Cáceres, L., Birattari, M., Stützle, T.: The irace package: iterated racing for automatic algorithm configuration. Oper. Res. Perspect. **3**, 43–58 (2016). https://doi.org/10.1016/j.orp.2016.09.002

Lu, C., Wang, J., Sun, P.: Short-term transmission maintenance scheduling based on the benders decomposition. In: APPEEC 2012 (2012). https://doi.org/10.1109/APPEEC.2012.6307696

Lv, C., Wang, J., You, S., Zhang, Z.: Short-term transmission maintenance scheduling based on the Benders decomposition. Int. Trans. Electric. Energy Syst. **25**(4), 697–712 (2015). https://doi.org/10.1002/etep.1867

Michel, L., Hentenryck, P.V.: In: Martí, R., Pardalos, P.M., Resende, M.G.C. (eds.) Constraint-Based Local Search, pp. 223–260. Springer, Cham (2018). https://doi.org/10.1007/978-3-319-07124-4_7

Mollahassani-Pour, M., Abdollahi, A., Rashidinejad, M.: Application of a novel cost reduction index to preventive maintenance scheduling. Int. J. Electr. Power Energy Syst. **56**, 235–240 (2014). https://doi.org/10.1016/j.ijepes.2013.11.026

Moro, L.M., Ramos, A.: Goal programming approach to maintenance scheduling of generating units in large scale power systems. IEEE Trans. Power Syst. **14**(3), 1021–1028 (1999). https://doi.org/10.1109/59.780915

Pandžić, H., Conejo, A.J., Kuzle, I., Caro, E.: Yearly maintenance scheduling of transmission lines within a market environment. IEEE Trans. Power Syst. **27**(1), 407–415 (2012). https://doi.org/10.1109/TPWRS.2011.2159743

Pisinger, D., Ropke, S.: Large neighborhood search. In: Handbook of Metaeuristics, pp. 399–419. Springer, Cham (2010). https://doi.org/10.1007/978-1-4419-1665-5_13

Reihani, E., Sarikhani, A., Davodi, M., Davodi, M.: Reliability based generator maintenance scheduling using hybrid evolutionary approach. Int. J. Electr. Power Energy Syst. **42**(1), 434–439 (2012). https://doi.org/10.1016/j.ijepes.2012.04.018

Rifai, A.P., Nguyen, H.-T., Dawal, S.Z.M.: Multi-objective adaptive large neighborhood search for distributed reentrant permutation flow shop scheduling. Appl. Soft Comput. **40**, 42–57 (2016). https://doi.org/10.1016/j.asoc.2015.11.034

ROADEF: Challenge ROADEF/EURO 2020: Maintenance planning Problem (2020). https://www.roadef.org/challenge/2020/en/

Ropke, S., Pisinger, D.: An adaptive large neighborhood search heuristic for the pickup and delivery problem with time windows. Transp. Sci. **40**(4), 455–472 (2006). https://doi.org/10.1287/trsc.1050.0135

Ruiz, M., Tournebise, P., Panciatici, P.: ROADEF Challenge RTE: Grid Operation-Based Outage Maintenance Planning. Technical report, RTE (2020)

Salinas San Martin, L., Yang, J., Liu, Y.: Hybrid NSGA III/dual simplex approach to generation and transmission maintenance scheduling. Int. J. Electric. Power Energy Syst. **135**, 10 (2022). https://doi.org/10.1016/j.ijepes.2021.107498

Saraiva, J.T., Pereira, M.L., Mendes, V.T., Sousa, J.C.: A simulated annealing based approach to solve the generator maintenance scheduling problem. Electr. Power Syst. Res. **81**(7), 1283–1291 (2011). https://doi.org/10.1016/j.epsr.2011.01.013

Schlünz, E.B., Van Vuuren, J.H.: An investigation into the effectiveness of simulated annealing as a solution approach for the generator maintenance scheduling problem. Int. J. Electr. Power Energy Syst. **53**(1), 166–174 (2013). https://doi.org/10.1016/j.ijepes.2013.04.010

Shahidehpour, M., Marwali, M.: Maintenance Scheduling in Restructured Power Systems. Springer, New York (2000). https://doi.org/10.1007/978-1-4615-4473-9

Shaw, P.: Using constraint programming and local search methods to solve vehicle routing problems. In: International Conference on Principles and Practice of Constraint Programming, pp. 417–431. Springer (1998)

Smith, S.L., Imeson, F.: GLNS: an effective large neighborhood search heuristic for the generalized traveling salesman problem. Comput. Oper. Res. (2017). https://doi.org/10.1016/j.cor.2017.05.010

Suresh, K., Kumarappan, N.: Hybrid improved binary particle swarm optimization approach for generation maintenance scheduling problem. Swarm Evol. Comput. **9**, 69–89 (2013). https://doi.org/10.1016/j.swevo.2012.11.003

Than Kyi, M., Maw, S.S., Naing, L.L.: Mathematical estimation for maximum flow in electricity distribution network by Ford–Fulkerson iteration algorithm. Int. J. Sci. Res. **9**(8), 9229 (2019). https://doi.org/10.29322/ijsrp.9.08.2019.p9229

Volkanovski, A., Mavko, B., Boševski, T., Čauševski, A., Čepin, M.: Genetic algorithm optimisation of the maintenance scheduling of generating units in a power system. Reliab. Eng. Syst. Saf. **93**(6), 779–789 (2008). https://doi.org/10.1016/j.ress.2007.03.027

Wang, Y., Zhong, H., Xia, Q., Kirschen, D.S., Kang, C.: An approach for integrated generation and transmission maintenance scheduling considering N-1 contingencies. IEEE Trans. Power Syst. **31**(3), 2225–2233 (2016). https://doi.org/10.1109/TPWRS.2015.2453115

Wen, M., Linde, E., Ropke, S., Mirchandani, P., Larsen, A.: An adaptive large neighborhood search heuristic for the electric vehicle scheduling problem. Comput. Oper. Res. **76**, 73–83 (2016). https://doi.org/10.1016/j.cor.2016.06.013

Woller, D., Kulich, M.: The ALNS metaheuristic for the maintenance scheduling problem. In: Proceedings of the 18th International Conference on Informatics in Control, Automation and Robotics, ICINCO 2021, pp. 156–164 (2021). https://doi.org/10.5220/0010552101560164