Journal of Heuristics

Heuristics and meta-heuristic to solve the ROADEF/EURO challenge 2020 maintenance planning problem

--Manuscript Draft--

Manuscript Number:	
Full Title:	Heuristics and meta-heuristic to solve the ROADEF/EURO challenge 2020 maintenance planning problem
Article Type:	Regular
Keywords:	Iterated local search; Large neighborhood search; Quantile minimization; Grid maintenance planning; 2020 ROADEF/EURO challenge
Corresponding Author:	Hue Chi Lam University of Technology Sydney Ultimo, NSW AUSTRALIA
Corresponding Author Secondary Information:	
Corresponding Author's Institution:	University of Technology Sydney
Corresponding Author's Secondary Institution:	
First Author:	Hue Chi Lam
First Author Secondary Information:	
Order of Authors:	Hue Chi Lam
	Hanyu Gu
	Yakov Zinder
	Thi Thanh Thu Pham
Order of Authors Secondary Information:	
Funding Information:	

Heuristics and meta-heuristic to solve the ROADEF/EURO challenge 2020 maintenance planning problem

Hanyu Gu, Hue Chi Lam*, Thi Thanh Thu Pham and Yakov Zinder

School of Mathematical and Physical Sciences, University of Technology Sydney, 15 Broadway, Ultimo, Sydney, 2007, NSW, Australia.

*Corresponding author(s). E-mail(s): hue.lam@student.uts.edu.au;
Contributing authors: hanyu.gu@uts.edu.au; thithanhthu.pham-1@student.uts.edu.au; yakov.zinder@uts.edu.au;

Abstract

This paper considers the planning problem arising in the maintenance of a power distribution grid. Maintenance works require the corresponding parts of the grid to be shut down for the entire duration of maintenance which could range from one day to several weeks. The planning specifies the starting times of the required outages for maintenance and should take into account the constrained resources as well as the uncertainty involved in the maintenance works which is characterized by the risk values provided by the grid operator. The problem was presented by the French company Réseau de Transport d'Électricité for the 2020 ROADEF/EURO challenge. Several approaches were developed during the competition and all approaches are reported in this paper. We evaluate our approaches on the benchmark instances proposed for the competition. It is reported that the iterated local search metaheuristic with self-adaptive perturbation performed the best.

Keywords: Iterated local search, Large neighborhood search, Quantile minimization, Grid maintenance planning, 2020 ROADEF/EURO challenge

1 Introduction

This paper is concerned with the grid operation-based maintenance planning problem introduced during the 2020 ROADEF/EURO challenge by the French company Réseau de Transport d'Électricité (also known as RTE). The challenge consists of four separate phases, i.e. sprint, qualification, semi-final, and final. The first set of instances (set A) was released at the beginning of the challenge (April 1, 2020) for the sprint and qualification phases. The second set of instances (set B) was published on January 15, 2021 for the semi-final phase. For the ranking of the qualified teams in the final phase, two sets of instances were used, namely sets C (published) and X (hidden). Set C was made available to the participants on April 6, 2021 while the hidden instances (set X) were published after the challenge ended.

RTE, Europe's largest electricity transmission system operator, is responsible for operating, maintaining and developing the electricity transmission system that spans over 105,000 kilometers of lines. The voltages on RTE's electricity network range from 63kV to 400kV [1]. Due to the extreme hazards involved when performing maintenance operations on the high-voltages lines, individual transmission lines have to be shut down for the duration of maintenance. Given that RTE has to handle hundreds of maintenance operations a year and that the maintenance of a transmission line is a long process, it is among the operator's highest priorities to carefully schedule the required outages due to maintenance. In the context of this paper, maintenance work and intervention have the same meaning. The two terms are used interchangeably.

Interventions are carried out by some workforce which is split into teams (or resources), each of which has different sizes and skill sets. The skilled workers are not available during weekends and public holidays. For this reason, resources are not available all the time. Consequently, the duration of an intervention is variable and depends on the time when it starts. Furthermore, resources, e.g. equipment and materials, must be brought to the maintenance site when the intervention commences and removed when it finishes. For this reason, it is expected that the amount of resources required at the beginning and the end of the intervention are higher. Therefore, the resource workload of an intervention is also time-dependent. For every time period, the total consumption of a resource is bounded from below and above.

Certain transmission lines are too close to each other and the corresponding interventions should not take place at the same time. This is because the system is unable to handle the electricity demand if an unexpected outage occurs on another close line during the interventions.

Because the transmission lines must remain switched off for the entire duration of maintenance which could range from one day to several weeks, this causes the electricity system to be weakened and implies a certain risk for RTE. For each intervention and each scenario of grid operation, RTE can compute the risk. According to the energy usage in France, the risk values are dependent on time, because it is less risky to perform interventions in summer (when the demand for electricity is low) than in winter.

The goal of the planning process is to determine a schedule that specifies the starting times of all interventions. The presented optimization procedures take into account the characteristics of the considered problem, including the limitation of resources, and the parameters provided by RTE which reflect the risk associated with maintenance works. RTE conducted studies and decided that the objective function is a weighted sum of two components: the average risk (which is expressed as a monetary cost) related to performing the interventions and the total cost for the deviation from the average risk.

The contributions of this research can be summarized as follows: (1) a new mixed integer linear programming (MILP) formulation of the grid maintenance planning problem, that takes into consideration the risk associated with maintenance works; (2) a new MILP that is based on approximating the quantile term in the objective function; and (3) several solution approaches are developed and compared by means of computational experimentation using instances provided by the competition.

The remainder of this paper is organized as follows. Section 2 provides a review of the related work. Section 3 presents a nonlinear mathematical programming formulation, as well as its linearisation, of the considered problem. In Section 4, we discuss an approximation of the quantile term in the objective function and derive a mixed integer linear programming formulation. In Sections 5 and 6, several heuristic and meta-heuristic algorithms are developed to solve the considered problem. Section 7 presents computational comparisons for the various proposed algorithms using data provided by the 2020 ROADEF/EURO Challenge. Finally, our conclusions are given in Section 8.

2 Related work

Maintenance management is an important function as energy industry organizations are making an effort to ensure the reliability of the electric power system for meeting demand, and is therefore a focus of a large number of scheduling studies [2]. Most of these studies deal with maintenance scheduling of generation units (see, for example [3–5]), while some consider finding an optimal outage schedules for both the generation units and the transmission lines (see, for example [6–8]). For recent literature review on generation units maintenance scheduling, see [9]. A thorough review of maintenance scheduling in the electricity industry is provided in [2]. In this section, we focus on articles that tackle only the transmission-line maintenance scheduling problem.

In [10], the authors present a bi-level outage scheduling model, where the upper-level objective is to maximize the unused transmission capacity while the lower-level objective is to minimize the impact on the functioning of the electricity market. Using equilibrium constraints, the problem is recast as a mixed integer-linear program and solved with CPLEX. This approach schedules considered maintenance during time periods in which its effect on transmission system adequacy is the least. [11] develops a short-term transmission maintenance scheduling model that minimizes the transmission line maintenance cost while satisfying hourly line maintenance constraints, line reservations and system reliability. Using Benders decomposition, the authors decompose the transmission maintenance problem into a maintenance master problem and two types of sub-problems which include transmission sub-problems and voltage sub-problems. [12] formulates the short-term transmission maintenance scheduling problem as a mixed-integer nonlinear program that aims to minimize the maintenance cost and the expected cost of lost load. The model considers failures of transmission components and system reliability as constraints. More recently, [13] proposes a maintenance scheduling optimization model that minimizes the equipment unavailability and avoids simultaneous disconnection of equipment that generates insecure conditions for the operation of the network. The uncertainty in both demand and wind-power generation for transmission line maintenance in long-term horizon is considered in [14]. They develop a two-stage stochastic program to minimize the total expected maintenance cost and cost of lost load of an outage schedule for the transmission lines, under different demand and wind scenarios. All five above-mentioned papers address the transmission-line maintenance scheduling problem with the objective of minimizing maintenance costs subject to achieving a certain required level of reliability, while our study aims at achieving the best level of reliability subject to constraints on resources.

Models for scheduling planned outages for transmission lines that consider the impact of a given outage schedule on maintenance costs and system reliability are presented in [15] and [16]. In particulars, the authors of [15] and [16] develop a machine learning tool for predicting power system operating conditions during the maintenance of grid components. The supervised learning model helps to identify the time periods during which a maintenance outage can be safely accommodated. The approach of [15] and [16] differs from our approach in that they use machine learning proxy for contingency analyses, while we focus on identifying the best outage schedule, considering contingency analyses as a preliminary stage.

In [17], the authors present a maintenance selection and scheduling approach that considers the long-term risk caused by equipment failure and outage consequence in term of overload and voltage security. Similarly, [18] describes a mixed-integer linear formulation for the long-term maintenance scheduling of distribution overhead lines based on the risk of equipment failure and its consequences on network reliability. More recently, [19] proposes an outage scheduling model over mid-term horizon that minimizes the overall risk of carrying out the maintenance over the set of scenarios of future operating conditions. As in our paper, manpower constraints are hard constraints, which must not be violated for an outage schedule to be considered feasible. To solve the problem, the authors design a greedy algorithm that schedules outages one by one, starting from the one with the maximal (negative) impact on system operation. The impact of an outage on system operation is evaluated by Monte Carlo simulations. The proposed approach is able to optimally solve a case study with five interventions and a scheduling horizon of 182 days.

In contrast to [19], our paper focus on problems with up to 528 interventions and a planning horizon of 300 days.

3 Mathematical programming formulation

In this section, we first introduce the notations and describe the objective function. Next, we present a mixed integer linear programming model to find a schedule of interventions, subject to available resources, non-overlapping restrictions between some pairs of intervention, and risk associated with the maintenance works.

3.1 Notations

Planning horizon: The schedule has to be established over a one-year period. The planning horizon is partitioned into intervals of equal length indexed $1, ..., H$ and the set of all time periods is denoted by $T = \{1, ..., H\}$. Depending on the required precision of the schedule, the time step of a schedule can be either a day or a week.

Interventions: Consider a set of N interventions: $I = \{1, ..., N\}$, that have to be planned in the coming year. Each intervention i has a duration $(\Delta_{i,d})$ which assumes integer values and depends on the period d at which it starts. During its processing, at each period t, an intervention $i \in I$ consumes $r_{i,d}^{k,t}$ units of resource k if it starts in period d, where $r_{i,d}^{k,t}$ is a non-negative integer and will be referred to as the resource workload. For each intervention i, the earliest starting period is 1 and the latest is $t_i^{\max}$. Denote by $T_i = \{1, ..., t_i^{\max}\}$ the list of allowed starting periods of intervention i. For any intervention i, any period t, let

$$D_{i,t} = \{d : d + \Delta_{i,d} \geq t + 1, d \leq t\} \cap \ T_i$$

denote the set of starting periods which makes intervention i to be processed during period t. The restriction on which interventions can be carried out simultaneously is given by the set of exclusions, denoted by E. It is a set of triplets (i, j, t) designate that i and j cannot be concurrently performed in period t, where $i, j \in I$ and $t \in T$.

Resources: The processing of each intervention requires M types of resources with different sizes. The set of resources is denoted by $K = \{1, ..., M\}$. In each period t, the total consumption of resource $k \in K$ should be at least l_t^k and should not exceed u_t^k, where both l_t^k and u_t^k are non-negative integers.

3.2 Objective function

According to RTE, the objective function is a weighted sum of two components: the average risk (which is expressed as a monetary cost) related to performing the interventions, and the total cost for the deviation from the average risk. Both criteria are quantified in Euros.

For each intervention, the grid operator characterized the risk related to performing the intervention by some risk values. These values are positive real numbers and are given as input data. For each period t, we are given a set Ω_t of grid operation scenarios. Let $risk_{i,d}^{\omega,t}$ denote the risk value for period t, scenario ω, and intervention i when it starts at d. Also, let $x_{i,d} \in \{0,1\}$ to be 1 if $i \in I$ starts at $d \in T_i$, and 0 otherwise. Then, the first component of the objective function, denoted by Z_1, can be expressed as follows:

$$risk^{\omega,t} = \sum_{i\in I}\sum_{d\in D_{i,t}} risk_{i,d}^{\omega,t}\ x_{i,d}, \quad \omega \in \Omega_t,\ t \in T \tag{1}$$

$$\overline{risk}^t = \frac{1}{|\Omega_t|}\sum_{\omega\in\Omega_t} risk^{\omega,t}, \quad t \in T \tag{2}$$

$$Z_1 = \frac{1}{H}\sum_{t=1}^{H}\overline{risk}^t \tag{3}$$

Alternatively, one could choose to express Z_1 based on $T_i,\ i \in I$:

$$\overline{risk}_{i,d} = \sum_{t=d}^{d+\Delta_{i,d}-1}\frac{1}{|\Omega_t|}\sum_{\omega\in\Omega_t} risk_{i,d}^{\omega,t}, \quad i \in I,\ d \in T_i \tag{4}$$

$$Z_1 = \frac{1}{H}\sum_{i\in I}\sum_{d\in T_i}\overline{risk}_{i,d}\ x_{i,d} \tag{5}$$

For each period t, let $\mathfrak{R}^t = \{risk^{\omega,t}, \omega \in \Omega_t\}$, where $risk^{\omega,t}$ is a sum of risk values in scenario ω of the interventions that are in process in period t, and can be obtained according to (1). Then, the τ-quantile is given by

$$Q_\tau^t = \min\{q \in \mathbb{R} : \exists \mathcal{R} \subseteq \mathfrak{R}^t, |\mathcal{R}| \geq \tau * |\mathfrak{R}^t| \text{ and for } r \in \mathcal{R}, r \leq q\},\ t \in T \tag{6}$$

The second component of the objective function, denoted by Z_2, can be expressed as follows:

$$Z_2 = \frac{1}{H}\sum_{t=1}^{H}\max\{0, Q_\tau^t - \overline{risk}^t\} \tag{7}$$

Let $\alpha \in [0,1]$ denote the weight. The goal is to minimize

$$Z = \alpha Z_1 + (1-\alpha)Z_2 \tag{8}$$

3.3 Mixed integer linear programming formulation

We can eliminate (6) by introducing binary variables $q^{\omega,t}$, auxiliary variables y_t, together with Constraints (13) - (16) to give the mixed integer linear program MILP. The resulting formulation is given as below.

$$\text{(MILP)}\min : \alpha \times \frac{1}{H}\sum_{i\in I}\sum_{d\in T_i}\overline{risk}_{i,d}\ x_{i,d} + (1-\alpha)\times\frac{1}{H}\sum_{t=1}^{H} y_t \tag{9}$$

$$\text{s.t. (1), (2), (4)}$$

$$\sum_{d\in T_i} x_{i,d} = 1, \qquad i\in I \tag{10}$$

$$l_t^k \le \sum_{i\in I}\sum_{d\in D_{i,t}} r_{i,d}^{k,t}\ x_{i,d} \le u_t^k, \quad k\in K,\ t\in T \tag{11}$$

$$\sum_{d\in D_{i,t}} x_{i,d} + \sum_{d\in D_{j,t}} x_{j,d} \le 1, \quad (i,j,t)\in E,\ D_{i,t}\ne\emptyset,\ D_{j,t}\ne\emptyset \tag{12}$$

$$risk^{\omega,t} - Q_\tau^t \le M_t q^{\omega,t}, \qquad \omega\in\Omega_t,\ t\in T \tag{13}$$

$$\sum_{\omega\in\Omega_t}(1-q^{\omega,t}) \ge \lceil \tau\times|\Omega_t|\rceil, \qquad t\in T \tag{14}$$

$$Q_\tau^t - \overline{risk}^t \le y_t, \qquad t\in T \tag{15}$$

$$y_t \ge 0, \qquad t\in T \tag{16}$$

$$q^{\omega,t}\in\{0,1\}, \qquad \omega\in\Omega_t,\ t\in T \tag{17}$$

$$x_{i,d}\in\{0,1\}, \qquad i\in I,\ d\in T_i \tag{18}$$

The objective function (9) is the weighted sum of two components: the average cost for performing the interventions and the total cost for the difference between the τ-quantile and the average risk. Constraint (10) ensures that each intervention must start within the planning horizon. Constraint (11) expresses the requirement that the total resource consumption must be between the limits l_t^k and u_t^k. Constraint (12) enforces that the pairwise exclusive interventions cannot be concurrently performed. Constraints (13) - (16) define the τ-quantile, where M_t is a large number and $\lceil a\rceil$ denotes the smallest integer greater than or equal to a. In particulars, for any period t and scenario ω, if $risk^{\omega,t}$ is larger than Q_τ^t, Constraint (13) ensures that $q^{\omega,t}$ is 1, but arbitrary otherwise. By Constraint (14), the total number of scenarios that have risks below Q_τ^t must be at least $\lceil\tau\times|\Omega_t|\rceil$. The term $\max\{0, Q_\tau^t - \overline{risk}^t\}$ in (7) is substituted by a new decision variable (y_t) together with Constraints (15) and (16). Constraints (17) and (18) state the integrality restriction on the variables $q^{\omega,t}$ and $x_{i,d}$, respectively.

The exclusion representation in the form (12) is widely used in the scheduling domain [20]. However, as the planning horizon and number of interventions grow very large, the existence of a large number of exclusion constraints may render the solution to the MILP model inefficient. By observing the triplets in E, it is not uncommon to have several interventions that are pair-wise exclusive. The exclusion constraints (12) can be accordingly modified to exclude these exclusions together, which reduces the number of exclusion constraints significantly. Formally, the set of pairwise exclusive interventions can be represented as a maximal clique on a undirected *conflict graph* $G_t = (I, E)$, where each vertex corresponds to one intervention in I and each edge between vertices i and j, i.e. $(i, j, t) \in E$, corresponds to the pairwise conflict relationship between interventions i and j in period t. We denote the set of all maximal cliques of G_t by $\mathcal{U}_t = \{U : U \text{ is a maximal clique of } G_t\}$. With the cliques-based constraint, an alternative is to replace Constraint (12) with (19) below.

$$\sum_{i\in U} \sum_{d\in D_{i,t}} x_{i,d} \le 1, \qquad U \in \mathcal{U}_t, t \in T \tag{19}$$

Our experience with the instances proposed by the competition indicates that the resulting problem with cliques-based constraint (19) has fewer constraints.

4 Approximation of quantile term in objective function and iterative updating algorithm

Although the MILP formulation is compact, solving it presents a formidable computational challenge. During the development of solution methods for the qualification phase, we conducted experiments on the MILP formulation and observed that even for a small test instance with 12 scenarios (179 interventions, 9 resources, and 90 time periods), it cannot be solved to optimality by CPLEX in a 2-hour time limit.

Since the team rankings were determined based on the solution values obtained within fifteen minutes execution (with weight 0.8) and one hour and a half execution (with weight 0.2) per instance on the organizers' computer, it is critical that we develop an approximation of the the grid operation-based outage maintenance planning problem that enables it to be solved in a relatively short period of time (e.g. fifteen minutes). With this in mind, we propose a new mixed-integer linear programming relaxation, denoted as A-MILP. The key idea is to approximate the τ-quantile Q^t_τ by the sum of τ-quantile of the individual interventions' risk. The A-MILP can be solved to provide an initial feasible solution for the heuristic and metaheuristic approaches in this paper.

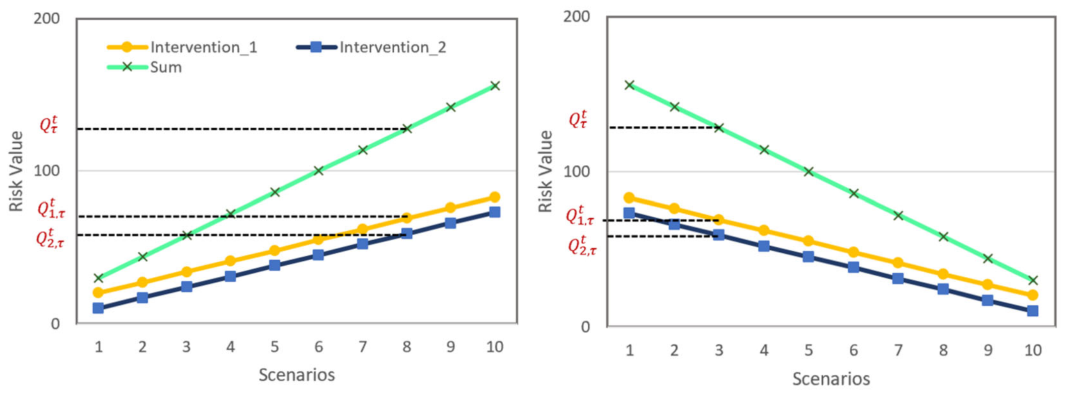
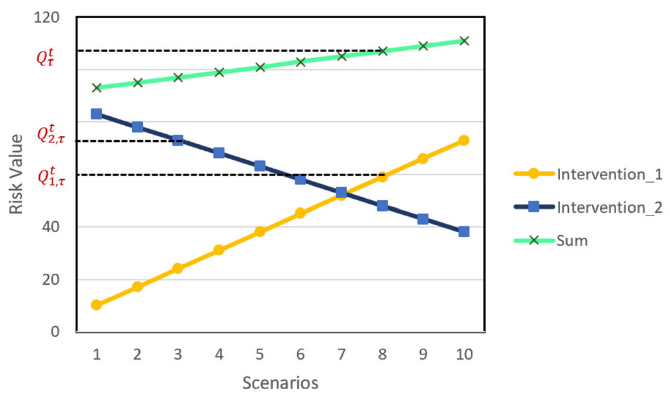


Fig. 1 An example of the interventions having (left) positive and (right) negative slopes. Given a time t and $\tau = 0.8$, in the left figure, we have $Q_\tau^t = 128$ and $\widehat{Q}_\tau^t = Q_{1,\tau}^t + Q_{2,\tau}^t = 69 + 59 = 128$. In the right figure, we have $Q_\tau^t = 128$ and $\widehat{Q}_\tau^t = Q_{1,\tau}^t + Q_{2,\tau}^t = 69 + 59 = 128$.

Fig. 2 An example of the interventions having both positive and negative slopes. Given a time t and $\tau = 0.8$, we have $Q_\tau^t = 107$ but $\widehat{Q}_\tau^t = Q_{1,\tau}^t + Q_{2,\tau}^t = 59 + 73 = 132$.



For each intervention i that starts in d, and for any period t, denote by $Q_{i,d,\tau}^t$ the τ-quantile of the risk values in period t when when i starts in d. The parameter $Q_{i,d,\tau}^t$ can be precalculated from the data. Then, for each period t, the approximation of the τ-quantile Q_τ^t is given by

$$\widehat{Q}_\tau^t = \sum_{i \in I} \sum_{d \in D_{i,t}} Q_{i,d,\tau}^t \, x_{i,d}, \qquad t \in T \tag{20}$$

A plot of the risk value versus scenario can help to visualize the relationship between Q_τ^t and $\widehat{Q}_\tau^t$. Consider the simple case with two interventions whose risk values are a linear function of scenarios (see Figure 1). The horizontal axis represents scenarios, and the vertical axis represents the risk values. The total risk values (green line) is the sum of the individual risk values for the ten scenarios. When both interventions have positive (negative) slopes, the τ-quantile of the total risk and the sum of the τ-quantile of the individual risk are the same, i.e. Q_τ^t and $\widehat{Q}_\tau^t (= Q_{1,\tau}^t + Q_{2,\tau}^t)$ are the same. However, for the combination of interventions with both positive and negative slopes (see Figure 2), the value of $\widehat{Q}_\tau^t$ will likely be an overestimation or underestimation of Q_τ^t.

With the above illustrations and discussions, consider a positive scaling factor $\beta_t, t \in T$, as the parameter compensating for the difference between Q^t_τ and $\widehat{Q}^t_\tau$. This leads to the following approximation of Z_2 in (8)

$$\frac{1}{H}\sum_{t=1}^{H}\max\{0, \beta_t\widehat{Q}^t_\tau - \overline{risk^t}\} \tag{21}$$

With this approximation and a linearisation of (21) in the same way as with (7), we introduce formulation A-MILP as follows:

$$\text{(A-MILP)}\min : \alpha \times \frac{1}{H}\sum_{i\in I}\sum_{d\in T_i}\overline{risk}_{i,d}\; x_{i,d} + (1-\alpha)\times\frac{1}{H}\sum_{t=1}^{H} y_t \tag{22}$$

subject to (1), (2), (4), (10) − (12), (20)

$$\beta_t\widehat{Q}^t_\tau - \overline{risk^t} \le y_t, \qquad t \in T \tag{23}$$

$$y_t \ge 0, \qquad t \in T \tag{24}$$

$$x_{i,d} \in \{0, 1\}, \qquad i \in I,\ d \in T_i \tag{25}$$

For problem with only one scenario, model A-MILP with $\beta_t = 1, t \in T$ is equivalent to the original model MILP. In other words, solving A-MILP to optimality leads to the optimal solution to the considered problem. For problem with more than one scenario, an optimal solution to A-MILP given some vector $\beta = (\beta_1, ...\beta_H)$ is a feasible solution to the original model MILP. The advantage of having A-MILP is that optimal solution is significantly easier to find as the omission of binary variables $q^{\omega,t}$ and constraints (13) and (14) leads to a model containing fewer variables and constraints.

A question of interest is, which values of $\beta_t, t \in T$ in (23) should we use to have a good approximation of the τ-quantile. To answer this question, an iterative procedure is proposed to update β iteratively. Assume $\beta = \beta^\eta$ at the η iteration and σ is the schedule obtained by solving A-MILP with β^η. Given σ, for each period t, we can calculate Q^t_τ and $\hat{Q}^t_\tau$. Based on the difference between $\beta^\eta_t\hat{Q}^t_\tau$ and $\overline{risk^t}$, and the difference between Q^t_τ and $\overline{risk^t}$, β is updated according to (26)

$$\beta^{\eta+1}_t = \begin{cases} \beta^\eta_t & \text{if } \beta^\eta_t\hat{Q}^t_\tau \le \overline{risk^t}, \text{ and } Q^t_\tau \le \overline{risk^t} \\ \overline{risk^t}/\hat{Q}^t_\tau & \text{if } \beta^\eta_t\hat{Q}^t_\tau > \overline{risk^t}, \text{ and } Q^t_\tau \le \overline{risk^t} \\ (1-\gamma)\beta^\eta_t + \gamma Q^t_\tau/\hat{Q}^t_\tau & \text{if } \beta^\eta_t\hat{Q}^t_\tau \le \overline{risk^t}, \text{ and } Q^t_\tau > \overline{risk^t} \\ Q^t_\tau/\hat{Q}^t_\tau & \text{if } \beta^\eta_t\hat{Q}^t_\tau \ge Q^t_\tau > \overline{risk^t} \\ (1-\gamma)\beta^\eta_t + \gamma Q^t_\tau/\hat{Q}^t_\tau & \text{if } Q^t_\tau > \beta^\eta_t\hat{Q}^t_\tau > \overline{risk^t}, \end{cases} \tag{26}$$

For any iteration η and any period t, when both Q^t_τ and $\beta^\eta_t\hat{Q}^t_\tau$ are less than $\overline{risk^t}$ (line 1 in (26)), both $\max\{0, Q^t_\tau - \overline{risk^t}\}$ and $\max\{0, \beta_t\widehat{Q}^t_\tau - \overline{risk^t}\}$

are equal to 0, and it is therefore not necessary to adjust the value of β_t^η. When $\beta_t^\eta \hat{Q}_\tau^t$ is an overestimation of Q_τ^t (lines 2 and 4 in (26)), the value of β_t^η must be reduced. When $\beta_t^\eta \hat{Q}_\tau^t$ is an underestimate of Q_τ^t (lines 3 and 5 in (26)), β_t^η must be increased. A new value at the $(\eta+1)$th iteration can be obtained by $\beta_t^{\eta+1} = (1-\gamma)\beta_t^\eta + \gamma Q_\tau^t / \hat{Q}_\tau^t$, where $\gamma \in (0,1]$. Given the values of $\beta_1^{\eta+1}, ..., \beta_H^{\eta+1}$, the resulting A-MILP model is solved to produce a feasible solution. We terminate the updating procedure when the estimation error (Δ^η) drops below a threshold $\epsilon \geq 0$. That is,

$$\Delta^\eta = \max_{t \in T}\{|\beta_t^{\eta+1} - \beta_t^\eta|\} \leq \epsilon \tag{27}$$

The proposed iterative updating approach, which will be referred to as IterUpdate, does not guarantee to yield an optimal solution to the considered problem since the updating rule for $\beta_t, t \in T$ in (26) only considers the estimation errors for the quantile term in the objective function. In other words, the solution with the smallest estimation error does not necessarily be the solution with the smallest value of the objective function. However, the IterUpdate algorithm (see Algorithm 1) will always yield a feasible solution to the considered problem.

Algorithm 1 Iterative Updating Algorithm (IterUpdate)

```
1: Input: A problem instance
2: Output: A feasible schedule $\sigma$
3: Step 1: Select the initial values for $\beta_t^0, \forall t \in T$
4: Step 2: Solve A-MILP using $\beta_t^0, \forall t \in T$, resulting in a solution $\sigma$.
5: Set $\eta = 1$
6: while stopping criterion (27) is not satisfied and $\eta <$ maximum number of iterations do
7:     Update $\beta_t^\eta, t \in T$ according to (26)
8:     Solve A-MILP using $\beta_t^\eta, t \in T$
9:     Set $\eta = \eta + 1$
10: end while
11: return the best feasible schedule $\sigma$
```

5 Confidence method approaches

The guaranteeing (confidence) approach [21] is a method for solving stochastic optimization problem in which the quantile of the distribution of an objective function is the criterion to be optimized. The problem is known by the name "Quantile Optimization Problem" in the literature. In this section, we develop two heuristic approaches which utilize the idea of confidence approach. The heuristics will be referred to as confidence-method heuristic and critical-scenario confidence-method heuristic, respectively.

In the confidence-method heuristic, we determine which values will be assigned to the binary variables $q^{\omega,t}$, $\omega \in \Omega_t$, $t \in T$ in the original model MILP, using information from an initial solution, which can be obtained using the IterUpdate algorithm described in Section 4. For any solution σ, denote by $\mathscr{C}_{t,\tau}(\sigma) = \{\omega : risk^{\omega,t} \leq Q_\tau^t,\ \omega \in \Omega_t\}$ the confidence set in period t. Then, the corresponding $q^{\omega,t}$ variable in (13) is set to 0 if the scenario belongs to $\mathscr{C}_{t,\tau}(\sigma)$, and to 1 otherwise. The process of finding the confidence sets is applied iteratively. When the confidence-method heuristic reaches the time limit or when it cannot find an improved solution after a maximal permissible number of consecutive iterations, the procedure is terminated and the solution from the last iteration is returned. To simplify notation, we suppress dependence on the given solution σ and simply use $\mathscr{C}_{t,\tau}$ instead of $\mathscr{C}_{t,\tau}(\sigma)$. The MILP model, subject to the imposed confidence sets, is as follows:

$$\text{(C-MILP)} \min : \alpha \times \frac{1}{H} \sum_{i \in I} \sum_{d \in T_i} \overline{risk}_{i,d}\ x_{i,d} + (1-\alpha) \times \frac{1}{H} \sum_{t=1}^{H} y_t \tag{28}$$

subject to (1), (2), (4), (10) – (12),

$$risk^{\omega,t} \leq Q_\tau^t, \quad \omega \in \mathscr{C}_{t,\tau},\ t \in T \tag{29}$$

$$Q_\tau^t - \overline{risk}^t \leq y_t, \quad t \in T \tag{30}$$

$$y_t \geq 0, \quad t \in T \tag{31}$$

$$x_{i,d} \in \{0,1\}, \quad i \in I,\ d \in T_i \tag{32}$$

where (29) imposes the requirements that the risk values corresponding to the scenarios in $\mathscr{C}_{t,\tau}$ must be less than or equal to the τ-quantile. For each period t, the set Ω_t is split into two sets $\mathscr{C}_{t,\tau}$ and $\Omega_t \setminus \mathscr{C}_{t,\tau}$ and the Constraint (13) is replaced by (29). At each iteration of the confidence-method heuristic, we solve the subproblem C-MILP. Since the solution at iteration η is feasible at iteration $\eta+1$, it can be provided to the IP solver as a "warm start". Naturally, each iteration therefore results in a solution no worse than the previous. The confidence-method heuristic is summarized below.

Confidence-method heuristic

Step 0. (Initialization) Generate an initial solution σ by the IterUpdate algorithm described in Section 4.

Step 1. Set $\eta = 1$, construct the confidence sets $\mathscr{C}_{t,\tau}^{\eta}(\sigma)$, $t \in T$ for the initial solution σ.

Step 2. Using $\mathscr{C}_{t,\tau} = \mathscr{C}_{t,\tau}^{\eta}(\sigma)$, $t \in T$, solve the C-MILP to obtain a new solution σ'.

Step 3. If stopping criterion is satisfied, then go to Step **5**.

Step 4. Set $\eta = \eta + 1$ and $\sigma = \sigma'$, construct the confidence sets $\mathscr{C}_{t,\tau}^{\eta}(\sigma)$, $t \in T$ and return Step **2**.

Step 5. Output solution σ'.

Another heuristic based on the confidence approach is the critical-scenario confidence-method heuristic. In the critical-scenario confidence-method heuristic, inequality in the form of (29) is only added to the problem when it is necessary. The reason is because having too many variables fixed as with the confidence-method heuristic, the opportunity for finding an improved solution can be low. Moreover, the resulting C-MILP model is still too large to be solved to optimality in a reasonable time given the large instances.

Given an initial solution σ, one can find the confidence sets $\mathscr{C}_{t,\tau}(\sigma),\ t \in T$. The scenario in the confidence set with largest $risk^{\omega,t}$ value will be used to generate a constraint (29) that is added to the reduced problem. We refer to this scenario as critical scenario. The process of finding the critical scenarios is applied iteratively. The constraints in (29) are incrementally added based on critical scenario at each iteration in an attempt to improve the lower approximation to the τ-quantile Q^t_τ. Let $risk^{\omega^\eta_t,t}$ denote the risk value for period t and critical scenario ω^η_t at iteration η, the reduced problem can be written as

(reduced-MILP)

$$\min : \alpha \times \frac{1}{H} \sum_{i \in I} \sum_{d \in T_i} \overline{risk}_{i,d}\ x_{i,d} + (1-\alpha) \times \frac{1}{H} \sum_{t=1}^{H} y_t \tag{33}$$

subject to (1), (2), (4), (10) − (12), (30), (31), (32)

$$risk^{\omega^i_t,t} \le Q^t_\tau, \quad i = 1, \dots \eta, t \in T \tag{34}$$

The critical-scenario confidence-method heuristic is summarized below.

Critical-scenario confidence-method heuristic

Step 0. (Initialization) Generate an initial solution σ by the IterUpdate algorithm described in Section 4.
Step 1. Set $\eta = 1$, construct the confidence sets $\mathscr{C}^\eta_{t,\tau}(\sigma),\ t \in T$ for the initial solution σ, and find the critical scenarios $\omega^\eta_t,\ t \in T$.
Step 2. Solve the reduced-MILP to obtain a new solution σ'.
Step 3. If stopping criterion is satisfied, then go to Step **5**.
Step 4. Set $\eta = \eta + 1$ and $\sigma = \sigma'$, construct the confidence sets $\mathscr{C}^\eta_{t,\tau}(\sigma),\ t \in T$, find the critical scenarios $\omega^\eta_t,\ t \in T$ and return to Step **2**.
Step 5. Output solution σ'.

6 Iterated local search

Iterated Local Search (ILS) [22] has been widely applied to solve a variety of combinatorial optimization problems and has delivered high-quality solutions (see, for example, [23–26]). In this section, we propose an ILS algorithm for the grid operation-based outage maintenance planning problem. The key idea of the proposed ILS is the self adaptive perturbation strategy, which dynamically modifies the perturbation strength based on the evaluation of the neighborhoods around the local optimum. Failure to improve the local optimum after a certain number of iterations is an indication that the perturbation strength should be amplified. Additionally, a restart strategy is incorporated into the ILS framework, which permits the algorithm to restart the search from the current best solution. This restart strategy can prevent the algorithm from spending too much time in an unpromising region of the search space and help find better solutions for some hard instances. In what follows, whenever this restart strategy is invoked, a new "path" is created.

The general framework of the proposed ILS is outlined in Algorithm 2. In this pseudocode, Z is the objective function (8), and λ and λ_{big} are the notations for perturbation strength. Additionally, the parameter W is the time limit imposed on the algorithm, Y specifies the maximum permissible number of consecutive unsuccessful attempts to improve the current best solution $\hat{\sigma}^*$, and Λ is the upper bound for the perturbation strength.

The proposed ILS starts with the subroutine INITIAL (line 1) which generates a high-quality feasible solution to the original problem. This initial solution is considered as the current best solution. In the pseudocode below, the current best solution of a particular path and over all paths is denoted by $\hat{\sigma}^*$ and σ^*, respectively.

The parameter W (line 8) specifies the time limit for the ILS procedure to find a better solution with respect to the value of the objective function (9) (WHILE loop lines 8 - 33). Each iteration of the WHILE loop starts with a solution σ obtained by applying a perturbation on either $\hat{\sigma}^*$ or σ^*. The perturbed solutions are always produced by a sequential application of two types of perturbation moves (a call of the subroutines PERTURB_SHIFT and PERTURB_SWAP). If, however, a new "path" is invoked as the perturbation strength exceeds the given permissible number Λ, then the perturbed solution is produced by the subroutine PERTURB_SWAP (line 24) only.

Given the perturbed solution, the ILS algorithm attempts to find a better solution using the subroutine SEARCH, which is a sequence of local search procedures. Three neighborhood operators will be used for this purpose: *one-shift*, *two-swap*, and *clique based large neighborhood search (C-LNS)*. Each iteration of the selected operator performs an exhaustive search in the corresponding neighborhood, and selects the solution with the smallest value of the objective function (8).

6.1 Subroutine INITIAL

The subroutine INITIAL for constructing an initial solution required in step 1 of Algorithms 2 includes the following steps. First, a feasible solution is obtained by solving the MILP model or the A-MILP model. The former is always used, except if the problem has a maximum number of scenarios ($\max_{t\in T} \Omega_t$) more than six. If this happens, the feasible solution is obtained using the A-MILP model. Next, the quality of the solution should be improved

Algorithm 2 Iterated local search

```
1: $\sigma \leftarrow$ INITIAL
2: $\sigma^* \leftarrow \sigma$                                  ▷ $\sigma^*$ is the global optimum
3: $\hat{\sigma}^* \leftarrow \sigma$                            ▷ $\hat{\sigma}^*$ is the local optimum
4: $\lambda,\ \lambda_{\text{big}} \leftarrow \lambda^0$
5: $i \leftarrow 0$
6: $\sigma \leftarrow$ PERTURB_SHIFT($\sigma^*$, $\frac{1}{3}\lambda$)
7: $\sigma \leftarrow$ PERTURB_SWAP($\sigma$, $\frac{1}{3}\lambda$)
8: while $time < W$ do
9:     $\sigma \leftarrow$ SEARCH($\sigma$)
10:    if $Z(\sigma) < Z(\hat{\sigma}^*)$ then
11:        $\hat{\sigma}^* \leftarrow \sigma$
12:    end if
13:    if $Z(\sigma) < Z(\sigma^*)$ then
14:        $\sigma^*, \hat{\sigma}^* \leftarrow \sigma$
15:        $i \leftarrow 0$
16:        $\lambda,\ \lambda_{\text{big}} \leftarrow \lambda^0$
17:        $\sigma \leftarrow$ PERTURB_SHIFT($\sigma^*$, $\frac{1}{3}\lambda$)
18:        $\sigma \leftarrow$ PERTURB_SWAP($\sigma$, $\frac{1}{3}\lambda$)
19:    else
20:        if $i > Y$ then
21:            $\lambda \leftarrow \gamma\lambda$                ▷ increase perturbation strength
22:        end if
23:        if $\lambda > \Lambda$ then
24:            $\sigma \leftarrow$ PERTURB_SWAP($\sigma^*$, $\lambda_{\text{big}}$)      ▷ start a new "path"
25:            $\lambda \leftarrow \lambda^0$
26:            $\lambda_{\text{big}} \leftarrow \lambda_{\text{big}} + 1$
27:        else
28:            $\sigma \leftarrow$ PERTURB_SHIFT($\hat{\sigma}^*$, $\frac{1}{3}\lambda$)
29:            $\sigma \leftarrow$ PERTURB_SWAP($\sigma$, $\frac{1}{3}\lambda$)
30:        end if
31:    end if
32:    $i \leftarrow i + 1$
33: end while
34: return $\sigma^*$
```

whenever possible before entering the ILS procedure. For this reason, the confidence method described in Section 5 is used, followed by a further improvement from applying the local search with *one-shift* and *two-swap*.

In our implementation of the subroutine INITIAL, we solve the A-MILP model in two stages. The first stage aims to find a feasible solution with high probability in a short time. This can be achieved by solving the model without an objective function. In the second stage, the populate function of CPLEX is used to generate a pool of solutions. Each solution in the pool is examined and the one with the smallest value of the objective function (8) will be selected.

6.2 Subroutine SEARCH

The big challenge for applying local search idea to the considered problem is the extensive computational effort for assessing the quality of candidate solutions with a large number of scenarios. This is due to the quantile term in the objective function. We tested the sample average approximation method commonly used in the literature, but found that it is not competitive in terms of solution quality. We believe it is important for the local search operators that we can efficiently calculate for a given solution the exact objective function value. Therefore, we propose three simple local search operators: (i) *clique based large neighborhood search (C-LNS)*, (ii) *one-shift*, and (iii) *two-swap*.

Clique based large neighborhood search (C-LNS) - The basic idea of large neighborhood search is to explore a complex neighborhood, aiming to travel across promising search path and find better solutions at each iteration [27]. In the case of *C-LNS*, a move consists of deleting some non-overlapping interventions from the current solution $\tilde{\sigma}$ and then finding the new starting times for these interventions by solving an integer program where the starting times of the remaining interventions are fixed. CPLEX might be able to find better solution than $\tilde{\sigma}$, and if this happens, the move is immediately executed and the search goes on. We discuss how to formulate the integer programming model below.

Let $\tilde{\sigma}$ be the current solution. Denote by U the list of selected non-overlapping interventions. The starting time of the remaining interventions will be fixed, i.e. $x_{i,t} = \tilde{x}_{i,t}, \forall i \in I \setminus U$. The resources consumed by this fixed partial solution is

$$\tilde{r}_t^k = \sum_{i \in I \setminus U} \sum_{d \in D_{i,t}} r_{i,d}^{k,t} \times \tilde{x}_{i,d}, \quad k \in K,\ t \in T \tag{35}$$

Given that the interventions in U are non-overlapping, i.e. cannot be processed concurrently, one can determine the set of allowed starting times $\tilde{T}_i,\ i \in U$ such that the resource constraints and disjunctive constraints are satisfied with respect to the fixed partial solution. That is,

$$\begin{aligned} \tilde{T}_i = &\{d \in T_i \mid\ t \in [d, d + \Delta_{i,d} - 1],\ k \in K,\ \tilde{r}_t^k + r_{i,d}^{k,t} \leq u_t^k, \\ &(i,j,t) \in E, j \notin U,\ t \in [d, d + \Delta_{i,d} - 1], \\ &j \text{ does not overlap with } [d, d + \Delta_{i,d} - 1]\} \end{aligned} \tag{36}$$

Let $\tilde{D}_{i,t} = \{d \in \tilde{T}_i : d + \Delta_{i,d} - 1 \geq t, d \leq t\}$, for $i \in U$. The cumulative planning risk $risk^{\omega,t}_{fixed}$ of the fixed partial solution can be computed as:

$$risk^{\omega,t}_{fixed} = \sum_{i \in I \setminus U} \sum_{d \in \tilde{D}_{i,t}} risk^{s,t}_{i,d} \times \tilde{x}_{i,d} \tag{37}$$

The mean cumulative planning risk at t of the fixed partial solution will be:

$$\overline{risk^{t}_{fixed}} = \frac{1}{|\Omega_t|} \sum_{\omega \in \Omega_t} risk^{\omega,t}_{fixed} \tag{38}$$

The τ quantile of the risk profile of the fixed partial solution at t, denoted by $Q^t_{\tau,fixed}$, is computed according to (6). The objective value at t resulting from the fixed partial solution is:

$$f^t_{fixed} = \alpha \, \overline{risk^{t}_{fixed}} + (1-\alpha) \max\{0, Q^t_{\tau,fixed} - \overline{risk^{t}_{fixed}}\} \tag{39}$$

Assigning intervention $i \in U$ to a starting time $d \in \tilde{T}_i$ leads to changes in the values of $risk^t_{fixed}$, $Q^t_{\tau,fixed}$, and therefore f^t_{fixed}. Equation (40) calculates the change in the objective value, denoted by $\gamma_{i,d}$, for starting intervention i at time $d \in \tilde{T}_i$.

$$\gamma_{i,d} = \sum_{t=d}^{d+\Delta_{i,d}-1} (f^t_{fixed+i} - f^t_{fixed}), \tag{40}$$

where $f^t_{fixed+i}$ denote the objective value at t as a result of including i to the fixed partial solution. The discussion above leads to the following integer programming model:

Model (LNS-IP) :

$$\min \sum_{i \in U} \sum_{d \in \tilde{T}_i} \gamma_{i,d} \; x_{i,d} \tag{41}$$

subject to (35), (36)

$$\sum_{t \in \tilde{T}_i} x_{i,t} = 1, \qquad i \in U \tag{42}$$

$$l^k_t \leq \sum_{i \in U} \sum_{d \in \tilde{D}_{i,t}} r^{k,t}_{i,d} \; x_{i,d} + \tilde{r}^k_t, \qquad k \in K, \; t \in T \tag{43}$$

$$\sum_{i \in U} \sum_{d \in \tilde{D}_{i,t}} x_{i,d} \leq 1, \qquad t \in T \tag{44}$$

$$x_{i,t} \in \{0,1\}, \qquad i \in U, \ t \in \tilde{T}_i \tag{45}$$

At each iteration of *(C-LNS)*, our procedure for selecting the interventions to free includes the following steps. First, the current list (I) of interventions contains all interventions and the list (U) of selected non-overlapping interventions is empty. One intervention from I will be chosen at random and inserted to U. Then, the procedure scans the list I and attempts to remove the interventions which overlap with the intervention just added to U. This procedure terminates when list I becomes empty, i.e. each intervention is either added to U or removed from I (because it overlaps with some interventions in U), or when the list U reaches the required size.

As might be expected, the more interventions whose starting times are fixed, the faster the (LNS-IP) model can be solved, but it is more likely that no improvement could be made in term of the objective function value. On the other hand, the fewer interventions whose starting times are fixed, there is greater opportunity for finding a solution with an improved value of the objective function, but significantly more computational effort might be required for solving the (LNS-IP) model. We propose not to impose a restriction on the size of U so that we can free as many non-overlapping interventions as possible. At the beginning of the first iteration of *(C-LNS)*, the input solution can be provided to the IP solver as a "warm start". In the subsequent iterations, the current best solution can be provided to the IP solver as a "warm start".

One-shift - The neighborhood explored by the operator *one-shift* is comprised of all feasible solutions that can be obtained from a solution σ by assigning a different starting time to a single intervention. A total of $n(H-1)$ possible solutions can be produced. The benefit of using *one-shift* is in the fast evaluation of each solution in the neighborhood. For example, we change the starting time of intervention 3 from time 6 to 1 in Figure 3. The objective value of the time periods marked in the boxes are the same, so we just need to consider the changed objective value for the affected periods, i.e. 1, 2, 3, 6, 7, and 8.

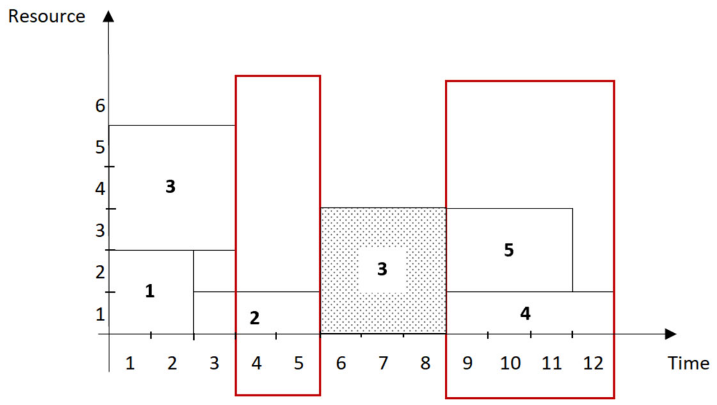


Fig. 3 *one-shift* evaluation.

Two-swap - The *two-swap* is motivated by the classic 2-Opt approach known from the traveling salesman literature [28]. It consists of exchanging the starting times of two interventions i and j, which generates a total of $n(n-1)/2$ possible solutions, where n is the number of interventions. As in *one-shift*, each neighbor solution obtained by *two-swap* can be evaluated efficiently. For example, we swap the starting times of interventions 1 and 5 in Figure 4. The objective value of the time periods marked in the boxes are the same, so we just need to consider the changed objective value for the affected periods, i.e. 1, 2, 3, 9, 10, and 11.

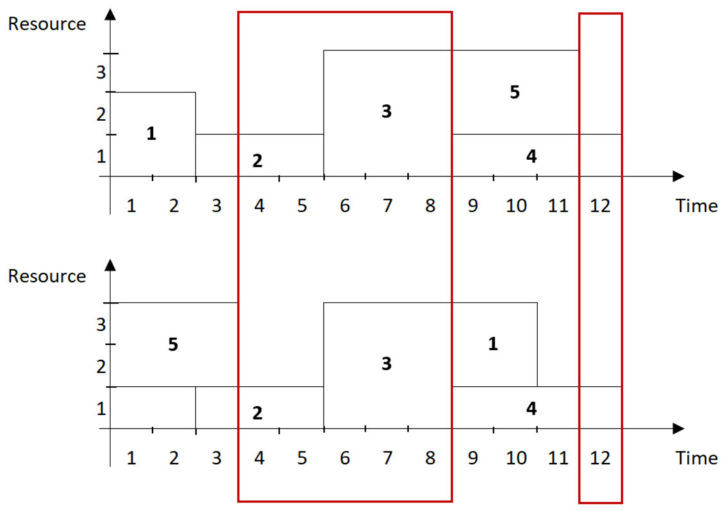


Fig. 4 *two-swap* evaluation.

Let N_0, N_1, N_2 denote the three operators *C-LNS*, *one-shift*, *two-swap*, respectively. For a current solution σ, let $N_i(\sigma)$ denote the output solution produced by the operator N_i, $i = 0, 1, 2$. Let $\hat{\sigma}$ be a solution in the neighborhood of σ with the smallest value of the objective function, then $\hat{\sigma} = N_i(\sigma)$. If such a solution does not exist in the current neighborhood, then $\sigma = N_i(\sigma)$. The algorithm 3 below outlines the subroutine SEARCH for an input solution σ.

Algorithm 3 SEARCH(σ)

```
1:  repeat
2:      $\bar{\sigma} = \sigma$
3:      $\sigma = N_0(\sigma)$
4:  until termination condition met
5:  repeat
6:      $\bar{\sigma} = \sigma$
7:      for $i$ from 1 to 2 do
8:          repeat
9:              $\hat{\sigma} = \sigma$
10:             $\sigma = N_i(\sigma)$
11:         until $Z(\sigma) = Z(\hat{\sigma})$
12:     end for
13: until $Z(\sigma) = Z(\bar{\sigma})$
14: return $\bar{\sigma}$
```

The subroutine SEARCH starts with an iterative local search optimization procedure with operator N_0 (REPEAT loop lines 1 - 4). This loop repeats until the permissible number of iterations is reached. The local optimum found by the local search with operator N_0 is used as an input to the composite local search with operators N_1 and N_2 (REPEAT loop lines 5 - 13). When the local search with the operator N_1 (REPEAT loop lines 8 - 11) finds a local minimum, this local minimum is used as an input to the local search with operator N_2. The subroutine SEARCH terminates if the composite local search algorithm is unable to improve the solution.

6.3 Subroutines PERTURB_SHIFT and PERTURB_SWAP

The perturbation mechanism is responsible for providing a new starting solution for the next iteration of the subroutine SEARCH. It is a crucial component of the ILS procedure as it controls the diversification aspect of ILS. If the amount of perturbation is too large, the algorithm may behave as a random restart method, resulting in much worse local optimal solutions. Conversely, if the perturbations are too small, it may prevent the algorithm to escape from the local optimum. For this reason, the characteristics of the perturbation operator and perturbation strength (i.e. the number of components that are modified in a solution) must be carefully controlled in order to avoid over-disturbance and/or under-disturbance. In this work, we propose an adaptive perturbation mechanism that changes the perturbation strength as the algorithm progresses, and two types of perturbation operators.

The adaptive perturbation mechanism dynamically tunes the perturbation strength using the information about the quality of the neighbor solutions. In the case that the local optimum sits among many good possible solutions and the subroutine SEARCH can progressively find solution with an improved value of the objective function, then the perturbation strength λ remains unchanged to allow for more detailed exploration of the current neighborhood of the search space. In the case that the subroutine SEARCH fails to improve the local optimum after a certain number of consecutive iterations (specified by the parameter Y in line 20 of Algorithm 2), then the perturbation strength λ is increased to $\gamma\lambda$, where $\gamma \geq 1$. This increment will enable new areas of the search space to be explored. When the value of λ reaches the upper bound Λ, further searches becomes unnecessary, so the restart strategy is invoked to replace the current best solution σ^* by a new "path". This "path" concept can enlarge the search space and help find better solution for some hard instances.

The subroutine PERTURB_SHIFT uses the *one-shift* operator, which randomly selects an intervention i, determines all feasible starting times for this intervention in the existing solution, and randomly chooses from this list a new starting time to assign to i. If there are no feasible starting times available, then no change will be made to intervention i. The subroutine PERTURB_SHIFT is terminated when the total number of operations equals to $\frac{1}{3}\lambda$, where λ is the perturbation strength.

The subroutine PERTURB_SWAP uses the *two-swap* operator, which randomly selects a pair of interventions, and examines whether the two may overlap with one another. If they are non-overlapping and swapping the starting times of these two interventions results in a feasible solution, then the swap is applied. The number of swaps depends on the perturbation strength. When the total number of swaps equals to $\frac{1}{3}\lambda$, the subroutine PERTURB_SWAP is terminated.

In the proposed ILS, whenever a new path is created (line 24 in Algorithm 2), the perturbed solution is produced by the subroutine PERTURB_SWAP. Otherwise, the perturbation mechanism consists of the subroutine PERTURB_SHIFT, followed by subroutine PERTURB_SWAP. The perturbation with *one-shift* operator is not as strong as the perturbation with *two-swap* operator. It is evident that a combination of these two types of perturbations introduces increasingly larger degrees of diversification to the search space.

7 Computational results

In this section, we apply the developed heuristic and meta-heuristic algorithms to the four sets of instances used during the ROADEF/EURO challenge 2020. Each set includes 15 test instances. These instances can be downloaded from the Github repository of the competition https://github.com/rte-france/challenge-roadef-2020/. The Roadef 2020 also provided a solution checker - developed in Python3 - that allows the participants to check whether or not a solution is feasible. More about the format of the instances and solution checker can be found on [29].

The code is implemented in cython [30] with C++ STL. IBM ILOG CPLEX 12.10.0 was used to solve the mathematical programming models. The testing

Instance	N	M	H	avg. Ω	E	avg. Δ	τ	α
Sprint and Qualification phases								
A_01	181	9	90	1	81	5.5	0.95	0.5
A_02	89	9	90	120	32	4.6	0.95	0.5
A_03	91	10	90	1	12	4.6	0.95	0.5
A_04	706	9	365	1	1377	8.6	0.95	0.5
A_05	180	9	182	120	87	4.9	0.95	0.5
A_06	180	10	182	1	87	4.9	0.95	0.5
A_07	36	9	17	6	3	1.4	0.50	0.5
A_08	18	9	17	646	4	1.2	0.95	0.5
A_09	18	10	17	6	0	1.8	0.50	0.5
A_10	108	9	53	6	40	1.8	0.50	0.5
A_11	54	9	53	640	4	1.2	0.95	0.5
A_12	54	10	53	6	0	1.1	0.50	0.5
A_13	179	9	90	12	136	5.2	0.50	0.5
A_14	108	10	53	160	22	1.7	0.95	0.5
A_15	108	10	53	320	22	1.7	0.95	0.5
Semi-final phase								
B_01	100	9	53	191	26	10.6	0.90	0.5
B_02	100	9	53	191	19	11.7	0.90	0.5
B_03	706	9	53	63	1192	11.0	0.90	0.5
B_04	706	9	53	63	1192	11.0	0.90	0.5
B_05	706	9	53	63	1377	1.7	0.90	0.5
B_06	100	9	53	255	19	11.7	0.90	0.5
B_07	250	9	53	191	186	10.8	0.80	0.5
B_08	119	9	42	254	37	8.8	0.95	0.5
B_09	120	9	42	127	44	7.5	0.95	0.5
B_10	398	9	25	192	344	5.1	0.80	0.5
B_11	100	9	53	191	34	11.1	0.90	0.5
B_12	495	9	102	63	570	24.8	0.95	0.5
B_13	99	9	102	159	4	24.6	0.90	0.5
B_14	297	9	191	95	207	47.1	0.80	0.5
B_15	495	9	250	63	665	52.2	0.80	0.5
Final phases								
C_01	120	9	53	191	54	11.1	0.95	0.5
C_02	120	9	53	191	43	11.4	0.80	0.5
C_03	706	9	53	63	1223	10.9	0.85	0.5
C_04	706	9	53	63	1194	11.0	0.90	0.5
C_05	706	9	53	63	1377	1.7	0.95	0.5
C_06	280	9	53	191	183	11.1	0.80	0.5
C_07	120	9	42	126	38	7.7	0.95	0.5
C_08	426	9	25	192	340	5.0	0.80	0.5
C_09	110	9	53	191	38	11.7	0.90	0.5
C_10	522	9	102	63	705	24.9	0.95	0.5
C_11	89	9	102	191	35	27.0	0.90	0.5
C_12	298	9	191	95	195	47.0	0.80	0.5
C_13	505	9	230	63	53	58.9	0.95	0.5
C_14	465	9	220	95	620	54.5	0.85	0.5
C_15	528	9	300	51	624	74.7	0.95	0.5
X_01	120	9	53	191	48	10.9	0.80	0.5
X_02	706	9	53	63	1234	11.0	0.85	0.5
X_03	280	9	53	191	162	10.7	0.80	0.5
X_04	426	9	25	188	490	5.1	0.80	0.5
X_05	467	9	220	95	604	55.3	0.85	0.5
X_06	528	9	300	50	703	77.7	0.95	0.5
X_07	209	9	300	63	80	74.9	0.90	0.5
X_08	209	9	300	63	57	75.0	0.90	0.5
X_09	548	9	30	156	820	6.5	0.80	0.5
X_10	460	9	35	159	527	7.3	0.95	0.5
X_11	521	9	131	63	725	32.4	0.95	0.5
X_12	522	9	131	63	723	33.1	0.95	0.5
X_13	336	9	212	95	248	54.7	0.90	0.5
X_14	613	9	180	63	951	47.3	0.95	0.5
X_15	613	9	180	63	917	45.8	0.95	0.5

Table 1 Instances characteristics in ROADEF/EURO 2020 Challenge.

system was a cluster with Intel Xeon E-2288G 3.7GHz 8 cores CPU with 64GB RAM, running Red Hat Enterprise Linux.

Table 1 summarizes the main characteristics of the 60 instances. In Table 1, N is the number of interventions, M is the number of resources, H is the length of planning horizon, avg. Ω is the average number of scenarios, E is the number of exclusions, avg. Δ is the average duration of interventions in all allowed starting times, τ is the quantile level for the planning risk, and α is the weight of the first objective (Z_1). avg. Ω is calculated as $1/H\sum_{t=1}^{H}|\Omega_t|$, while avg. Δ is calculated as $1/N\sum_{i=1}^{N}(1/t_i^{\max}\sum_{t=1}^{t_i^{\max}}\Delta_{i,t})$.

For the parameter $M_t, t \in T$ in (13), a choice of very large M can lead to slow progress in solving the MILP due to weak relaxation. On the other hand, if M is too small, a valid choice of $risk^{\omega,t}$ may violate the constraint even when $q^{\omega,t} = 1$. With this in mind, in our implementation, we use one M parameter for each time t and compute the values using the data for risk and resources.

7.1 Model MILP vs. Model A-MILP

We first evaluate the proposed models MILP and A-MILP, by solving the instances in dataset A by CPLEX. Table 2 summarizes the outcomes. The columns 'Time (s)', 'UB', 'LB', and 'Gap (%)' report the running time (in seconds), upper bounds, lower bounds, and gaps obtained by CPLEX, respectively. For model A-MILP, parameter β_t, $\forall t \in T$ was fixed at 1; the objective values of the solutions are computed according to (8) and reported under the column titled 'Z'; the last column 'Gap (%)' reports the deviation of the objective value from LB and is calculated as Gap (%) = $(Z-\text{LB})/\text{LB}*100$. For both models, CPLEX stopped after reaching a 2 hours limit or when an optimal solution was found.

Instance	MILP				A-MILP		
	Time (s)	UB	LB	Gap (%)	Time (s)	Z	Gap (%)
A_01	2	**1767.81561**	1767.81561	0	2	1767.81561	0
A_02	7200	4673.95911	2112.63630	54.80	4	4736.84755	124
A_03	1	**848.17861**	848.17861	0	0.4	848.17861	0
A_04	74	**2085.87605**	2085.87605	0	68	2085.87605	0
A_05	7200	638.44659	594.46740	6.89	4	645.42793	8.57
A_06	4	**590.62359**	590.62359	0	3	590.62359	0
A_07	0.3	**2272.78227**	2272.78227	0	0.5	2280.60656	0.34
A_08	7200	744.70441	692.65000	6.99	0.2	749.95088	8.27
A_09	0.2	**1507.28478**	1507.28478	0	0.1	1525.64978	1.21
A_10	4	**2994.84873**	2994.84873	0	0.3	3016.16708	0.71
A_11	7200	495.86537	461.75280	6.88	2	504.78716	9.31
A_12	2	**789.63492**	789.63492	0	0.7	789.78896	0.02
A_13	7200	1998.90557	1998.84030	0.003	12	2004.39403	0.28
A_14	7200	2275.50198	2085.27950	8.35	3	2295.81830	10.10
A_15	7200	2290.71433	2072.28270	9.53	4	2302.47084	11.11

Table 2 Comparison of models MILP and A-MILP on dataset A instances.

The results in Table 2 show that CPLEX obtained optimal solutions to MILP when avg. $\Omega \leq 6$. The smallest instance A_09 is solved to optimality in less than 1 second while the largest instance A_04 requires approximately

74 seconds. As the number of scenarios in an instance increases, the required computational effort for a solution of MILP grows rapidly. For the instances where CPLEX could not produce an optimal solution in 2 hours, the average gap was 13.35%. A_02 has the largest gap of 54.8% among the instances in dataset A. Further investigation into the characteristics of A_02 suggests that the difficulty of this instance is due to its high risk values. When A-MILP is used, CPLEX can solve all instances to optimality in less than 100 seconds. The times taken to solve A-MILP is on average 71% less than MILP, but solution quality is on average 0.59% worse than MILP.

7.2 Evaluation of the IterUpdate algorithm

We conduct an experiment to assess the effect of the initial values $\beta_t^0,\ t \in T$ (in line 3 of Algorithm 1) on solution quality of the IterUpdate algorithm. We initialized the parameter $\beta_t^0,\ t \in T$ with three different settings, i.e. 1, 0.01, and $U[0,1]$. For all instances, the parameter γ in (26) was set to 0.6. Termination criteria is a 900 second time limit (excluding the time to read the data file) or the smallest estimation error of less than 1×10^{-5} or the number of iterations without improvement of no more than 20. For each iteration within the heuristic, CPLEX stopped after reaching a 500 seconds limit or when the problem is solved to within 1%-optimality.

We use two criteria of "Number of best obtained solutions (N_{best})" and "Average relative percentage time ($ARPT$)" to compare the results obtained for different settings of β_t^0. The metric $ARPT$ is obtained as follows: first, we compute the average computation time, denoted by ACT, for all three settings on the same instance; then, for each instance, we find the relative percentage computation time of a setting, denoted by RPT, using the formula $RPT = (Time(s) - ACT)/ACT \times 100$; and finally, $ARPT$ can be obtained by averaging the RPT of all instances in one dataset. It is possible that CPLEX cannot find a solution in 500 seconds for some large instances (e.g. dataset X). If this happens, the instance is excluded from the comparison.

A summary of the parameters tuning for the IterUpdate algorithm can be found in Table 3. The highlighted numbers denote the outperforming values. For dataset A, the random initialization obtains better solutions than the other two settings at a cost of longer computing time. For dataset B, both settings of 1 and 0.01 achieve the same number of best obtained solution of 7 but the computing time of the former is significantly shorter. $\beta_t^0 = 1$ is the superior setting for instances of dataset C, while $\beta_t^0 = 0.01$ is more suitable for dataset X instances. Overall, $\beta_t^0 = 0.01$ is the best setting in term of N_{best}, and $\beta_t^0 = 1$ is the second best. However, the latter takes nearly a hundredfold less in computing time. To conclude, in terms of solution quality and time, $\beta_t^0 = 1$ is the preferred setting for the IterUpdate algorithm. Therefore, the IterUpdate algorithm with $\beta_t^0 = 1, \forall t \in T$ is used to generate initial solutions in the remainder of this study.

We conclude this section with Figure 5 illustrating the convergence of the approximate Z_2 to the true Z_2 for instances A_02 and A_11.

Metric	Dataset	$\beta^0 = 1$	$\beta^0 = 0.01$	$\beta^0 =$ random
N_{best}	A	4	8	**10**
	B	**7**	**7**	3
	C	**8**	4	4
	X	1	**9**	4
	Sum	20	**30**	19
ARPT	A	**-17.45**	6.77	10.68
	B	**-64.00**	76.02	-12.02
	C	**-51.28**	39.44	11.84
	X	**-45.05**	26.45	18.60
	Average	**-44.44**	37.17	7.27

Table 3 Summary of $\beta_t^0, \forall t \in T$ for the IterUpdate algorithm on four datasets.

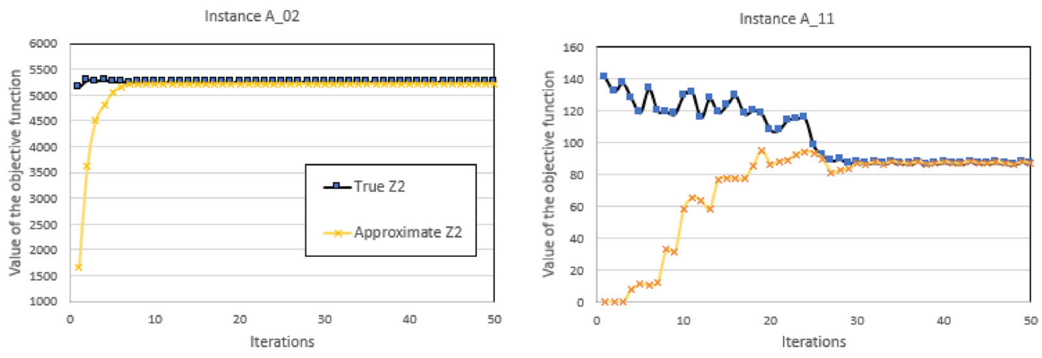


Fig. 5 Convergence of the approximate Z_2 to the true Z_2.

7.3 CM-heuristic vs. cs-CM heuristic

In the following, we compare the performance of CM-heuristic and cs-CM heuristic on the four datasets, the results of which are provided in Tables 4 - 7. We report the change in objective value which is given by subtracting the objective value of the final solution from the objective value of the initial solution; the total time (in seconds) which includes time for running the methods but does not include time for generating the initial solution; the average time spent by CPLEX (in seconds); the average MIP relative gap by CPLEX (%); the average number of constraints of the MIP models over all iterations; and total number of iterations the procedure took to terminate.

For the sake of brevity, these tables do not present results of the instances, where both CM and cs-CM fail to improve the initial solutions. To ensure fair comparisons, we initialize both methods with the same initial solution, i.e. we use the solution obtained by CPLEX for the IterUpdate algorithm with $\beta_t^0 = 1$. The CM-heuristic terminates when the solution has converged, whereas the cs-CM heuristic is to stop after 10 iterations. For both approaches, CPLEX is used to solve the resulting mixed integer linear programming models and a time limit of 100 seconds is imposed on CPLEX. For the first iteration of both methods, we use the initial solution to warm start CPLEX. For iteration

Instance	Method	Change in objective value	Total time	Average solution time	Average gap	Average constraints	Total Iterations
A_02	CM	57.45	408	100	0.073%	13468	4
	cs-CM	58.94	320	32	0.001%	2964	10
A_05	CM	6.89	230	100	0.775%	28980	2
	cs-CM	3.75	842	84	0.334%	7560	10
A_07	CM	5.43	0.5	0.2	0.000%	348	2
	cs-CM	5.43	0.3	0.0	0.000%	254	10
A_08	CM	0.14	2.9	0.7	0.000%	11192	2
	cs-CM	0.00	1.5	0.1	0.000%	250	10
A_09	CM	18.02	0.1	0.0	0.000%	318	2
	cs-CM	18.02	0.2	0.0	0.000%	223	10
A_10	CM	16.83	1.2	0.3	0.001%	1685	3
	cs-CM	16.15	1.8	0.1	0.001%	1394	10
A_11	CM	1.52	146	34	0.001%	34628	4
	cs-CM	0.00	6.2	1	0.001%	818	10
A_12	CM	0.34	0.5	0.1	0.000%	991	2
	cs-CM	0.00	1.2	0.1	0.000%	707	10
A_13	CM	1.76	47	15	0.001%	7198	3
	cs-CM	2.04	68	7	0.001%	6162	10
A_14	CM	29.47	403	100	0.230%	9672	4
	cs-CM	1.65	367	36	0.001%	1344	10
A_15	CM	19.99	508	100	0.350%	18139	5
	cs-CM	0.00	434	43	0.001%	1340	10

Table 4 Results obtained by CM-heuristic and cs-CM heuristics for instances in dataset A.

$\eta > 1$, we use both the current best solution and the solution from iteration $\eta - 1$ to warm start CPLEX.

Results in Table 4 indicate that the CM-heuristic works well on small instances, quickly finding a better solution and exiting in a few iterations. This is because CPLEX can easily solve these small problems to optimality in 100 seconds. With cs-CM, at the early iterations of this method, the solutions are often of poor quality because the estimation of the quantile term in the objective function is not accurate. As more constraints are added to the model in consecutive iterations, the discrepancy between the actual objective value of model objective value reduces, and therefore, CPLEX can start to improve the initial solution.

Results in Table 5 - 7 indicate that, for the large instances in dataset B, C, and X, cs-CM outperform CM-heuristic in terms of solution quality. For example, for X_07, cs-CM yields an improvement of about 0.05% as compared to the initial solution, whereas CM-heuristic is not able to give any improvement on this instance. Solution time can grow with the number of iterations, and so cs-CM, which could stop only after a predetermined number of iterations, i.e. 10 iterations, can require more computational effort. However, note that, for example, for X_07, the number of iterations of cs-CM is 10 times that of CM, yet the solution time only grew 4 times. This is because solving 10 smaller problems (cs-CM with 11002 constraints on average) can be more efficient than solving one large problem (CM with 29633 constraints).

7.4 Comparison of the ILS with benchmark results

We test the proposed ILS algorithm and compare its performance with the best known results from other participants. As in the competition, we run the proposed ILS with a time limit of 15 minutes and another one of one hour and a half. Tables 8 - 11 report the results on the four datasets, respectively. The highlighted numbers denote the outperforming values. In these tables, 'Best' is the best known results from other participants; 'ILS' is the results from our

Instance	Method	Change in objective value	Total time	Average solution time	Average gap	Average constraints	Total Iterations
B_02	CM	0.86	214	101	0.430%	11186	2
	cs-CM	0.00	908	90	1.370%	1256	10
B_03	CM	9.67	324	100	0.060%	22799	3
	cs-CM	15.81	1022	100	0.053%	19592	10
B_04	CM	19.92	324	100	0.008%	22799	3
	cs-CM	16.91	923	91	0.005%	19589	10
B_05	CM	4.27	412	100	0.103%	25542	4
	cs-CM	3.29	923	91	0.055%	22351	10
B_09	CM	1.34	41	18	0.000%	6442	2
	cs-CM	1.33	28	3	0.001%	1096	10
B_12	CM	0.00	157	101	100%	23257	1
	cs-CM	18.79	962	92	0.009%	17001	10

Table 5 Results obtained by CM-heuristic and cs-CM heuristics for instances in dataset B.

Instance	Method	Change in objective value	Total time	Average solution time	Average gap	Average constraints	Total Iterations
C_03	CM	17.77	416	100	0.070%	22735	4
	cs-CM	18.06	1011	100	0.040%	19533	10
C_04	CM	27.75	715	100	0.013%	23299	7
	cs-CM	22.06	914	90	0.007%	10107	10
C_05	CM	4.55	807	100	0.110%	25542	8
	cs-CM	1.22	1004	100	0.137%	22320	10
C_10	CM	0.00	145	100	100%	27071	1
	cs-CM	25.74	964	94	0.005%	20817	10
C_11	CM	0.00	86	63	0.001%	22015	1
	cs-CM	0.44	26	2	0.001%	2730	10

Table 6 Results obtained by CM-heuristic and cs-CM heuristics for instances in dataset C.

Instance	Method	Change in objective value	Total time	Average solution time	Average gap	Average constraints	Total Iterations
X_02	CM	6.48	217	100	0.088%	23715	2
	cs-CM	0.00	1012	100	0.113%	20472	10
X_07	CM	0.00	268	124	100%	29633	1
	cs-CM	7.04	1051	99	0.032%	11002	10
X_11	CM	0.00	165	101	100%	34672	1
	cs-CM	20.51	1029	99	0.030%	26643	10

Table 7 Results obtained by CM-heuristic and cs-CM heuristics for instances in dataset X.

proposed ILS algorithm; and '%Diff' is the percentage difference calculated as %Diff = (ILS − Best)/Best × 100.

For the instances with avg. $\Omega \leq 6$, initial solution to the iterated local search in Algorithm 2 is obtained by solving the (MILP). For all other instances, the IterUpdate algorithm with $\beta_t^0 = 1$ is used to provide the initial solution. A time limit of 500 seconds is given to the IterUpdate algorithm. At each iteration of the local search LS(*C-LNS*), the set of non-overlapping tasks U is formed in the following way: a task u is chosen at random from the set I; The remaining tasks are removed from I if they overlap with u; the procedure continues until set I is empty. At each iteration, the model (LNS-IP) is solved by CPLEX with a time limit of 100 seconds.

The parameters of ILS are set as follows. The value of W is calculated using the formula $W = \mathcal{T} - \mathcal{T}_R - \mathcal{T}_I$, where $\mathcal{T}$ is either 15 minutes or 1.5 hours, $\mathcal{T}_R$ is the amount of time it takes to read the instance, and $\mathcal{T}_I$ is the time used for obtaining the initial solution. The initial value of perturbation strength (λ) is 3. We set $\Lambda = 12$, $\gamma = 1.2$, and $Y = 10 + N/250$, where N is the number of interventions. Note that Y will be rounded to the nearest integer if the division $N/250$ gives a non-integer value.

Instance	15 minutes			1.5 hours		
	Best	ILS	%Diff	Best	ILS	%Diff
A_01	1767.81561	**1767.81561**	0.00%	1767.81560	**1767.81561**	0.00%
A_02	4671.37661	**4671.37661**	0.00%	4671.37660	4671.37661	0.00%
A_03	848.17861	**848.17861**	0.00%	848.17861	**848.17861**	0.00%
A_04	2085.87605	**2085.87605**	0.00%	2085.87605	**2085.87605**	0.00%
A_05	635.22178	635.59898	0.06%	635.22178	635.298076	0.01%
A_06	590.62359	**590.62359**	0.00%	590.62359	**590.62359**	0.00%
A_07	2272.78227	**2272.78227**	0.00%	2272.78227	**2272.78227**	0.00%
A_08	744.29323	**744.29323**	0.00%	744.29323	**744.29323**	0.00%
A_09	1507.28478	**1507.28478**	0.00%	1507.28478	**1507.28478**	0.00%
A_10	2994.84873	**2994.84873**	0.00%	2994.84873	**2994.84873**	0.00%
A_11	495.25577	495.32171	0.01%	495.25577	**495.25577**	0.00%
A_12	789.63492	**789.63492**	0.00%	789.63492	**789.63492**	0.00%
A_13	1998.66216	1999.62679	0.05%	1998.66216	1998.79003	0.01%
A_14	2264.12432	**2264.12432**	0.00%	2264.12432	**2264.12432**	0.00%
A_15	2268.56915	**2268.56915**	0.00%	2268.56915	2269.54047	0.04%

Table 8 Performance of ILS on dataset A instances.

Instance	15 minutes			1.5 hours		
	Best	ILS	%Diff	Best	ILS	%Diff
B_01	3986.20283	**3986.20283**	0.00%	3986.20283	**3986.20283**	0.00%
B_02	4302.77452	**4300.05660**	-0.06%	4301.65660	**4299.00566**	-0.06%
B_03	35279.53018	35284.81226	0.01%	35277.22830	35281.73773	0.01%
B_04	34827.86981	34828.87547	0.00%	34826.94622	34828.84056	0.01%
B_05	2397.09905	**2396.70660**	-0.02%	2397.10094	2397.18301	0.00%
B_06	4287.89434	**4283.36320**	-0.11%	4284.67169	4284.73867	0.00%
B_07	7564.03490	**7555.35566**	-0.11%	7555.95000	**7551.01792**	-0.07%
B_08	7435.71904	**7435.71904**	0.00%	7435.71904	**7435.71904**	0.00%
B_09	7491.75357	7493.61666	0.02%	7491.75357	7496.87857	0.07%
B_10	10637.62000	**10602.87600**	-0.33%	10633.01600	**10611.42199**	-0.20%
B_11	3626.27169	3629.57735	0.09%	3626.03490	3623.07452	-0.08%
B_12	37602.96666	**37600.49166**	-0.01%	37601.38382	37601.55637	0.00%
B_13	5024.49264	5024.96813	0.01%	5024.49264	**5024.49264**	0.00%
B_14	11905.10994	11915.00471	0.08%	11901.76858	11908.87356	0.06%
B_15	22566.00340	**22564.34420**	-0.01%	22563.53880	22563.64279	0.00%

Table 9 Performance of ILS on dataset B instances.

Tables 8 - 9 indicate that the proposed ILS performs very well on dataset A and B which contain mostly small and medium instances. The algorithm obtains better solution than the best known result for some instances in dataset B. Comparing the results of 15 minutes and 1.5 hours, it is noted that the improvement in solution quality is not significant.

Tables 10 - 11 indicate that the proposed ILS is less effective on the large instances of dataset C and X. This is not surprising since increasing the size of the instances will increase the computational burden of model (A-MILP). Hence, achieving an initial solution takes too much time because we rely on CPLEX. Another possible reason is that the subroutine SEARCH in the ILS, which involves exhaustive search on the neighborhood by *one-shift* and *two-swap*, become time consuming for large instances. This significantly reduces the number of perturbation in the ILS procedure due to the imposed time limits. For a few instances, e.g. C_14 and X_02, running the algorithm for 1.5 hours leads to a worse solution than the result obtained in 15 minutes.

Instance	15 minutes			1.5 hours		
	Best	ILS	%Diff	Best	ILS	%Diff
C_01	8515.90377	8517.58301	0.02%	8515.90377	**8515.90377**	0.00%
C_02	3541.65377	3548.55471	0.19%	3539.80377	3541.53301	0.05%
C_03	33511.70000	33517.10094	0.02%	33512.25660	33519.07261	0.02%
C_04	37585.73113	37589.06981	0.01%	37586.30849	37588.03867	0.00%
C_05	3166.88962	3167.43773	0.02%	3166.18207	3167.00000	0.03%
C_06	8396.00094	8418.44245	0.27%	8394.48301	8411.25943	0.20%
C_07	6083.27023	6083.60000	0.01%	6083.04404	6083.60000	0.01%
C_08	11162.83600	11193.16198	0.27%	11155.64000	11191.93800	0.33%
C_09	5586.97924	5598.36037	0.20%	5585.65188	5595.71037	0.18%
C_10	43342.48872	43343.28235	0.00%	43341.83676	43344.15490	0.01%
C_11	5749.95735	**5749.95735**	0.00%	5749.95735	**5749.95735**	0.00%
C_12	12721.13324	12735.89214	0.12%	12718.79057	12730.73507	0.09%
C_13	42487.99282	42489.57130	0.00%	42484.56065	42485.73760	0.00%
C_14	26467.22113	26467.73954	0.00%	26457.11454	26479.44295	0.08%
C_15	39758.02750	39759.60233	0.00%	39757.54750	39759.35333	0.00%

Table 10 Performance of ILS on dataset C instances.

Instance	15 minutes			1.5 hours		
	Best	ILS	%Diff	Best	ILS	%Diff
X_01	4014.37075	4020.20377	0.15%	4011.37641	4018.60849	0.18%
X_02	32231.43867	32237.97075	0.02%	32228.63679	32238.56698	0.03%
X_03	8104.53773	8126.47075	0.27%	8102.58962	8118.05283	0.19%
X_04	11315.94600	11354.26599	0.34%	11303.40000	11335.16399	0.28%
X_05	22858.11477	22872.21181	0.06%	22837.42068	22857.14295	0.09%
X_06	47032.95633	47035.15216	0.00%	47032.16366	47033.91750	0.00%
X_07	13221.61783	13222.48349	0.00%	13221.35849	13222.34350	0.00%
X_08	13717.37033	13726.39583	0.07%	13707.28500	13724.84233	0.13%
X_09	20195.40500	20195.99833	0.00%	20180.45000	20196.45833	0.08%
X_10	17289.31571	17302.40000	0.08%	17267.81857	17269.86714	0.01%
X_11	39121.52404	39121.53206	0.00%	39115.26526	39119.86755	0.01%
X_12	47502.80992	47582.45305	0.17%	47441.36908	47462.10419	0.04%
X_13	15784.25141	15794.64339	0.07%	15784.16933	15788.57924	0.03%
X_14	79417.02750	**79416.08194**	0.00%	79416.86527	**79414.84444**	0.00%
X_15	45491.80749	45493.04388	0.00%	45422.28999	45426.09194	0.00%

Table 11 Performance of ILS on dataset X instances.

8 Conclusion

In this paper, we studied the grid operation-based outage maintenance planning problem which was proposed for the 2020 ROADEF/EURO Challenge. The problem possesses several unique features and is fundamentally different from the typical stochastic programming problems due to the quantile criterion in the objective function. Several mixed integer linear programming (MILP) models were derived and three heuristic algorithms were developed for the considered problem.

The first MILP formulation uses binary variables as indicators to model quantile. The advantage of this model is that we can obtain exact solutions to problem instances with up to 6 scenarios in less than 100 seconds. However, the MILP becomes intractable as the number of scenarios grow very large, such as the test cases considered in this study. This is because the size of the first model is dependent on the total number of scenarios as well as the planning horizon. The second MILP formulation uses general variables to model quantile, owing to the assumption that the quantile of the sum of distributions is proportional by a fixed parameter to the sum of quantile of the individual distribution.

Solving this model provides a good feasible solution which is important for the proposed heuristics and metaheuristic.

We apply the confidence (guaranteeing) approach for solving stochastic program with quantile objective function and discrete distribution of the random parameters. The confidence approach leads to two solution methods: (i) confidence method (CM) which uses the confidence sets to measure quantile; and (ii) critical-scenario confidence method (cs-CM) which uses the critical scenarios of the confidence sets. In our experiments, CM works well on dataset A which contains mostly small-to-medium instances but is less effective as the problem size grows much larger. On the other hand, cs-CM produces poor-quality solution in the first few iterations due to the large differences between the approximate objective function and the true objective function. As a result, cs-CM requires more iterations to improve the initial solution but each iteration needs less time than that of CM.

An iterated local search algorithm was developed for solving the maintenance planning problem. The performance of ILS on the four given datasets was compared with the best known results from the other participants. Based on the computational results, we observed that our ILS performed very well on the instances of datasets A, giving an average relative percentage difference of about 0.004% for both 15 minutes and 1.5 hours tests. The effectiveness of our ILS was more profound for instances in dataset B where we obtained better solution for 50% of instances. As the problem size increases, the growth in computation time for executing the local search procedures is significant. This, in turn, substantially reduces the number of perturbation in the ILS procedure due to the imposed time limit. As a consequence, ILS could not escape the local optimum and find high-quality solutions to the large instances in datasets C and X.

References

[1] Skytte, K., Ropenus, S.: Regulatory Review and International Comparison of EU-15 Member States. Technical report (2005)

[2] Froger, A., Gendreau, M., Mendoza, J.E., Pinson, E., Rousseau, L.: Maintenance scheduling in the electricity industry: A literature review. European Journal of Operational Research **251**, 695–706 (2016)

[3] Mazidi, P., Tohidi, Y., Ramos, A., Sanz-Bobi, M.A.: Profit-maximization generation maintenance scheduling through bi-level programming. European Journal of Operational Research **264**(3), 1045–1057 (2018)

[4] Schlünz, E., van Vuuren, J.: An investigation into the effectiveness of simulated annealing as a solution approach for the generator maintenance scheduling problem. International Journal of Electrical Power & Energy Systems **53**, 166–174 (2013)

[5] Reihani, E., Sarikhani, A., Davodi, M.: Reliability based generator maintenance scheduling using hybrid evolutionary approach. International Journal of Electrical Power & Energy Systems **42**(1), 434–439 (2012)

[6] 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 Transactions on Power Systems **31**(3), 2225–2233 (2016)

[7] 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. Applied Soft Computing **21**, 72–83 (2014)

[8] Fu, Y., Shahidehpour, M., Li, Z.: Security-constrained optimal coordination of generation and transmission maintenance outage scheduling. IEEE Transactions on Power Systems **22**(3), 1302–1313 (2007)

[9] Khalid, A., Ioannis, K.: A survey of generator maintenance scheduling techniques. Global Journal of Researches in Engineering **12**(1) (2012)

[10] Pandžić, H., Conejo, A.J., Kuzle, I., Caro, E.: Yearly maintenance scheduling of transmission lines within a market environment. IEEE Transactions on Power Systems **27**, 407–415 (2012)

[11] Marwali, M., Shahidehpour, S.: Short-term transmission line maintenance scheduling in a deregulated system. IEEE Transactions on Power Systems **15**(3), 1117–1124 (2000)

[12] Lv, C., Wang, J., You, S., Zhang, Z.: Short-term transmission maintenance scheduling based on the Benders decomposition. International Transactions on Electrical Energy Systems **25**, 697–712 (2014)

[13] Alayo, H., Paucar, E.: A MILP Model for Maintenance Scheduling in Transmission Systems and an Example Application to the Peruvian System. IEEE Latin America Transactions **16**(4), 1099–1104 (2018)

[14] Kulkarni, R., Khuntia, S.R., Joseph, A., Rueda, J.L., Palensky, P.: Economic outage scheduling of transmission line for long-term horizon under demand and wind scenarios. (2018)

[15] Dalal, G., Gilboa, E., Mannor, S., Wehenkel, L.: Chance-constrained outage scheduling using a machine learning proxy. IEEE Transactions on Power Systems **34**(4) (2019)

[16] Toubeau, J., Pardoen, L., Hubert, L., Marenne, N., Sprooten, J., De Grève, Z., Vallée, F.: Machine learning-assisted outage planning for

maintenance activities in power systems with renewables. Energy **238**, 121993 (2022)

[17] Jiang, Y., Zhang, Z., Mccalley, J., Van Voorhis, T., Ames, I.A.: Risk-based maintenance optimisation for transmission equipment. IEEE Transactions on Power Systems **21**, (2006)

[18] Abiri-Jahromi, A., Fotuhi-Firuzabad, M., Abbasi, E.: An efficient mixed integer linear formulation for long-term overhead lines maintenance scheduling in power distribution systems. IEEE Transactions on Power Delivery **24**(4), 2043–2053 (2009)

[19] Marin, M., Karangelos, E., Wehenkel, L.: A computational model of mid-term outage scheduling for long-term system studies, pp. 1–7 (2017)

[20] Erschler, J.: Analysis under constraints and decision support for certain scheduling problems. PhD thesis, University of Toulouse (1976)

[21] Kibzun, A.I., Kurbakovskiy, V.Y.: Guaranteeing approach to solving quantile optimisation problems. Annals of Operations Research **30**, 81–94 (1991)

[22] Lourenço, H.R., Martin, O.C., Stützle, T.: Iterated local search: framework and applications. In: Gendreau, M., Potvin, J. (eds.) Handbook of Metaheuristics vol. 2, pp. 363–397. Springer, Boston (2010)

[23] Brandão, J.: A memory-based iterated local search algorithm for the multi-depot open vehicle routing problem. European Journal of Operational Research **284**(2), 559–571 (2020)

[24] Almakhlafi, A., Knowles, J.: Iterated Local Search for the Generator Maintenance Scheduling Problem, pp. 708–742 (2015)

[25] Gu, H., Lam, H.C., Zinder, Y.: Planning rolling stock maintenance: optimisation of train arrival dates at a maintenance centre. Journal of Industrial & Management Optimisation **18**(2), 747–772 (2022). https://doi.org/10.3934/jimo.2020177

[26] Khatami, M., Salehipour, A., Hwang, F.J.: Makespan minimization for the m -machine ordered flow shop scheduling problem. Computers & Operations Research **111**, 400–414 (2019)

[27] Shaw, P.: Using constraint programming and local search methods to solve vehicle routing problems. Lecture Notes in Computer Science, 417–431 (1998)

[28] Lin, S., Kernighan, B.W.: An effective heuristic algorithm for the

travelling-salesman problem. Operations Research **21**(2), 498–516 (1973)

[29] Ruiz, M.: Github Repository for ROADEF/EURO 2020 Challenge. https://github.com/rte-france/challenge-roadef-2020/ [Accessed: 01 September 2020] (2020)

[30] Behnel, S., Bradshaw, R., Citro, C., Dalcin, L., Seljebotn, D.S., Smith, K.: The best of both worlds. Computing in Science & Engineering **13**(2), 31–39 (2011)