A two-stage robust approach for minimizing the weighted number of tardy jobs with objective uncertainty

Henri Lefebvre, François Clautiaux, Boris Detienne

A two-stage robust approach for minimizing the weighted number of tardy jobs with objective uncertainty

H. Lefebvre[1]

Dipartimento di Ingegneria dell'Energia Elettrica e dell'Informazione "Guglielmo Marconi", Università di Bologna

F. Clautiaux[2], B. Detienne[3]

Université de Bordeaux, UMR CNRS 5251, Inria Bordeaux Sud-Ouest

Abstract

Minimizing the weighted number of tardy jobs on one machine is a classical and intensively studied scheduling problem. In this paper, we develop a two-stage robust approach, where exact weights are known after accepting to perform the jobs, and before sequencing them on the machine. This assumption allows diverse recourse decisions to be taken in order to better adapt one's mid-term plan.

The contribution of this paper is twofold: first, we introduce a new scheduling problem and model it as a min-max-min optimization problem with mixed-integer recourse by extending existing models proposed for the deterministic case. Second, we take advantage of the special structure of the problem to propose two solution approaches based on results from the recent robust optimization literature: namely the finite adaptability (Bertsimas and Caramanis, 2010) and a convexification-based approach (Arslan and Detienne, 2022). We also study the additional cost of the solutions if the sequence of jobs has to be decided before the uncertainty is revealed. Computational experiments are reported to analyze the effectiveness of our approaches.

Keywords: One-machine scheduling, robust optimization, two-stage optimization, mixed-integer recourse, exact approach, integer programming

Historically, scheduling optimization problems have been solved in a deterministic fashion assuming that every input data were perfectly known at decision time. These problems have been, and still are, extensively studied in the literature. For an overview of the broad scheduling literature, the reader may refer to [21], which introduced the widely used notation for scheduling problems, and to [30], which covers important theoretical models and significant scheduling problems occurring in the real world.

More recently, researchers have started to focus on scheduling problems where the input data are no longer considered to be known in advance. Rather, input data are often considered to be random variables for which one knows a probability distribution (e.g., stochastic scheduling) or a support of its density function (e.g., robust scheduling). While the first situation yields solutions that are good in average (i.e., leading to an optimal expected objective value), the second approach gives solutions that are never too bad (worst-case objective value). In the static framework, one has to choose the value of all decision variables before the realization of the random variable is revealed, whereas two-stage models ask the decider to fix only a part of the solution before knowing the uncertain parameters, at the *first stage*. At the *second stage*, he is given the opportunity to react after the revelation of the true input data, by determining *recourse* decisions.

A number of approaches have been proposed in the literature to deal with two-stage robust optimization problems. Exact solution approaches mainly focus on problems with continuous recourse decisions and are based on mixed-integer programming large-scale reformulations and dynamic generation of the obtained formulation (see for example [4, 9, 24, 39]). Due to the discrete nature of scheduling problems, the natural

[1]henri.lefebvre@unibo.it
[2]francois.clautiaux@u-bordeaux.fr
[3]boris.detienne@u-bordeaux.fr

mathematical models involve mixed-integer variables in both the first and second stages, which invalidates many solution algorithms. In [3], the authors propose a general framework for solving exactly two-stage mixed-integer robust problems with interdiction binary linking constraints (i.e., constraints of the form $y \leq x$ where both y and x are binary, x denotes the first-stage decision variable and y the recourse decision variable). Most of the other suitable methods are approximate in the sense that they restrict the set of possible recourse decisions. The *decision rules*-based approaches (see *e.g.* [8]) restrain the second-stage variables to simple functions of the random parameters. In [7], the authors present another type of conservative approximation, known as finite adaptability or K-adaptability, which implies limiting the number of recourse actions in the second stage. Those possible recourse policies are determined at the first stage, and second stage only selects the best action to implement in reaction to the uncertain parameters revealed.

Regarding robust scheduling approaches, [2] and [38] present different complexity results for single machine problems. They show that even simple scheduling problems become $\mathcal{NP}$-hard as soon as the uncertainty set contains more than one scenario. In [13], the authors provide approximation algorithms for the problem of minimizing the weighted and unweighted sum of completion times on a single machine where the processing time of the tasks are uncertain. Problems with stochastic breakdowns on one machine (resp. two machines) are studied in [12] (resp. [1]). Affine decision rules are proposed for two-stage robust batch process scheduling under polyhedral uncertainty in [26], based on continuous-time models oriented towards chemistry applications. In [35], a variant of $1||\sum U_j$ with uncertain processing times is studied. Given a discrete scenario-based uncertainty set, one has to determine an initial sequence of jobs that is feasible for nominal processing times. At second stage, once the scenario of actual processing times is revealed, the sequence can be adapted by rejecting some jobs. The objective is to minimize the expected cost of the repaired solution. The authors propose a dynamic programming, a branch-and-bound and a branch-and-price algorithms to solve the problem exactly.

In this manuscript, we introduce a new scheduling problem as an extension of the well known $1|r_j|\sum w_j U_j$ problem where the weighted sum of tardy jobs has to be minimized. In our problem, the jobs are subject to failures, which lead to additional costs. Once the uncertain parameters are revealed (i.e. the weights of the jobs), the decision maker is allowed to take discrete recourse actions: determining the sequence of jobs, outsourcing or spending more time on the jobs to fix them. We address this problem with a robust approach.

This problem has several practical applications. Consider the following example, which arises in the astronomical field when one needs to allocate observatory time. At planning time, a number of sessions have to be reserved for observation purposes, yet, many factors which can alter the quality of the observation are not known, *e.g.*, weather, air quality (see *e.g.* [19, 36]). As a result, it may happen that the sessions lead to lower quality observations, which can be fixed by increasing the time allocated to it, or outsourcing it to another facility. The costs involved in such situations are typically high, therefore, a worst-case-type optimization approach is appropriate.

From an operational point of view, rescheduling jobs might be difficult or costly, and decision-makers may favour recourse solutions where the modifications are easier to handle. In this case, one may seek solutions involving minor modifications to the original plan (see *e.g.*, [6]). The advantages of such solutions are well understood: they reduce the operational costs, reduce the possibility of error in the process, and are generally better accepted by the operators. We therefore propose an alternative version of the scheduling problem where the sequence of jobs cannot be modified after the uncertainty is revealed.

We thus consider two hard scheduling problems with integer recourse, which were never studied before. In general, solving a robust combinatorial problem with integer recourse is a Σ_2^P-hard problem (see e.g. [14]), which induces that even verifying that a first-stage solution is feasible (for any realization of the uncertain parameter) is an $\mathcal{NP}$-hard problem. In deterministic optimization problem, most classical scheduling problems obviously belong to $\mathcal{NP}$, so the main question is generally whether the problem belongs to $\mathcal{P}$ or if is at least as hard as any problem in $\mathcal{NP}$. This is different in robust optimization, where the question of whether a problem belongs to the $\mathcal{NP}$ class is crucial from both theoretical and practical points of view. For example, if the problem does not belong to $\mathcal{NP}$, the problem cannot be modelled as a Mixed-Integer Linear Program (MILP) of polynomial size (unless $\mathcal{NP} = \mathcal{P}$). In this paper, we show that our specific scheduling problems belong to the subclass of robust problems identified by [3], and, thus, are $\mathcal{NP}$-complete.

We show that the two new scheduling problems can be reformulated in such a way that recent works on robust optimization can be instantiated to reformulate this problem exactly [3] or heuristically [22] by deterministic (exponentially large) MILP models. In both cases, our work consists in finding non-trivial

efficient reformulations that lead to models satisfying the technical conditions imposed by the two general frameworks. The computational experiments confirm that the practical difficulty of both problems is considerable: even with state-of-the art methodologies, some instances with 25 jobs remain open after one hour of computing time. We also show a surprising result: forbidding to modify the order of the jobs in the second stage increases the cost of the solution only marginally.

Among the work mentioned above, only [35] proposes exact solving approaches. The problem that we study is different in the following ways. First, we include release dates constraints and different weights for the jobs. Second, we extend the set of possible recourse actions by adding, to the possibility of keeping or rejecting a job at the second stage, the option of repairing it at the expense of an extra processing time. Third, the uncertainty is modeled in [35] by a finite set of discrete scenarios that affects the processing times only, while we consider a polyhedral uncertainty set defining the objective function. Both papers aim at optimizing the worst-case cost, added to average cost in [35]. In terms of methodology, although the two works use branch-and-price algorithms, the reformulations used are totally different. The work in [35] is based on a classical deterministic equivalent formulation, where the recourse decisions for each scenario are modeled using one set of variables and constraints, whereas the current work is based on a robust two-stage programming formulation, which is rewritten as a static robust program of very large size.

In Section 1, we recall some useful results on the deterministic $1|r_j|\sum w_jU_j$, which will be used in the remainder of the paper. In Section 2, we formally describe a first robust version of this problem, before proposing solution methods in Section 3. Section 4 is devoted to the problem version where the order of the jobs cannot be changed at the second stage. We report our computational experiments in Section 5 before concluding.

1. Minimizing the weighted number of tardy jobs: literature review

Minimizing the weighted number of tardy jobs on a single machine, denoted $1|r_j|\sum w_jU_j$ in the literature, is a well known $\mathcal{NP}$-hard scheduling problem (see [21]) and can be stated as follows.

PROBLEM: $1|r_j|\sum w_jU_j$ (DECISION)
INPUT DATA: $(V, \mathcal{J}, (r, d, w, p))$, where V is a positive value, $\mathcal{J}$ a set of jobs, each of which are characterized by the following data: r_j: a release date (i.e., the time before which the job cannot start); d_j: a due date (i.e., the time after which the job is considered tardy; w_j: a weight (i.e., the fixed cost for executing the job tardy) ; p_j: a processing time (i.e., the time needed to execute the job).
QUESTION: Is there a permutation σ of the tasks whose cost (i.e., the weighted number of tardy jobs) is smaller than V?

This problem has been extensively studied in the literature. In particular, [23] proposes a dominance rule for cases with equal release dates known as the Earliest Deadline First rule. Heuristic approaches and lower bounds are given in [15, 16] while exact approaches are given in [5, 27, 31, 32]. To our knowledge, the best exact results are described in [17], where up to 500-job instances are solved in less than one hour.

Since it is of particular interest for our approaches, we formally recall a mixed integer linear programming (MILP) formulation introduced in [17] for solving the $1|r_j|\sum w_jU_j$ problem. The approach is based on two distinct decisions: (1) decide which jobs are to be executed tardy and (2) in what order will the on-time jobs be executed. This is possible since late jobs can be postponed arbitrarily without incurring additional costs. Moreover, we know [25] that if jobs have *agreeable time windows* (i.e., the tasks can be ordered in such a way that $i < j$ implies $r_i \leq r_j$ and $d_i \leq d_j$), then a feasible sequence of on-time jobs exists if and only if the earliest due-date first rule yields a feasible solution. Therefore, an advantage of this approach is that one is able to order the jobs *a priori*, which avoids the need for variables determining the sequence of the jobs. The main idea of [17] is to reformulate $1|r_j|\sum w_jU_j$ into the selection of jobs on a single machine, so that the dominance rule can be exploited, even if the initial instance does not have agreeable time windows. Formally, for any pair of jobs $i \in \mathcal{J}$ and $j \in \mathcal{J}$ such that there are solutions where j is scheduled after i and both jobs are on-time (i.e., $r_i + p_i + p_j \leq d_j$) and that have non-agreeable time windows (i.e., $r_i < r_j$ and $d_i > d_j$), a job occurrence $k \in \widetilde{\mathcal{J}}$ is created, which represents scheduling i before j. It has a hard deadline

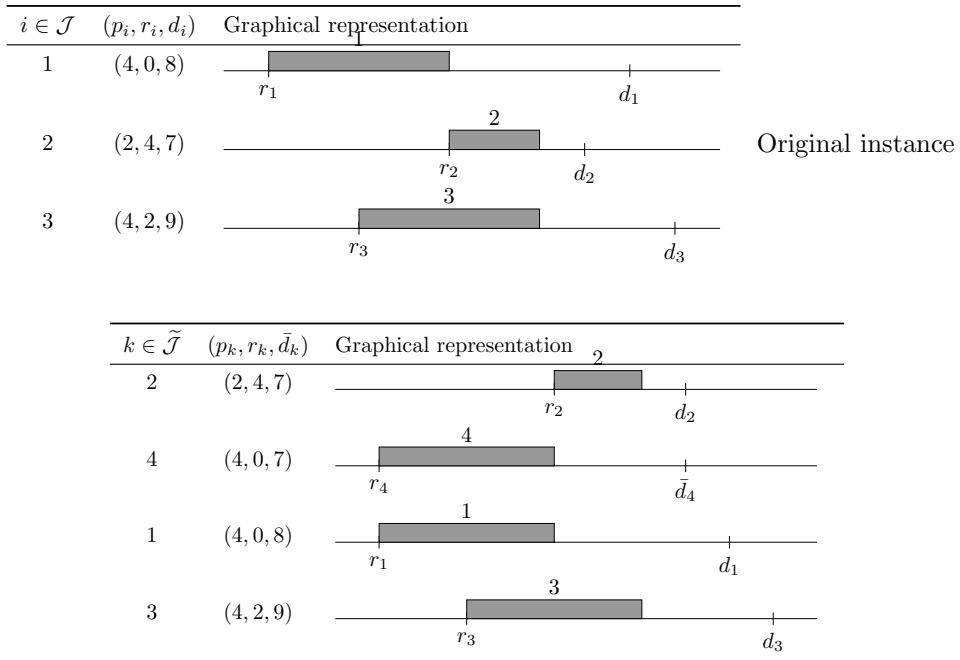


Figure 1: An instance with three jobs, and the job-occurrence representation of the instance. Jobs 1 and 3 already have agreeable time windows, 2 and 3 have non agreeable time windows, but cannot both be scheduled on-time. Since 1 and 2 have non agreeable time windows, a job occurrence numbered 4 representing scheduling 1 before 2 is created. Occurrences are sorted according to Dominance rule 1.

$\bar{d}_k = d_j$, and $r_k = r_i, p_k = p_i, w_k = 0$. The original job i is also added to the set of job occurrences $\widetilde{\mathcal{J}}$, with a null weight, and a deadline $\bar{d}_i = d_i$. The hard deadline is imposed to ensure that if the job occurrence is selected then it is scheduled on time. For every job $j \in \mathcal{J}$, let $\mathcal{G}_j$ be the set gathering all job occurrences related to j. At most one job occurrence in $\mathcal{G}_j$ can be selected in a solution, since all of them correspond with the same original job j. Figure 1 gives a small example of an initial instance defined on jobs, and a modified instance defined on job occurrences.

The following dominance rule extends the earliest deadline first rule to the general $1|r_j|\sum w_j U_j$ problem. It states that when job occurrences are built as described above, the dominance rule applies (although some pairs of time windows may not be agreeable).

Dominance rule 1 ([17]). *There is at least one optimal solution to $1|r_j|\sum w_j U_j$ in which the selected job occurrences of the on-time jobs are ordered according to a non-decreasing order of their deadlines with ties being broken by non-decreasing order of release dates.*

In the remainder, we assume that the job occurrences are ordered according to Dominance Rule 1. Moreover, $\bullet_k$ denotes the data $\bullet$ of the kth job occurrence in that order. For instance, p_k denotes the processing time of the kth job occurrence in that order.

We now recall in detail an MILP model, which is based on the following consideration: assume having fixed the sequencing of the on-time tasks, a straightforward way to check the feasibility of the corresponding schedule is to plan every task as soon as possible. This allows [17] to derive an efficient MILP model for $1|r_j|\sum w_j U_j$, close to the one proposed in [16]. In this model, one has to choose which jobs are late, and for each on-time job j, a job occurrence from G_j has to be selected. Then timing variables are used to ensure that job occurrences are scheduled in their hard time window. For every job $j \in \mathcal{J}$, let U_j be a binary variable equal to 1 if j is tardy, 0 otherwise. Then, for every job occurrence $k \in \mathcal{G}_j$, x_k is the binary variable equal to 1 if k is selected, 0 otherwise, and t_k is a variable equal to its completion time if it is scheduled, or to t_{k-1} if $x_k = 0$. The following MILP models $1|r_j|\sum w_j U_j$.

minimize

$$\sum_{j\in\mathcal{J}} w_j U_j \tag{1}$$

subject to

$$\sum_{k\in\mathcal{G}_j} x_k = 1 - U_j \quad \forall j \in \mathcal{J} \tag{2}$$

$$t_k \leq \bar{d}_k \quad \forall k \in \widetilde{\mathcal{J}} \tag{3}$$

$$t_k - t_{k-1} - p_k x_k \geq 0 \quad \forall k > 1, k \in \widetilde{\mathcal{J}} \tag{4}$$

$$t_k - p_k x_k - M_k x_k \geq r_k - M_k \quad \forall k \in \widetilde{\mathcal{J}} \tag{5}$$

$$U_j \in \{0,1\} \quad \forall j \in \mathcal{J} \tag{6}$$

$$x_k \in \{0,1\} \quad \forall k \in \widetilde{\mathcal{J}} \tag{7}$$

$$t_k \geq 0 \quad \forall k \in \widetilde{\mathcal{J}} \tag{8}$$

Objective function (1) minimizes the weighted number of tardy jobs. Constraints (2) enforce that exactly one job occurrence is selected for an on-time job while no job occurrence should be selected if the job is tardy.

Constraints (3) enforce that no job ends after its deadline, while constraints (4) ensure that jobs do not overlap. Finally, constraints (5) force each scheduled job to begin after its release date. A suitable value for constant M_k is given by $\max(0, \min_{\ell>k}\{r_\ell\})$, which ensures that if a job is not scheduled, then the value of t_k will not be larger than the smallest release date of a job occurrence of larger index.

2. Robust $1|r_j|\sum w_j U_j$

In this section, we introduce a problem where the weighted number of tardy jobs has to be minimized and where the weight associated with the execution of a task is subject to uncertainty. In $1|r_j|\sum w_j U_j$, if a job is not processed on time, it can be arbitrarily delayed. So, deciding that a job will be processed late can be seen as the decision not to accept the order at all. Thus the cost associated with the late job can be seen as a loss of income.

In the two-stage robust version of $1|r_j|\sum w_j U_j$, we assume that the precise weights w_j are known only after deciding which jobs will be on time, and before processing them. Such hypothesis holds for example when orders are accepted at a mid-term level without precise knowledge of the future technical difficulties that may arise from specific jobs, incurring extra production costs.

Our approach is also a conservative approximation of multi-stage decision processes, where the actual production costs would be revealed along time. To the best of our knowledge, uncertain multi-stage integer problems are far out of reach of existing optimization methodologies (see for example [20] or [33] for algorithms dedicated to multi-stage continuous optimization problems), so that such an approximation can still be useful.

2.1. Problem description

We now consider that the executed jobs may fail in such a way that the output of the task is deteriorated, leading to an additional cost. When a deterioration is detected, the decision maker can take recourse decisions of three types: (1) accept the output of the job as it is, thus undertaking a penalty for producing faulty goods; (2) spend more time on the job to perform it correctly and thus avoiding the additional cost or (3) outsource the execution of jobs. Note that it is possible to outsource any job j, even if its weight w_j has not been modified, since it gives additional time for other jobs to be executed or repaired.

We assume that the maximum possible penalty for the faulty production of each job $j \in \mathcal{J}$ is known, and denoted by $\bar{\delta}_j$. The penalty to pay for a given task $j \in \mathcal{J}$ is computed as $\bar{\delta}_j \xi_j$, where ξ_j is the (uncertain) ratio of penalty incurred by j. Following the robust optimization framework, vector $\xi \in \mathbb{R}^{|\mathcal{J}|}$ belongs to the *uncertainty set* $\Xi \subset \mathbb{R}^{|\mathcal{J}|}$. In this paper, we consider the *budgeted uncertainty set*

$$\Xi = \left\{ \xi \in \mathbb{R}_+^{|\mathcal{J}|} \middle| \xi_j \leq 1, \forall j \in \mathcal{J} \text{ and } \sum_{j\in\mathcal{J}} \xi_j \leq \Gamma \right\}$$

where Γ is referred to as the uncertainty budget or uncertainty parameter. This uncertainty set was introduced in [11] and is conservative in the following sense: if Γ increases, the size of Ξ increases. A given vector $\xi \in \Xi$ can be interpreted as a job failure scenario. In any scenario, at most Γ jobs can incur their maximum penalty, but more of them can be partially impacted since vector x_i does not have to take integer values, and the worst-case scenario is generally not an extreme point of Ξ in the two-stage robust optimization context. Remark that when $\Gamma \geq |\mathcal{J}|$, the uncertainty set embeds every possible job failure scenarios.

The decision flow goes as follows: in a here-and-now phase (or first stage), the decision maker decides a set of jobs to be executed on time, then, as the jobs fail, the decision maker is allowed, in a wait-and-see phase (or second stage), to take recourse actions. Note that the optimal solution may be to decide to execute more tasks on time than what is feasible and to finally outsource some of these jobs to restore feasibility.

The problem can now be enunciated formally as follows. We first describe the (second-stage) repairing problem.

PROBLEM: REPAIRING PROBLEM (DECISION)
INPUT DATA: $(V, \mathcal{J}, (r, d, w, p, \bar{\delta}, \tau, f), A, \xi)$, where $V \in \mathbb{R}$ is a target value, $\mathcal{J}$, a set of jobs characterized by data (r, d, w, p), and for each job j, a maximum additional cost $\bar{\delta}_j$ if j fails, a fixed extra time τ_j needed to repair j, a fixed cost f_j for outsourcing j, a set of initially on-time jobs A, and $\bar{\xi}$ is a failure scenario.
QUESTION: Is there a partition (B, C, D) of A and a permutation σ of $B \cup C$, where B is the set of jobs to be scheduled without modification, C is the set of jobs to be fixed and scheduled, and D is the set of jobs to be outsourced, such that all jobs of $B \cup C$ are on-time and $\sum_{j \in \mathcal{J} \setminus A} w_j + \sum_{j \in B} \bar{\delta}_j \xi_j + \sum_{j \in D} f_j \leq V$?

Using this definition, one can enunciate the two-stage problem formally.

PROBLEM: ROBUST $1|r_j|\sum w_j U_j$ (DECISION)
INPUT DATA: $(V, \mathcal{J}, (r, d, w, p, \bar{\delta}, \tau, f), \Xi)$, where $V \in \mathbb{R}$ is a target value, $\mathcal{J}$, a set of jobs characterized by data (r, d, w, p), and for each job j, a maximum additional cost $\bar{\delta}_j$ if j fails, a fixed extra time τ_j needed to fix j, a fixed cost f_j for outsourcing j, and Ξ is an uncertainty set.
QUESTION: Is there a subset $A \subseteq \mathcal{J}$ of on-time jobs such that for any scenario $\xi \in \Xi$, REPAIRING PROBLEM with data $(V, \mathcal{J}, (r, d, w, p, \bar{\delta}, \tau, f), A, \xi)$ has answer yes?

Our scheduling problem consists in seeking the minimum value of V such that the decision problem has answer yes. This optimization problem will be further referred to as problem $(\mathcal{P})$. When $\bar{\delta}_j = 0$ for any job j, the problem becomes the classical $1|r_j|\sum w_j U_j$ and is therefore $\mathcal{NP}$-hard. However, note that for given input data, if one is given a subset A of jobs, determining if the repairing problem has solution yes for all possible scenarios is not straightforward. This means that in terms of theoretical complexity, a first-stage solution A does not provide a direct polynomial certificate, as would be the case in a deterministic setting. We need to introduce complex reformulations before being able to prove that the problem belongs to class NP, and so, is NP-complete.

2.2. Formulation

In this section, we show that (1)-(8) can be extended to model problem $(\mathcal{P})$. To do so, we use an approach similar to [17] and which was recalled in Section 1. Its validity is based on dominance rule 1, we therefore need to ensure that it still holds for the robust case.

We create a set of job occurrences $\widetilde{\mathcal{J}}$ from the original set of jobs $\mathcal{J}$ in order to turn the problem into a job occurrence selection problem with agreeable time windows. Formally, consider a job $i \in \mathcal{J}$, for any job $j \in \mathcal{J}$ whose time window is included in that of i (i.e., $r_i < r_j$, $d_i > d_j$), and such that i and j can both be on-time in a solution (i.e. $r_i + p_i + p_j \leq d_j$), we create a job occurrence $k \in \widetilde{\mathcal{J}}$ such that $r_k = r_i, p_k = p_i, w_k = 0, f_k = f_i, \bar{\delta}_k = \bar{\delta}_i, \tau_k = \tau_i$ and $d_k = d_j$. Again, the original job i is also added to $\widetilde{\mathcal{J}}$ and we introduce the set $\mathcal{G}_j$ as the set of job occurrences related to a given job j.

We can now extend the dominance rule 1 in the following sense:

Dominance rule 2. *There is at least one optimal solution* $((B^*, C^*, D^*), \sigma^*)$ *for REPAIRING PROBLEM such that jobs of* $B^* \cup C^*$ *are scheduled according to a non-decreasing order of their deadlines*

Proof. Let $((B, C, D), \sigma)$ be a feasible solution for REPAIRING PROBLEM, and consider two jobs i and j such that $i, j \in B \cup C$. Assume moreover that i is before j in σ.

If $d_i < d_j$ (or $d_i = d_j$ and $r_i < r_j$), then i and j are already scheduled in the desired order. Otherwise, for any job ℓ, let $\hat{p}_\ell = p_\ell$ if $\ell \in B$ and $\hat{p}_\ell = p_\ell + \tau_\ell$ if $\ell \in C$. Note that since (B, C, D) is feasible, it holds that

$$r_i + \hat{p}_i + \hat{p}_j \leq d_j. \tag{9}$$

Two cases remain:

- $d_i > d_j$ and $r_i < r_j$: (9) implies that $r_i + p_i + p_j \leq d_j$. Therefore, there exists a job occurrence $k \in \mathcal{G}_i$ such that $d_k = d_j$ and with which we can replace i. Doing so, we end up in the desired order and the objective value remains unchanged.
- $d_i \geq d_j$ and $r_i \geq r_j$: swapping i and j leads to the desired order, without modifying the cost of the solution. Since $r_j \leq r_i$ and $d_i \geq d_j$, it holds from (9) that $r_j + \hat{p}_j + \hat{p}_i \leq d_i$, which shows that the solution remains feasible.

□

In the exact same way as what has been done in [17] and recalled in Section 1, we sort the job occurrences according to a non-decreasing order of their deadlines.

We now propose a characterization of valid recourse decisions. Schedule-feasibility of a selection of jobs has been studied in the static case by [17] (see Section 1) and can be extended to our case.

For any $k \in \widetilde{\mathcal{J}}$, binary variable y_k takes value 1 if k is selected, 0 otherwise ; variable z_k takes value 1 if k is repaired, 0 otherwise. For each occurrence $k \in \widetilde{\mathcal{J}}$, variable $\rho_k \in \mathbb{R}_+$ is equal to the processing time of the job (including possible repairing).

We define the set $\bar{\mathcal{Y}} \subset \{0,1\}^{2|\widetilde{\mathcal{J}}|} \times \mathbb{R}_+^{2|\widetilde{\mathcal{J}}|}$, which contains every feasible schedule, as follows.

$$\bar{\mathcal{Y}} = \begin{cases} \rho_k = p_k y_k + \tau_k z_k \quad \forall k \in \widetilde{\mathcal{J}} & (10) \\ z_k \leq y_k \quad \forall k \in \widetilde{\mathcal{J}} & (11) \\ \sum_{k \in \mathcal{G}_j} y_k \leq 1 \quad \forall j \in \mathcal{J} & (12) \\ t_k \leq \bar{d}_k \quad \forall k \in \widetilde{\mathcal{J}} & (13) \\ t_k - t_{k-1} - \rho_k \geq 0 \quad \forall k > 1, k \in \widetilde{\mathcal{J}} & (14) \\ t_k - \rho_k - M_k y_k \geq r_k - M_k \quad \forall k \in \widetilde{\mathcal{J}} & (15) \\ t_k \geq 0 \quad \forall k \in \widetilde{\mathcal{J}} & (16) \\ y_k, z_k \in \{0,1\} \quad \forall k \in \widetilde{\mathcal{J}} & (17) \\ \rho_k \geq 0 \quad \forall k \in \widetilde{\mathcal{J}} & (18) \end{cases}$$

Constraints (10) ensure that each value ρ_k is equal to the processing time of k with respect to the recourse action. Constraints (11) enforce that a job may be fixed only if it is scheduled while constraints (12)-(17) are understood exactly as (1)-(8) where the constant processing times p have been substituted by decision variables ρ. Recall that a job can be outsourced, which explains why constraints (12) are less-or-equal constraints whereas (2) are equality constraints.

Let us also denote the set of second-stage decisions that admit a feasible timing of the jobs by

$$\mathcal{Y} = \{(y, z) | \exists t, \rho : (y, z, t, \rho) \in \bar{\mathcal{Y}}\}.$$

To enforce the non-anticipation property, which stipulates that the decided recourse action may not contradict the first-stage decision, we add so-called linking constraints between the first-stage decisions and the second-stage decisions. These linking constraints are expressed as

$$\sum_{k \in \mathcal{G}_j} y_k \leq 1 - U_j \quad \forall j \in \mathcal{J} \tag{19}$$

We also introduce the set $\mathcal{Y}(U)$ of admissible recourse decisions respecting both the non-anticipation and the schedule-feasibility property. It is given by $\mathcal{Y}(U) = \{(y, z) \in \mathcal{Y} \mid (19)\}$.

We now take interest in the objective value. Let $j \in \mathcal{J}$ be a job to be scheduled, then:

- if $U_j = 1$, then j is executed tardy and we have $\forall k \in \mathcal{G}_j, y_k = z_k = 0$ (i.e., no recourse action)
- if $U_j = 0$, then j is accepted in the first stage. In the second stage (i.e., once the uncertainty is revealed) the sequencing of the jobs has to be decided as well as the recourse actions. The following cases may arise:
  - there is $k \in \mathcal{G}_j$ such that $y_k = z_k = 1$: the job is executed and fixed
  - there is $k \in \mathcal{G}_j$ such that $y_k = 1$ and $z_k = 0$: the job is executed
  - for all $k \in \mathcal{G}_j, y_k = z_k = 0$: the job is outsourced.

The problem can finally be cast as:

$$(\mathcal{P}) : \min_{U \in \{0,1\}^{|\mathcal{J}|}} \sum_{j \in \mathcal{J}} w_j U_j + f_j(1 - U_j) + \max_{\xi \in \Xi} \min_{(y,z) \in \mathcal{Y}(U)} R(\xi, y, z)$$

where $R(\xi, y, z)$ denotes the cost of recourse action (y, z) corresponding to scenario ξ given by:

$$R(\xi, y, z) = \sum_{j \in \mathcal{J}} \sum_{k \in \mathcal{G}_j} \left[(\bar{\delta}_k \xi_j - f_k) y_k - \bar{\delta}_k \xi_j z_k \right].$$

Note that the outsourcing cost appears both in the first-stage and second-stage objective functions. This technical manipulation is required to preserve their linearity, and can be interpreted as always paying outsourcing, unless the job is scheduled in the second stage.

3. Solution approaches

In this section, we develop two solution approaches for solving this min-max-min problem based on two recent studies on robust optimization. First, we present a conservative approximation known as K-adaptability, introduced in [7], which uses restrictive assumptions on the recourse set. Then, an exact approach is developed based on polyhedral results introduced in [3], which are further briefly recalled for the sake of completeness.

3.1. Finite adaptability

One of the most effective approaches for two-stage robust problems is the so-called finite adaptability introduced in [7] also referred to as the K-adaptability. To obtain a tractable problem, the idea is to restrict the set of possible recourse actions. The derivation of an exact MILP model of the finite adaptability approximation for robust problems with recourse where the uncertainty is confined in the objective function has been summarized in [22] and is well established. In our setting, this consists in deciding at the first stage, in addition to the set of jobs accepted (*i.e.* a vector U), K recourse solutions $(y^q, z^q) \in \mathcal{Y}(U)$, $q = 1, \ldots, K$, each of which prescribing which jobs should be executed and repaired if we select this specific recourse solution. The second stage reduces to choosing one of these recourse solutions, once the uncertain parameters ξ are revealed. We get the following model:

$$\text{minimize}$$

$$\sum_{j \in \mathcal{J}} w_j U_j + f_j(1 - U_j) + \max_{\xi \in \Xi} \min_{q=1,\ldots,K} R(\xi, y^q, z^q) \tag{20}$$

$$\text{subject to}$$

$$(y^q, z^q) \in \mathcal{Y}(U) \quad q = 1, \ldots, K \tag{21}$$

$$U \in \{0,1\}^{|\mathcal{J}|} \tag{22}$$

The reformulation process proposed in [22] goes by writing the $max-min$ problem in (20) as a single stage maximization linear program (LP), by making use of an epigraph formulation of the inner finite minimum. Using LP duality, an equivalent minimization LP model is derived for expressing the cost of the second-stage $max-min$ sub-problem. Integrated into the first-stage model, this yields a bilinear model which is further linerarized with help of additional decision variables.

More precisely, let us focus on the inner maximization problem in (20) and employ an epigraph formulation of the finite minimum, we get that
$\max_{\xi\in\Xi} \min_{q=1,\dots,K} R(\xi, y^q, z^q)$ is equivalent to the following problem:

$$\text{maximize } \theta \tag{23}$$

$$\text{subject to } \theta \leq R(\xi, y^q, z^q) \quad q = 1, \dots, K \qquad (\beta_q \geq 0) \tag{24}$$

$$\sum_{j\in\mathcal{J}} \xi_j \leq \Gamma \qquad (u \geq 0) \tag{25}$$

$$\xi_j \leq 1 \quad \forall j \in \mathcal{J} \qquad (v_j \geq 0) \tag{26}$$

$$\xi_j \geq 0 \quad \forall j \in \mathcal{J} \tag{27}$$

$$\theta \in \mathbb{R} \tag{28}$$

where constraints (25)-(27) ensure that $\xi \in \Xi$. Observe that model (23)-(28) is a feasible and bounded linear program. We can then use the strong duality theorem in linear programming to obtain an equivalent dual linear program, where β, u and v are the vectors of dual variables respectively associated with constraints (24), (25) and (26) of conforming dimensions. For the sake of completeness, we note that the fully developed expression of Constraints (24) at rank q is:

$$\theta - \sum_{j\in\mathcal{J}} \left[\sum_{k\in\mathcal{G}_j} \bar{\delta}_k (z_k^q - y_k^q) \right] \xi_j \leq \sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{G}_j} -f_k y_k^q.$$

The dual program reads:

minimize

$$-\sum_{q=1}^{K} \sum_{j\in\mathcal{J}} \sum_{k\in\mathcal{G}_j} f_k y_k^q \beta_q + \Gamma u + \sum_{j\in\mathcal{J}} v_j \tag{29}$$

subject to

$$\sum_{q=1}^{K} \beta_q = 1 \qquad (\theta \in \mathbb{R}) \tag{30}$$

$$\sum_{k\in\mathcal{G}_j} \sum_{q=1}^{K} \bar{\delta}_k \left(z_k^q - y_k^q\right) \beta_q + u + v_j \geq 0 \quad \forall j \in \mathcal{J} \qquad (\xi \geq 0) \tag{31}$$

$$\beta_q \geq 0, q = 1, \dots, K \tag{32}$$

$$v_j \geq 0, \forall j \in \mathcal{J} \tag{33}$$

$$u \geq 0 \tag{34}$$

This equivalent formulation contains a bilinear term in (y, z) and β, which can be linearized using standard techniques and introducing auxiliary variables such that $\psi_k^q = y_k^q \beta_q$ and $\zeta_k^q = z_k^q \beta_q$ for all $q = 1, \dots, K$ and $k \in \widetilde{\mathcal{J}}$. Doing so, we obtain the following MILP finite adaptability formulation:

minimize

$$\sum_{j\in\mathcal{J}}[w_jU_j + f_j(1-U_j) + v_j] + \Gamma u - \sum_{q=1}^{K}\sum_{k\in\widetilde{\mathcal{J}}} f_k\psi_k^q \tag{35}$$

subject to

$$\rho_k^q = p_k y_k^q + \tau_k z_k^q \quad \forall k \in \widetilde{\mathcal{J}}, q = 1,\dots,K \tag{36}$$

$$t_k^q \leq \bar{d}_k \quad \forall k \in \widetilde{\mathcal{J}}, q = 1,\dots,K \tag{37}$$

$$t_k^q - t_{k-1}^q - \rho_k^q \geq 0 \quad \forall k > 1, k \in \widetilde{\mathcal{J}}, q = 1,\dots,K \tag{38}$$

$$t_k^q - \rho_k^q - M_k y_k^q \geq r_k - M_k \quad \forall k \in \widetilde{\mathcal{J}}, q = 1,\dots,K \tag{39}$$

$$z_k^q \leq y_k^q \quad \forall k \in \widetilde{\mathcal{J}}, q = 1,\dots,K \tag{40}$$

$$\sum_{k\in\mathcal{G}_j} y_k^q \leq 1 - U_j \quad \forall j \in \mathcal{J}, q = 1,\dots,K \tag{41}$$

$$\sum_{k\in\mathcal{G}_j}\sum_{q=1}^{K} \bar{\delta}_k\left(\zeta_k^q - \psi_k^q\right) + u + v_j \geq 0 \quad \forall j \in \mathcal{J} \tag{42}$$

$$\psi_k^q \leq y_k^q \quad \forall k \in \widetilde{\mathcal{J}}, q = 1,\dots,K \tag{43}$$

$$\psi_k^q \leq \beta_q \quad \forall k \in \widetilde{\mathcal{J}}, q = 1,\dots,K \tag{44}$$

$$\psi_k^q \geq \beta_q - 1 + y_k^q \quad \forall k \in \widetilde{\mathcal{J}}, q = 1,\dots,K \tag{45}$$

$$\zeta_k^q \leq z_k^q \quad \forall k \in \widetilde{\mathcal{J}}, q = 1,\dots,K \tag{46}$$

$$\zeta_k^q \leq \beta_q \quad \forall k \in \widetilde{\mathcal{J}}, q = 1,\dots,K \tag{47}$$

$$\zeta_k^q \geq \beta_q - 1 + z_k^q \quad \forall k \in \widetilde{\mathcal{J}}, q = 1,\dots,K \tag{48}$$

$$(30) - (34)$$

$$t_k^q \geq 0 \quad \forall k \in \widetilde{\mathcal{J}}, q = 1,\dots,K \tag{49}$$

$$U_j \in \{0,1\} \quad \forall j \in \mathcal{J} \tag{50}$$

$$y_k^q \in \{0,1\} \quad \forall k \in \widetilde{\mathcal{J}}, q = 1,\dots,K \tag{51}$$

$$z_k^q \in \{0,1\} \quad \forall k \in \widetilde{\mathcal{J}}, q = 1,\dots,K \tag{52}$$

$$\psi_k^q \geq 0 \quad \forall k \in \widetilde{\mathcal{J}}, q = 1,\dots,K \tag{53}$$

$$\zeta_k^q \geq 0 \quad \forall k \in \widetilde{\mathcal{J}}, q = 1,\dots,K \tag{54}$$

Here, equation (35) defines the objective function to be minimized. Constraints (36),(37)-(39) correspond to scheduling constraints (10),(13)-(15) for each $q = 1,\dots,K$, derived from [17], enforcing that the final decision must yield a feasible schedule. Constraints (40) make sure that a job can be fixed only if it is scheduled while constraints (41) enforce that one may only process a job which was chosen to be on-time in the first stage. Constraints (42) correspond to the dualized cost corresponding to the uncertain event, obtained from (31) through linearization of the bilinear terms. Finally, constraints (43)-(48) correspond to the linearization of $y_k^q\beta_q$ and $z_k^q\beta_q$.

This problem will be referred to as problem $(\mathcal{P}_K)$ and its model as model KAdapt1 -a. Additionally, [34] has introduced a generic branch-and-bound algorithm in order to solve finite adaptability approaches. Their approach is based on disjunctive programming considerations and scenario generation. We will denote this approach by KAdapt1-b.

3.2. Convexification of the recourse set

In [3], the authors introduce an exact single-stage MILP formulation for two-stage robust problems with mixed recourse decisions and binary variables in linking constraints between the first and second stages. In problem $(\mathcal{P})$, the inner $max - min$ problem involves continuous decision variables for the max part, and mixed-binary decision variables for the min part. Thanks to the special structure of the linking constraints (19), one can use Proposition 1 recalled below and employ the following reformulation process. First, replace

the feasible space of the inner *min* sub-problem with its convex hull **(step 1)**. It is then possible to swap the *max* and *min* operators **(step 2)**. This second step leads to a static robust MILP model. The third step applies the classical LP duality-based reformulation [11] to obtain a single-stage deterministic model **(step 3)**. To write a proper MILP model, the final step expresses $\mathcal{Y}(U)$ in terms of its extreme points **(step 4)**. This step implies an exponential growth of the model, which, at solution time, is taken care of with help of a column generation algorithm.

We first detail the reformulation process applied to problem $(\mathcal{P})$, and then show how the large-scale MILP model obtained can be solved.

3.2.1. Reformulation

To perform **step 1**, observe that the recourse cost function R is affine in (y, z). It follows that minimizing R over $\mathcal{Y}(U)$ and $\text{conv}(\mathcal{Y}(U))$ is equivalent. Problem $(\mathcal{P})$ is then equivalent to:

$$\min_{U\in\{0,1\}^{|\mathcal{J}|}} \sum_{j\in\mathcal{J}} [w_j U_j + f_j(1-U_j)] + \max_{\xi\in\Xi} \min_{(y,z)\in\text{conv}(\mathcal{Y}(U))} R(\xi, y, z)$$

Step 2: since function R is affine in both (y, z) and ξ, it is convex in (y, z) and concave in ξ. Moreover, thanks to step 1, both *max* and inner *min* operators are performed over compact convex sets, so that we can use the well-known minimax theorem [28] to swap them. Grouping both *min* operators yields:

$$\min_{\substack{U\in\{0,1\}^{|\mathcal{J}|}\\(y,z)\in\text{conv}(\mathcal{Y}(U))}} \left[\sum_{j\in\mathcal{J}} [w_j U_j + f_j(1-U_j)] + \max\left\{ R(\xi,y,z) : \begin{array}{ll} \sum_{j\in\mathcal{J}} \xi_j \le \Gamma & (u\ge 0)\\ \xi_j \le 1, \forall j\in\mathcal{J} & (v_j \ge 0)\\ \xi_j \ge 0, \forall j\in\mathcal{J} & \end{array} \right\}\right]$$

Step 3 relies on the fact that the inner maximization problem is a feasible and bounded LP. Using the strong LP duality theorem, one can replace it with its dual problem to get the following formulation, where u and v are dual variables associated with the constraints imposing $\xi \in \Xi$:

$$\begin{aligned}
&\text{minimize}\\
&\sum_{j\in\mathcal{J}} [w_j U_j + f_j(1-U_j) + v_j] + \Gamma u - \sum_{k\in\widetilde{\mathcal{J}}} f_k y_k\\
&\text{subject to}\\
&(y,z) \in \text{conv}(\mathcal{Y}(U)) \qquad (55)\\
&u + v_j \ge \sum_{k\in\mathcal{G}_j} \bar{\delta}_k (y_k - z_k) \quad \forall j\in\mathcal{J}\\
&U_j \in \{0,1\} \quad \forall j\in\mathcal{J}\\
&y_k, z_k \ge 0 \quad \forall k\in\widetilde{\mathcal{J}}\\
&v_j \ge 0 \quad \forall j\in\mathcal{J}\\
&u \ge 0
\end{aligned}$$

This model is linear except for constraint (55). In order to write a linear system for these conditions, **step 4** alleviates a key obstacle: considering a fixed vector $\bar{U}$, it is easy to express the set $\text{conv}(\mathcal{Y}(\bar{U}))$ in terms of the extreme points of $\mathcal{Y}(\bar{U})$ since it is a bounded mixed-integer set. However, the set of extreme points to consider depends on the value of $\bar{U}$. In a general setting, this naturally leads to a disjunctive formulation whose numerical solution seems to be very challenging (the reader may refer to [3] for details about this technical difficulty and approaches to cope with it). Problem $(\mathcal{P})$ enjoys a convenient structure that allows us to use the convex hull of $\mathcal{Y}$ instead of $\mathcal{Y}(U)$, and impose the restrictions over y, z and U independently. That means that a single set of extreme points, independent of U, can be considered in the model. To this end, we use the following key result:

Proposition 1 (Arslan and Detienne [3])**.**
Consider the following two-stage robust mixed-integer linear problem with objective uncertainty:

$$\min_{x\in\mathcal{X}}\left\{c^T x+\max_{\xi\in\Xi}\min_{y\in\tilde{\mathcal{Y}}(x)}\xi^T Q x\right\}$$

where $\mathcal{X}\subset\{0,1\}^N\times\mathbb{R}^M$ *denotes the set of feasible first-stage decision,* Ξ *represents the uncertainty polyhedron and* $\tilde{\mathcal{Y}}(x)$ *denotes the set of eligible second-stage decisions defined as* $\{y\in\mathcal{Y}|y_1\leq x_1\}$ *with* $\mathcal{Y}\subset\{0,1\}^N\times\mathbb{R}^M$ *and* $y_1\in\{0,1\}^N$, $x_1\in\{0,1\}^N$. *It holds* $\text{conv}(\tilde{\mathcal{Y}}(x))=\text{conv}(\mathcal{Y})\cap\{y|y_1\leq x_1\}$ *for any* $x\in\mathcal{X}$.

From which we easily derive the following corollary:

Corollary 1.

$$\text{conv}(\mathcal{Y}(U))=\text{conv}(\mathcal{Y})\cap\left\{\begin{array}{l} y\in\mathbb{R}^{|\tilde{\mathcal{J}}|}\\ z\in\mathbb{R}^{|\tilde{\mathcal{J}}|}t\in\mathbb{R}_+^{|\tilde{\mathcal{J}}|}\end{array}\middle|\ \sum_{k\in\mathcal{G}_j}y_k\leq 1-U_j,\forall j\in\mathcal{J}\right\}$$

Let us denote the set of extreme points of $\mathcal{Y}$ by $(\mathbf{y}^e,\mathbf{z}^e), e\in E$ (E being a list for their indices). Problem $(\mathcal{P})$ is finally modeled by this deterministic equivalent program:

$$[DEP]:\ \text{minimize } F(U,u,v,\alpha)$$

$$=\sum_{j\in\mathcal{J}}[w_jU_j+f_j(1-U_j)+v_j]+\Gamma u-\sum_{k\in\tilde{\mathcal{J}}}\left[f_k\sum_{e\in E^R}\mathbf{y}_k^e\alpha_e\right] \tag{56}$$

subject to

$$\sum_{e\in E}\alpha_e=1 \tag{57}$$

$$\sum_{k\in\mathcal{G}_j}\sum_{e\in E}\mathbf{y}_k^e\alpha_e\leq 1-U_j\quad\forall j\in\mathcal{J} \tag{58}$$

$$u+v_j\geq\sum_{k\in\mathcal{G}_j}\left[\bar{\delta}_k\sum_{e\in E}(\mathbf{y}_k^e-\mathbf{z}_k^e)\,\alpha_e\right]\quad\forall j\in\mathcal{J} \tag{59}$$

$$U_j\in\{0,1\}\quad\forall j\in\mathcal{J}$$
$$\alpha_e\geq 0\quad\forall e\in E$$
$$u\geq 0$$
$$v_j\geq 0\quad\forall j\in\mathcal{J}$$

Here, decision vector α represents the convex combination multipliers from the reformulation of $\text{conv}(\mathcal{Y})$. Again, u and v are the dual variables associated to the constraint $\xi\in\Xi$. Constraints (58) link the recourse action with the first-stage decision. Constraint (57) enforces that the recourse actions are convex combinations of the extreme points of $\text{conv}(\mathcal{Y})$. Finally, constraints (59) embed the dualized cost associated to the worst case scenario. This model will be referred to as ColGen1.

We remark that such a characterization of the convex hull as its Minkowski-Weyl formulation is akin to the Dantzig-Wolfe decomposition. Unlike the typical application of Dantzig-Wolfe decomposition, there is no integrality requirements over the reformulated variables (here, the second-stage variables y and z). This stems from the reformulation process that we use. Although $(\mathcal{P})$ involves integer variables in the second-stage, **step 1** allows considering $\text{conv}(\mathcal{Y}(U))$ instead of $\mathcal{Y}(U)$ while keeping an equivalent problem, hence dropping the integrality requirements on variables y and z. It follows that the Dantzig-Wolfe reformulation is applied to a subsystem that involves only continuous variables. Note that, in optimal solutions of $[DEP]$, second-stage variables are most of the time non-integer. Intuitively, several different second-stage solutions are required to prevent the adversary that maximizes the cost of the solution from increasing the value of the first-stage solution by moving slightly the uncertain parameters.

Problem $(\mathcal{P})$ is trivially $\mathcal{NP}$-hard, since considering null penalties yields a problem equivalent to the deterministic $1|r_i|\sum w_iU_i$. However, it is often not clear whether two-stage robust problems lie higher in the

polynomial hierarchy or not (see [10] for example). As a by-product, this reformulation shows that problem $(\mathcal{P})$ is not harder than $\mathcal{NP}$-complete problems (the result is proven in a more general setting in [3]).

Corollary 2. *Problem $(\mathcal{P})$ is $\mathcal{NP}$-complete.*

3.2.2. Column generation-based solution algorithm

Model $[DEP]$ has an exponential number of variables. A classical approach to solve such problems is to use the column generation algorithm to compute its linear relaxation. In this section, we formally present the master program and the pricing problem that has to be solved in this purpose. We then describe the column generation procedure. Finally, we depict the so-called branch-and-price algorithm, which is a tree search embedding the column generation routine to compute the optimal feasible solution of $[DEP]$.

The column generation procedure solves the linear relaxation of model $[DEP]$. Its basic idea is to consider only a subset E^R of the α-variables and solve optimally the so-called restricted master program (RMP) $[DEP]^R$, using for example the simplex algorithm. The linear relaxation of RMP can be stated as follows (constraints $U_j \leq 1$ are dropped since they are implied by constraints (62)).

$$[DEP]^R: \text{ minimize } F^R(U,u,v,\alpha)$$

$$= \sum_{j\in\mathcal{J}}[w_jU_j + f_j(1-U_j) + v_j] + \Gamma u - \sum_{k\in\bar{\mathcal{J}}}\left[f_k \sum_{e\in E^R} \mathbf{y}_k^e \alpha_e\right] \tag{60}$$

subject to

$$\sum_{e\in E^R} \alpha_e = 1 \tag{61}$$

$$\sum_{e\in E^R} \mathbf{y}_k^e \alpha_e + U_j \leq 1 \quad \forall k \in \mathcal{G}_j, \forall j \in \mathcal{J} \tag{62}$$

$$u + v_j \geq \sum_{k\in\mathcal{G}_j}\left[\bar{\delta}_k \sum_{e\in E^R} (\mathbf{y}_k^e - \mathbf{z}_k^e)\,\alpha_e\right] \quad \forall j \in \mathcal{J} \tag{63}$$

$$\alpha_e \geq 0 \quad \forall e \in E^R \tag{64}$$

$$U_j \geq 0, v_j \geq 0 \quad \forall j \in \mathcal{J} \tag{65}$$

$$u \geq 0 \tag{66}$$

Basic LP theory tells us that the solution obtained is optimal for the linear relaxation of $[DEP]$ if the reduced costs of all the α-variables are non-negative. Let λ, μ and π be the dual variables respectively associated with constraints (61), (62) and (63). Given an optimal dual solution $(\lambda^*, \mu^*, \pi^*)$ to $[DEP]^R$, the so-called pricing problem that seeks a minimum reduced cost α-variable can be cast as:

$$[Pricing(\lambda^*,\mu^*,\pi^*)]: \text{ minimize } G(\lambda^*,\mu^*,\pi^*,y,z)$$

$$= -\lambda^* + \sum_{j\in\mathcal{J}}\sum_{k\in\mathcal{G}_j}\left[(-f_k - \mu_k^* + \bar{\delta}_k\pi_j^*)y_k - \bar{\delta}_k\pi_j^* z_k\right]$$

$$\text{subject to } (y,z,t,\rho) \in \bar{\mathcal{Y}}$$

This problem can be interpreted as a variant of $1|r_i|\sum w_iU_i$ where each job comes in two possible modes (related with variables y or z), having different processing times and weights.

When the optimal solution $(U^*, u^*, v^*, \alpha^*)$ of the linear relaxation of $[DEP]$ satisfies the integrality requirements (*i.e.* $U^* \in \{0,1\}^{|\mathcal{J}|}$), then it provides an optimal first-stage solution for $(\mathcal{P})$. Otherwise, one has to branch in order to exclude the current fractional solution and explore the feasibility set. Algorithm 1 summarizes the branch-and-price procedure proposed to solve problem $(\mathcal{P})$ through its formulation $[DEP]$. Line 1 initializes the set of columns so that the restricted master problem is feasible. The best primal bound found, $PrimalBound$ and the best feasible solution found, S^* are initialized in Line 2. Each node is encoded as the set of branching constraints, $\mathcal{B}$, defining the set of solutions of that node. The list of open nodes, $\mathcal{Q}$, is thus initialized in Line 2 with the root node, that has no branching constraints. Loop 3-14 processes the open nodes. The solution of the relaxation at the current node is computed in Line 5. If the solution

Algorithm 1: Branch-and-price algorithm for solving model $[DEP]$.

1 *Initialize the set of columns so that* $[DEP]^R$ *is feasible:* $(\bar{\mathbf{y}}^1, \bar{\mathbf{z}}^1) \leftarrow \{(\mathbf{0}, \mathbf{0})\}$, $E^R \leftarrow \{1\}$
2 $PrimalBound \leftarrow \infty$, $S^* \leftarrow \emptyset$, $\mathcal{Q} \leftarrow \{\emptyset\}$
3 **while** $\mathcal{Q} \neq \emptyset$ **do**
4 Pop a node/set of branching constraints $\mathcal{B}$ from $\mathcal{Q}$
5 $(U^*, u^*, v^*, \alpha^*) \leftarrow optimizeRelaxation(\mathcal{B}, E^R)$
6 $DualBound \leftarrow F(U^*, u^*, v^*, \alpha^*)$
7 **if** $DualBound \geq PrimalBound$ **then**
8 *Current node is pruned by bound*
9 **else**
10 **if** $U^* \in \{0,1\}^{|J|}$ **then**
11 Update $PrimalBound$ and S^* with $DualBound$ and $(U^*, u^*, v^*, \alpha^*)$
12 **else**
13 Choose $i \in \{1, \ldots, |J|\}$ such that $U_i^* \in]0,1[$
14 Add two nodes $\mathcal{B}_0 = \mathcal{B} \cup \{U_i = 0\}$ and $\mathcal{B}_1 = \mathcal{B} \cup \{U_i = 1\}$ to $\mathcal{Q}$
15 **return** S^*, an optimal solution of $[DEP]$

satisfies the integrality requirements (Line 10), $PrimalBound$ and S^* are updated (line 11). When U^* is not integer, branching is performed in Lines 13 and 14.

Algorithm 2 depicts the column generation procedure used to compute the relaxation at each node of the search tree in Line 5 of Algorithm 1. Loop 1-8 adds new columns to the restricted master $[DEP]^R$ until no negative reduced cost column is found. Model $[DEP]^R$ is solved in Line 2, providing optimal dual variables that are used as input to the pricing problem in Line 4. Lines 6-7 add a new column to $[DEP]^R$ if the pricing problem returns a column with a negative reduced cost.

Algorithm 2: $optimizeRelaxation(\mathcal{B}, E^R)$: column generation algorithm for computing the dual bound at each node of the search tree when solving $[DEP]$.

Input: $\mathcal{B}$: set of branching constraints, E^R: set of indices of columns
1 **repeat**
2 Solve $[DEP]^R$ with additional branching constraints $\mathcal{B}$
3 Let $(U^*, u^*, v^*, \alpha^*)$ be the optimal solution and λ^*, μ^* and π^* be the optimal dual values associated with constraints (61), (62) and (63)
4 Solve $[Pricing(\lambda^*, \mu^*, \pi^*)]$
5 Let (y^*, z^*, t^*, ρ^*) be the optimal solution
6 **if** $G(\lambda^*, \mu^*, \pi^*, y, z) < 0$ **then**
7 $E^R \leftarrow E^R \cup \{|E^R| + 1\}$, $(\bar{\mathbf{y}}, \bar{\mathbf{z}})^{|E^R|} \leftarrow (y^*, z^*)$
8 **until** $G(\lambda^*, \mu^*, \pi^*, y, z) \geq 0$
9 **return** $(U^*, u^*, v^*, \alpha^*)$

4. Order-fixing first stage

In this section, we study a variant of problem $(\mathcal{P})$, denoted $(\widetilde{\mathcal{P}})$, where the first-stage decisions include not only the selection of jobs to process, but also their order. In a wait-and-see phase, the recourse actions to be decided for each accepted job are: process the job (possibly with a decreased cost), outsource the job, or repair the job. In problem $(\mathcal{P})$, one can decide the actual processing order of the jobs after knowing their true weight, while in $(\widetilde{\mathcal{P}})$ the sequence of accepted jobs is decided at first stage, and can only be amended by removing some elements of the sequence in the second stage.

We first formalize this variant, characterize its relation with problem $(\mathcal{P})$ and derive MILP formulations. The same two solution approaches, namely finite adaptability and convexification can be applied. Since their application to this variant uses the same mathematical results as the first problem, we report them in Appendix A.

4.1. Formulation

We formally state problem $(\widetilde{\mathcal{P}})$. Similarly to problem $(\mathcal{P})$, we first define the recourse problem.

PROBLEM: FIXED-ORDER-REPAIRING PROBLEM (DECISION)
INPUT DATA: $(V, \mathcal{J}, (r, d, w, p, \bar{\delta}, \tau, f), A, \xi, \sigma)$, where $V \in \mathbb{R}$ is a target value, $\mathcal{J}$, a set of jobs characterized by data (r, d, w, p), and for each job j, a maximum additional cost $\bar{\delta}_j$ if j fails, a fixed extra time τ_j needed to repair j, a fixed cost f_j for outsourcing j, a set of initially on-time jobs A, $\bar{\xi}$ a failure scenario, and σ a permutation of the elements of A.
QUESTION: Is there a partition (B, C, D) of A, where B is the set of jobs to be scheduled without modification, C is the set of jobs to be fixed and scheduled, and D is the set of jobs to be outsourced, σ restricted to $B \cup C$ is feasible, and
$\sum_{j \in \mathcal{J} \setminus A} w_j + \sum_{j \in B} \bar{\delta}_j \xi_j + \sum_{j \in D} f_j \leq V$?

The order-fixing version of the problem can now be defined formally.

PROBLEM: ORDER-FIXING ROBUST $1|r_j|\sum w_j U_j$ (DECISION)
INPUT DATA: $(V, \mathcal{J}, (r, d, w, p, \bar{\delta}, \tau, f), \Xi)$, where $V \in \mathbb{R}$ is a target value, $\mathcal{J}$, a set of jobs characterized by data (r, d, w, p), and for each job j, a maximum additional cost $\bar{\delta}_j$ if j fails, a fixed extra time τ_j needed to fix j, a fixed cost f_j for outsourcing j, and Ξ is an uncertainty set.
QUESTION:
Is there a subset $A \subseteq \mathcal{J}$ of on-time jobs, and a permutation σ of the elements of A such that for any scenario $\xi \in \Xi$, FIXED-ORDER-REPAIRING PROBLEM with data $(V, \mathcal{J}, (r, d, w, p, \bar{\delta}, \tau, f), A, \xi, \sigma)$ has answer yes?

Problem $(\widetilde{\mathcal{P}})$ can be formulated in a similar way as what has been done for problem $(\mathcal{P})$. Just like in Section 1, let us denote, for any job occurrence $k \in \widetilde{\mathcal{J}}$, by x_k the selection of the kth job occurrence in the non-decreasing order of their deadlines (i.e., 1 if the kth job occurrence is used, 0 otherwise). Variables y_k, z_k for job occurrences and U_j will keep the same meaning as in the previous section.

Again, regarding the set of admissible recourses, the schedule-feasibility property is dealt with by set $\mathcal{Y}$ introduced in Section 2. Concerning the non-anticipativity property, which stipulates that the recourse action should not contradict a first-stage decision, we impose that $y_k \leq x_k, \forall k \in \widetilde{\mathcal{J}}$. That is, that one may confirm the execution of a job, or fix a job, only if it were actually accepted in the first stage. The set of admissible recourses is then given by the following set: $\widetilde{\mathcal{Y}}(x) = \{(y, z) \in \mathcal{Y} \mid y_k \leq x_k \quad \forall k \in \widetilde{\mathcal{J}}\}$.

We now detail the different possible decisions. Let $j \in \mathcal{J}$ be a job:

- if $U_j = 1$, then the job is executed tardy and no recourse action may be taken (i.e., $y_k = z_k = 0$)
- if there is k such that $k \in \mathcal{G}_j$ and $x_k = 1$, the job is to be scheduled on time in the first stage:
  - if $y_k = z_k = 1$, job j is executed and fixed on time
  - if $y_k = 1$ and $z_k = 0$, job j is executed on time and a penalty is paid
  - if $y_k = z_k = 0$, job j is outsourced

The objective function can therefore be expressed as follow:

$$\min_{(x,U) \in \widetilde{\mathcal{X}}} \sum_{j \in \mathcal{J}} [w_j U_j + f_j (1 - U_j)] + \max_{\xi \in \Xi} \min_{(y,z) \in \widetilde{\mathcal{Y}}(x)} R(\xi, y, z)$$

where $\widetilde{\mathcal{X}}$ denotes the set of feasible first-stage solutions (i.e., the set of solutions which define a sequence of tasks), that is:

$$\widetilde{\mathcal{X}} = \left\{ (x, U) \in \{0,1\}^{|\widetilde{\mathcal{J}}|} \times \{0,1\}^{|\mathcal{J}|} \middle| \sum_{k \in \mathcal{G}_j} x_k + U_j = 1 \quad \forall j \in \mathcal{J} \right\}$$

Again, note that the first-stage decision does not have to be physically feasible in the sense that the optimal solution may be to decide a sequence of jobs in the first stage and to outsource some of them in the second stage so as to make the schedule feasible.

4.2. Relation with problem $(\mathcal{P})$

This section is devoted to show that the first problem yields a lower bound for the second problem, which is an intuitive result: compared to $(\widetilde{\mathcal{P}})$, in $(\mathcal{P})$ some decisions are postponed. That means that, for different realizations of the uncertainty those decisions can be different in $(\mathcal{P})$ but must be identical in $(\widetilde{\mathcal{P}})$. In that sense, $(\mathcal{P})$ relaxes some of the non-anticipativity constraints of $(\widetilde{\mathcal{P}})$.

Observation 1. *Denoting by* $(\bullet)^*$ *the optimal value of problem* $\bullet$*, the following relation holds:*

$$(\mathcal{P})^* \leq (\widetilde{\mathcal{P}})^*$$

We now provide an example showing that $(\widetilde{\mathcal{P}})$ is a strict relaxation of $(\widetilde{\mathcal{P}})$.

Observation 2. *Given one problem instance, optimal solutions to* $(\mathcal{P})$ *may attain a strictly lower objective value than the optimal solutions to* $(\widetilde{\mathcal{P}})$*.*

Proof. Consider the following instance:

j	r_j	d_j	p_j	τ_j	w_j	$\bar{\delta}_j$	f_j
i	0	6	1	4	100	6	∞
j	5	8	2	2	100	4	∞
k	1	9	2	3	100	5	∞

, $\Gamma = 1$

where the outsourcing of a task is never considered (not affordable) for simplicity. The three jobs can be scheduled on time and it is never optimal to execute a job tardy, even after knowing the penalty (i.e., it is always better to pay the penalty than to execute the job tardy). That being said, it is clear that the optimal sequence is either (i, k, j) or (i, j, k). Since the uncertainty parameter Γ is set to one, exactly one job will be affected by the uncertainty (note that, because we are in a two-stage robust context, the uncertainty can be spread among different random parameters in the worst case, but this does not happen for this instance).

Let us first consider problem $(\widetilde{\mathcal{P}})$ where one decides the sequencing of the jobs before knowing the uncertainty. Figure 2 depicts the two solutions detailed below. If the decision, in the first stage, implies using sequence i, k, j, then it is only possible to fix k. Thus, the cost of the worst case is given by $\max(\bar{\delta}_i, \bar{\delta}_j, 0) = \max(6, 4, 0) = 6$. If one were to choose sequence i, j, k however, the only fixable task is i which means that the worst-case scenario costs $\max(0, \bar{\delta}_j, \bar{\delta}_k) = \max(0, 4, 5) = 5$. The optimal solution to the overall problem minimizes the worst-case cost, hence $(\widetilde{\mathcal{P}})^* = 5$.

If we consider $(\mathcal{P})$ where one only selects on-time jobs in the first stage however, one can better react to the uncertainty in the second stage. Indeed, if the uncertainty affects job j one is forced to pay $\bar{\delta}_j = 4$ since it is never possible to fix it. However, if the uncertainty affects job i, one can react to that scenario by choosing the sequence i, j, k under which job j can be fixed. If, to the contrary, the uncertainty affects job k, the optimal recourse decision is realised by using the sequence i, k, j under which one can fix job k. This shows easily that, for this problem, the worst case is realised when the uncertainty hits job j. In any way, $(\widetilde{\mathcal{P}})^* > (\mathcal{P})^*$ $(5 > 4)$.

□

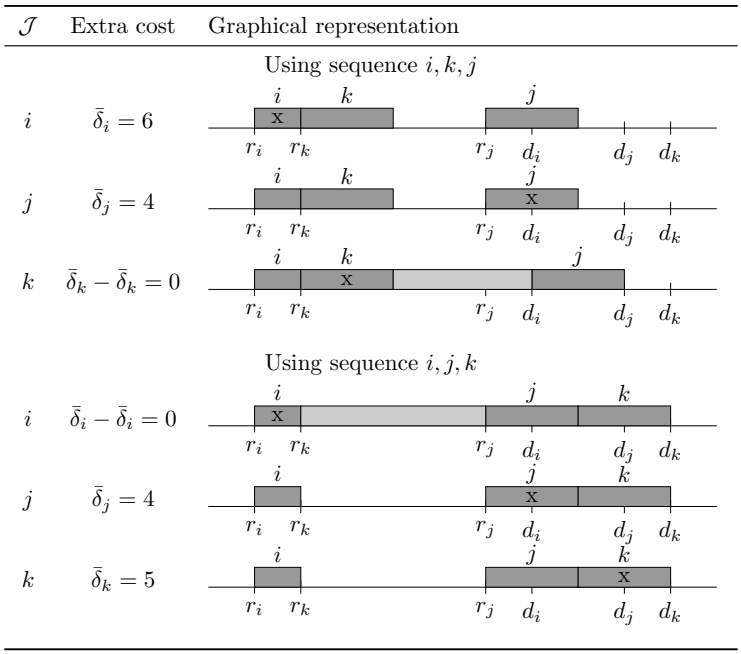


Figure 2: Two schedules and the associated extra cost under the failure of each job if the sequence of jobs is fixed before the uncertainty is revealed

From left to right : the failing job, the associated extra cost, a graphical representation of the schedule

5. Computational experiments

This section reports the main computational results for the two problems we are addressing. We first give some details about our implementation, and then explain how random instances were generated. We also describe our protocol to compare the exact approach with the finite adaptability methods, which are exact only if the input parameter K is large enough.

5.1. Implementation details and experimental setting

All mixed integer linear programs, as well as linear programs inside the column generation procedures, are solved using IBM ILOG Cplex 12.9, through the C callable library, using default parameters and four threads. The generic implementation BapCod [37] of the branch-and-price Algorithm 1 is used to optimize models ColGen1 and ColGen2. At each node of the search tree, the linear relaxation of the problem is computed using column generation (Algorithm 2). The pricing sub-problem is solved using the MILP solver. At most one column is added to the master program $[DEP]^R$ at each iteration. To improve the convergence of the column generation procedure, we use stabilization by automatic smoothing of the dual variables of the master program, as described in [29]. When the optimal solution of the corresponding relaxation does not satisfy the integrality requirements of first-stage variables, one fractional variable is chosen and two child nodes are created in order to exclude its current value from the search space. This variable is chosen to be the closest from 0.5. The open nodes are processed according to the best first rule. The implementation of the branch-and-price algorithm is sequential.

The approach from [34] (i.e., KAdapt1-b and KAdapt2-b) for solving the finite-adaptability counterpart has been implemented in C++ using the author's code which is publicly available[4].

[4]https://github.com/AnirudhSubramanyam/KAdaptabilitySolver

All our experiments are conducted using a 2 Dodeca-core Haswell Intel Xeon E5-2680 v3 2.5 GHz machine with 128Go RAM running Linux OS, part of the PlaFRIM[5] experimental platform. The resources of this machine are strictly partitioned using Slurm Workload Manager[6] to run several tests in parallel. The resources available for each run (algorithm-instance) are set to 4 threads and a 20 Go RAM limit (we remark that our branch-and-price algorithm does not benefit from parallel processing). This virtually creates six independent machines, each running one single instance at a time.

5.2. Instances

The test bed was randomly generated based on the technique used in [15] in which the authors generate a random test bed for the deterministic $1|r_j|\sum w_j U_j$ problem. Their approach takes as input three parameters: the number of jobs N, a factor for the dispersion of the release dates R_1 and a factor controlling the dispersion of the deadlines R_2. Having fixed these parameters, we generate, for each of the N jobs, random characteristics defined as follows[7]:

$$\begin{aligned} p_j &\sim \mathcal{U}(1,100) & w_j &\sim \mathcal{U}(1,100) \\ \bar{\delta}_j &\sim \mathcal{U}(1,100) & f_j &\sim \mathcal{U}(1,100) \\ r_j &\sim \mathcal{U}(0, N \times R_1) & \Delta_j &\sim \mathcal{U}(0, N \times R_2) \\ d_j &= r_j + p_j + \Delta_j & \tau_j &\sim \mathcal{U}(0, \lambda \Delta_j) \end{aligned}$$

Here, Δ_j denotes the slack time of jobs j within its time window. Note that the extra time needed to fix one job τ_j is generated depending on an extra parameter λ, fixed, for our experiments, to $\frac{5}{4}$. This implies that it is *generally* feasible to fix a job if it had its whole time window to be scheduled (i.e., if no other job interferes with it). The parameters which were used are combinations of $N \in \{5, 10, 15, 20, 25\}$, $R_1 \in \{5, 10, 20, 30\}$ and $R_2 \in \{5, 10, 20, 30\}$. In total, 480 instances were generated. Yet, each of these instances are parameterized by the uncertainty budget Γ. Throughout our experiments, the value for Γ varied, from 1 to 3 for 5-job instances, from 1 to 7 for 10-job instances, from 1 to 10 for 15, 20 and 25-job instances. Therefore, we compared the two approaches over 3200 instances.

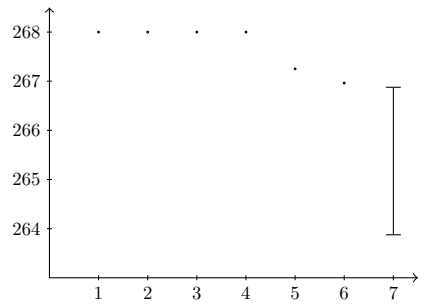
5.3. Protocol for comparing the two solution methods

The finite adaptability method solves the problem exactly only if parameter K is large enough; otherwise it solves an approximation which is tighter and tighter as its parameter K grows. For this reason, we must be careful when comparing its numerical performance with the one of the convexification approach. For a given problem $(\mathcal{P})$, let $(\mathcal{P}_K)$ be its approximation as a K-adaptability problem. We denote by $(\bullet)^*$ the optimal solution of problem $\bullet$ (or the best known upper bound in case of reached time limit) and by $t(\bullet)$ the computation time to solve problem $\bullet$. An attractive case to compare the finite adaptability and the exact approach is when $(\mathcal{P})^* = (\mathcal{P}_K)^*$, since it amounts to comparing two exact methods. However, large values of K lead to intractable models, so we need to find the smallest value for K which fulfills this condition. More formally, we are interested in the following problem : $K^* = \min\{K : (\mathcal{P}_K)^* = (\mathcal{P})^*, K \in \mathbb{N}^*\}$. This problem is feasible and has an upper bound equal to $\dim \Xi + 1 = |\mathcal{J}| + 1$ ([22]). Note that the a priori knowledge of the optimal value $(\mathcal{P})^*$ is assumed. If this assumption is not satisfied, we do not have a practical method to find K^*, since a local minimum of function $K \mapsto (\mathcal{P}_K)^*$ sometimes fails to be global. This is illustrated in Figure 3, which reports the optimal objective value for a specific 15-job instance of $(\widetilde{\mathcal{P}})$ for various values of K. We can see that from the 1-adaptability to the 4-adaptability, the optimal value does not change while it does for greater values of K. This shows that guessing the value K^*, for which the approximation is, in fact, an exact solution, is hard and to our knowledge there is no straightforward stopping criterion for the search for K^*. As a matter of fact, for that specific instance, we are unable to conclude if $K^* = 7$ or if $K^* > 7$ since we were unable to solve it within one hour. That particular observation holds for the two problems $(\mathcal{P})$ and $(\widetilde{\mathcal{P}})$.

[5] PlaFRIM: Plateforme Fédérative pour la Recherche en Informatique et Mathématiques (`https://www.plafrim.fr/fr/accueil/`)

[6] `https://slurm.schedmd.com/` (accessed June 2020)

[7] $\mathcal{U}$ denotes the discrete uniform distribution law



K	Objective	Gap
1	269.000	
2	269.000	
3	269.000	
4	269.000	
5	268.252	
6	267.963	
7	267.888	1.129

Figure 3: A K-adaptability plateau for a 15-jobs instances : the optimal objective value does not change between the 1-adaptability and the 4-adaptability, we were unable to solve the instance using the 7-adaptability within one hour (the vertical bar represents the optimality gap)

We run our algorithms under a given time limit and not all instances are solved to optimality, thus $(\mathcal{P})^*$, $(\mathcal{P}_K)^*$ and their fixed-order counterparts are sometimes approximated. To overcome this difficulty, we focus on a slightly different problem: find the smallest value for K which, under a given time limit T, yields an objective function at least as good as the solution of the exact approach. Formally, we estimate K^* by

$$\widehat{K}^* = \min \left\{ K : \begin{array}{l} (\mathcal{P}_K)^* \leq (\mathcal{P})^*, \\ t(\mathcal{P}) \leq T, \\ t(\mathcal{P}_K) \leq T, \\ K \in \mathbb{N}^* \end{array} \right\}$$

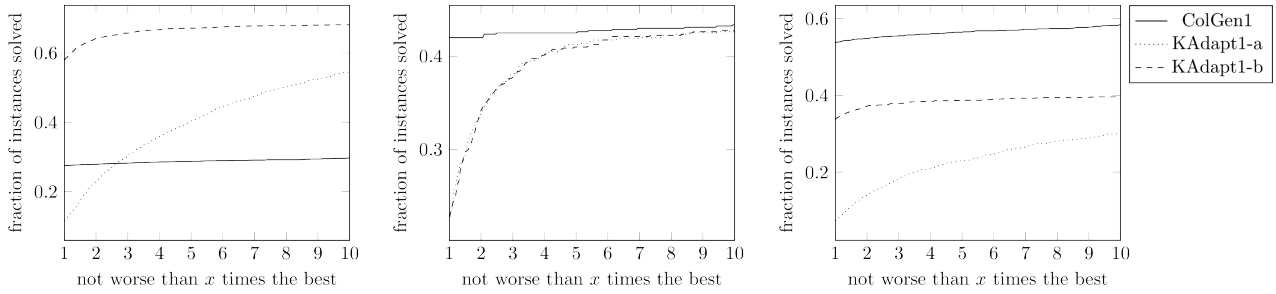
In our experiments, the search for $\widehat{K}^*$ is done iteratively starting from $K = 1$ and increasing K by one unit until one of the four conditions is reached:

- $(\mathcal{P}_K)^* \leq (\mathcal{P})^*, t(\mathcal{P}) \leq T$ and $t(\mathcal{P}_K) \leq T$: the two problems were solved optimally and we set $\widehat{K}^* = K$ (in this case, the equality of the objectives hold and the approximation is tight: $\widehat{K}^* = K^*$);
- $(\mathcal{P}_K)^* \leq (\mathcal{P})^*, t(\mathcal{P}) > T$ and $t(\mathcal{P}_K) \leq T$: the exact approach could not achieve and/or prove optimality, and $(\mathcal{P})^*$ equals the best upper bound found. We set $\widehat{K}^* = K$ and we know that $\widehat{K}^* \leq K^*$;
- $t(\mathcal{P}) \leq T$ and $t(\mathcal{P}_K) > T$: the finite adaptability approach could not achieve and/or prove optimality, the search for $\widehat{K}^*$ is stopped since increasing K typically increases the computation time to solve $(\mathcal{P}_K)$. We set $\widehat{K}^* = K$ and we know that $\widehat{K}^* \leq K^*$;
- $t(\mathcal{P}) > T$ and $t(\mathcal{P}_K) > T$: none of the two problems could be solved to proven optimality, the search for $\widehat{K}^*$ is stopped and the two methods are considered to perform as badly as the other. We set $\widehat{K}^* = K$ and we know that $\widehat{K}^* \leq K^*$.

Note that, as K^* is typically unknown, comparing $(\mathcal{P})$ with $(\mathcal{P}_{K^*})$ or $(\mathcal{P}_{\widehat{K}^*})$ in fact gives an advantage to the finite adaptability.

5.4. Comparison of the approaches for problem $(\mathcal{P})$

In Table 1, we present a comparison between finite adaptability from Section 3.1 (columns KAdapt1-a and KAdapt1-b) and the exact approach from Section 3.2 (columns ColGen1). We use the experimental protocol described above with a time limit $T = 1$ hour for each run. The two first columns describe the main characteristics of the instances considered (number of jobs and value of Γ). The left-hand part of the table gathers the percentage of instances that were not solved to optimality within the time limit. Then, the average computing time is reported (instances for which the time limit is reached count for 3600 seconds). In the right-hand part of the table, we finally reported the percentage of times in which one solution was found to be the fastest among the three. The same data are illustrated as performance profiles in Figure 4.



(a) All instances (b) 25-job instances (c) Instances with $\Gamma \leq |\mathcal{J}|/4$

Figure 4: Performance profiles [18] for different sets of instances. Each curve is associated with one method, and shows the fraction of instances it solves not slower than the value of the abscissa times the time required for the fastest approach

The finite adaptability approach KAdapt1-a solves all five-job instances, but fails to solve 11.25% of the instances for 10 jobs and $\Gamma = 2$. Less than 15% of the 25-job instances instances are solved by this method when $\Gamma \leq 4$. Indeed, we have noticed that the hardest instances correspond to those having a value of Γ such that the ratio $\Gamma/|\mathcal{J}|$ is around 0.3-0.4. For very small values of Γ the problem often becomes easy since a small number of critical jobs have to be found. For large values of Γ, to the contrary, the problem almost reduces to the deterministic problem where all the jobs are penalized. The in-between instances are the most challenging. Regarding KAdapt1-b, our results show that the method proposed in [34] is more efficient than the MILP formulation. Indeed, the average CPU time for KAdapt1-b is typically smaller than the time spent solving the MILP model for KAdapt1-a. Moreover, it solves significantly more hard instances than the MILP approach (see Figure 4c). The branch-and-price algorithm ColGen1 is able to solve all instances up to 20 jobs to optimality, and more than 88% of the 25-job instances. When finite adaptability is able to find the optimal solution, it is generally faster than ColGen1. Note that the reported computing times are those of the *last* run of KAdapt1 during the search for $\hat{K}^*$ (those can be obtained only if one is able to "guess" value of $\hat{K}^*$ beforehand, which is not the case in a practical context).

Table 2 shows the percentage of 25-job instances for which each method could find a feasible solution within the time limit T although it was not able to prove its optimality. KAdapt1-a always finds a feasible solution while it is clearly not the case for the convexification approach. Once again, note that the results for KAdapt1-a are obtained with the first value of K for which the execution time exceeded T, which may not be equal to K^*. This means that the cost reported is an upper bound of the actual cost of the first-stage solution found by K-adaptability (since the recourse used is heuristic if K is not large enough). For these instances, KAdapt1-a always finds a feasible solution but with a very large optimality gap: the gap between the lower and upper bounds of the MILP solver at the time limit is larger than 70% on average. This is partly explained by the poor linear relaxation of the MILP model. Conversely, the branch-and-price algorithm ColGen1 based on the convexification approach does not always find a feasible solution, but when it does, the optimality gap is often small (4.5% on average). Similarly, KAdapt1-b always finds a feasible solution within the given time limit.

In Table 3, we study the values of K that are needed to obtain the optimal solution with the $K-$adaptability method. For this purpose, we report for every value of K from 1 to K^*, the average gap between the value found by KAdapt1-a and the exact method ColGen1. We also compare computation times of KAdapt1-a and ColGen1, according to the different values of K. An important information gathered from the table is that a very large proportion of instances can be solved to optimality within one hour with a value of $K = 1$. This means that for many instances, the so-called static model where the recourse actions are decided a priori is sufficient to solve the problem.

It can also be noted that $K-$adaptability with a small value of K can be a good heuristic: for the 16 instances where $\hat{K}^* = 5$, setting K to 1 produces a gap of less than 7%, for computation times often two order of magnitude smaller than the time required to solve the problem optimally using the branch-and-price algorithm. This table also shows the high sensitivity of the $K-$adaptability approach with respect to

parameter K. For example, let us consider instances with $\hat{K}^* = 2$. When $K = 1$, 587 instances can be solved within one hour, whereas incrementing the value of K to 2 allows solving only 110 instances within the same time limit. KAdapt1-a appears, at first glance, to perform surprisingly better when $\hat{K}^*$ increases. Indeed, when $\hat{K}^* = 5$ and $K = 3$, and when $\hat{K}^* = 6$ and $K = 4$, its computation time looks significantly smaller than the one for ColGen1. But this anomaly is explained by the fact that only the instances that could be solved using KAdapt1-a with $K = 5$ and $K = 6$, respectively, are reported in these sections of the table. They are very likely to be well-suited for this approach, which explains the very good results when the value of K is smaller. Notice that we could determine the value of K^* for 2231 out of the 3200 instances, leaving this question open for the 969 others.

Finally, looking at the last column of table 3, one can see that KAdapt1-a, when $K = 1$, is significantly faster than ColGen1. In this case, the MILP model simplifies to a static robust optimization model (thanks to the structure of constraint (30)), which explains the very good performance. However, for the more complex settings, ColGen1 outperforms KAdapt1 by one to two orders of magnitude.

5.5. Comparison of the approaches for the order-fixing problem ($\widetilde{\mathcal{P}}$)

Table 4 compares computation times for approaches KAdapt2-a, KAdapt2-b and ColGen2. The values reported for KAdapt2-a and KAdapt-b are obtained with parameter $K = \hat{K}^*$ for each instance.

We can see that problem ($\widetilde{\mathcal{P}}$) is much more challenging to solve than ($\mathcal{P}$), since the $K-$adaptability methods reach limitations for 10-job instances while the branch-and-price algorithm ColGen2 is unable to solve some 15-job instances. This is mainly explained by the relatively large number of variables in the models. Indeed, while the number of first-stage variables was $\mathcal{O}(|\mathcal{J}|)$ for problem ($\mathcal{P}$), it is now $\mathcal{O}(|\widetilde{\mathcal{J}}|) = \mathcal{O}(|\mathcal{J}|^2)$. Hence, there is a significant number of additional variables subject to integrality constraints in problem ($\widetilde{\mathcal{P}}$) compared to ($\mathcal{P}$).

Table 5 shows the computation time ratio between KAdapt2-a and ColGen2. As for problem ($\mathcal{P}$), we can see that the closer K gets to $\hat{K}^*$, the faster the branch-and-price algorithm becomes compared to K-adaptability.

We also studied the increase of the objective function when one has to decide the sequence of the selected jobs before uncertainty is revealed. In Table 6, for several sizes of instances and several values of Γ, we report the average costs related to problem ($\mathcal{P}$) (column Free) and ($\widetilde{\mathcal{P}}$) (column fixed order), respectively.

Our experiments show that fixing the sequence of jobs beforehand only leads to a marginal increase of the cost on average. The largest gap we obtained was 0.45% for instances with ten jobs.

As a conclusion, it appears that for this problem, the cost to pay to keep the order fixed in the first-stage is not the cost of the solutions itself, but the practical difficulty to solve the optimization problem with state-of-the-art algorithms.

6. Conclusion

In this paper, we have described a robust version of the classical one-machine scheduling problem where one minimizes the weighted number of tardy jobs. Although solving general robust integer programs with integer recourse is typically Σ_P^2-hard, we were able to show that this problem is $\mathcal{NP}$-complete, and proposed two solution approaches: an exact reformulation which can be solved by means of the branch-and-price algorithmic procedure, and a (MILP) conservative approximation. Our computational experiments show that this problem is hard to solve in practice, since state-of-the-art methods may fail to solve 25-job instances in one hour.

Regarding the exact method we proposed, we think that the development of good heuristic procedures for the pricing problem may substantially improve the computing times. As for the conservative approximation, its main drawback is its poor linear relaxation, which is a known issue in the literature. We also have investigated another version of the problem where the sequencing decisions from the first stage cannot be modified. It appears that this version of problem is harder than the first: some 15-job instances are left unsolved by both approaches.

7. Acknowledgments

Experiments presented in this paper were carried out using the PLAFRIM experimental testbed, being developed under the Inria PlaFRIM development action with support from Bordeaux INP, LABRI and IMB and other entities: Conseil Régional d'Aquitaine, Université de Bordeaux, CNRS and ANR in accordance to the programme d'investissements d'Avenir (see `https://www.plafrim.fr/`).

Appendix A. Solution approaches for problem $(\widetilde{\mathcal{P}})$

Appendix A.1. Finite adaptability

In this section, we give a K-adaptability formulation of the second problem. The main constraints are identical to those derived in Section 3.1. Only the modified constraints and variables are explained below:

minimize

$$\sum_{j\in\mathcal{J}}[w_jU_j+(1-U_j)f_j+v_j]+\Gamma u-\sum_{q=1}^{K}\sum_{k\in\widetilde{\mathcal{J}}}f_k\psi_k^q \tag{A.1}$$

subject to

$(37)-(39)$

$$y_k^q\leq x_k\quad\forall k\in\widetilde{\mathcal{J}},q=1,\dots,K \tag{A.2}$$

$$z_k^q\leq y_k^q\quad\forall k\in\widetilde{\mathcal{J}},q=1,\dots,K \tag{A.3}$$

$$\sum_{k\in\mathcal{G}_j}x_k=1-U_j\quad\forall j\in\mathcal{J} \tag{A.4}$$

$(42)-(48)$

$$t_k^q\geq 0\quad\forall k\in\widetilde{\mathcal{J}},q=1,\dots,K \tag{A.5}$$

$$y_k^q\in\{0,1\}\quad\forall k\in\widetilde{\mathcal{J}},q=1,\dots,K \tag{A.6}$$

$$z_k^q\in\{0,1\}\quad\forall k\in\widetilde{\mathcal{J}},q=1,\dots,K \tag{A.7}$$

$$\psi_k^q\geq 0\quad\forall k\in\widetilde{\mathcal{J}},q=1,\dots,K \tag{A.8}$$

$$\zeta_k^q\geq 0\quad\forall k\in\widetilde{\mathcal{J}},q=1,\dots,K \tag{A.9}$$

$$U_j\in\{0,1\}\quad\forall j\in\mathcal{J} \tag{A.10}$$

$$u\geq 0 \tag{A.11}$$

$$v_j\geq 0\quad\forall j\in\mathcal{J} \tag{A.12}$$

$$\beta_q\geq 0\quad q=1,\dots,K \tag{A.13}$$

Here, the binary (first-stage) decision variable x_k represents the selection of the kth job occurrence in the non-decreasing order of the deadlines while U_j denotes the variable indicating if a job is executed tardy or not. Constraint (A.2) links the first- and second-stage decisions, constraint (A.3) enforces that a job is repaired only if it is scheduled and constraint (A.4) enforces that exactly one job occurrence is selected for on-time jobs. The other variables and constraints have the same meaning as in KAdapt1. This model will be referred to as KAdapt2.

Appendix A.2. Convexification of the recourse set

In a very similar way as what has been done for problem $(\mathcal{P})$, we can derive an exact formulation for this problem variant by using proposition 1 on the set of eligible second-stage solutions $\widetilde{\mathcal{Y}}$. Then, by enumerating the extreme points of the convex hull of $\mathcal{Y}$, we can derive the following model:

minimize

$$\sum_{j\in\mathcal{J}}[w_jU_j + f_j(1-U_j)] - \sum_{k\in\widetilde{\mathcal{J}}}\sum_{e\in E} f_k\mathbf{y}_k^e\alpha_e + \Gamma u + \sum_{j\in\mathcal{J}} v_j$$

subject to

$$\sum_{e\in E}\alpha_e = 1 \tag{A.14}$$

$$\sum_{e\in E}\mathbf{y}_k^e\alpha_e \leq x_k \quad \forall k\in\widetilde{\mathcal{J}} \tag{A.15}$$

$$\sum_{k\in\mathcal{G}_j} x_k = 1 - U_j \quad \forall j\in\mathcal{J} \tag{A.16}$$

$$u + v_j \geq \sum_{k\in\mathcal{G}_j}\left[\bar{\delta}_k\sum_{e\in E}(\mathbf{y}_k^e - \mathbf{z}_k^e)\,\alpha_e\right] \quad \forall j\in\mathcal{J} \tag{A.17}$$

$$x_k\in\{0,1\} \quad \forall k\in\widetilde{\mathcal{J}} \tag{A.18}$$

$$U_j\in\{0,1\} \quad \forall j\in\mathcal{J} \tag{A.19}$$

$$\alpha_e \geq 0 \quad \forall e\in E \tag{A.20}$$

$$u \geq 0 \tag{A.21}$$

$$v_j \geq 0 \quad \forall j\in\mathcal{J} \tag{A.22}$$

Again, decision vector α represents the convex combination multipliers from the inner description of $\text{conv}(\mathcal{Y})$ and u and v are the dual variables associated to the constraint $\xi\in\Xi$. Constraint (A.14) enforces that the second-stage variables must be a convex combinations of some extreme points. Constraint (A.15) links the first-stage variables with the second-stage variables while constraint (A.16) enforces that exactly one job occurrence per job is selected in the first stage. Finally, constraint (A.17) corresponds to the dualized cost implied by the venue of a scenario $\xi\in\Xi$.

This model will be referred to as ColGen2.

We solve this large-scale MILP model using a simple adaptation of the branch-and-price algorithm presented in Section 3.2.2. Algorithm 1 is modified by checking, at line 10, the integrality of both U^* and x^*, and lines 13 and 14 are adapted to branch either on a U- or an x-variable. Surprisingly, the pricing problem differs only by the value of the dual variable μ in input, which is now associated with constraint (A.15) instead of (58).

References

[1] Ali Allahverdi and John Mittenthal. Scheduling on a two-machine flowshop subject to random breakdowns with a makespan objective function. *European Journal of Operational Research*, 81(2):376 – 387, 1995.

[2] Mohamed Ali Aloulou and Federico Della Croce. Complexity of single machine scheduling problems under scenario-based uncertainty. *Oper. Res. Lett.*, 36(3):338–342, May 2008.

[3] Ayse N. Arslan and Boris Detienne. Decomposition-Based Approaches for a Class of Two-Stage Robust Binary Optimization Problems. *INFORMS Journal on Computing*, 34(2):857–871, March 2022. Publisher: INFORMS.

[4] Josette Ayoub and Michael Poss. Decomposition for adjustable robust linear optimization subject to uncertainty polytope. *Computational Management Science*, 13(2):219–239, 2016.

[5] Philippe Baptiste, Laurent Peridy, and Eric Pinson. A branch and bound to minimize the number of late jobs on a single machine with release time constraints. *European Journal of Operational Research*, 144(1):1 – 11, 2003.

[6] Pascale Bendotti, Philippe Chrétienne, Pierre Fouilhoux, and Alain Quilliot. Anchored reactive and proactive solutions to the cpm-scheduling problem. *European Journal of Operational Research*, 261(1):67 – 74, 2017.

[7] D. Bertsimas and C. Caramanis. Finite adaptability in multistage linear optimization. *IEEE Transactions on Automatic Control*, 55(12):2751–2766, Dec 2010.

[8] Dimitris Bertsimas and Angelos Georghiou. Design of near optimal decision rules in multistage adaptive mixed-integer optimization. *Operations Research*, 63(3):610–627, 2015.

[9] Dimitris Bertsimas, Eugene Litvinov, Xu Andy Sun, Jinye Zhao, and Tongxin Zheng. Adaptive robust optimization for the security constrained unit commitment problem. *IEEE Transactions on Power Systems*, 28(1):52–63, 2013.

[10] Dimitris Bertsimas, Ebrahim Nasrabadi, and Sebastian Stiller. Robust and Adaptive Network Flows. *Operations Research*, 61(5):1218–1242, October 2013.

[11] Dimitris Bertsimas and Melvyn Sim. The Price of Robustness. *Operations Research*, 52(1):35–53, February 2004.

[12] J. Birge, J. B. G. Frenk, J. Mittenthal, and A. H. G. Rinnooy Kan. Single-machine scheduling subject to stochastic breakdowns. *Naval Research Logistics (NRL)*, 37(5):661–677, 1990.

[13] Marin Bougeret, Artur Pessoa, and Michael Poss. Robust scheduling with budgeted uncertainty. *Discrete Applied Mathematics*, 08 2018.

[14] Matthias Claus and Maximilian Simmoteit. A note on σ_2^p-completeness of a robust binary linear program with binary uncertainty set. *Operations Research Letters*, 48(5):594–598, September 2020.

[15] Stéphane Dauzère-Pérès and Marc Sevaux. Using lagrangean relaxation to minimize the weighted number of late jobs. *Naval Research Logistics*, 50(3):273–288, 2003.

[16] Stéphane Dauzère-Pérès. Minimizing late jobs in the general one machine scheduling problem. *European Journal of Operational Research*, 81(1):134 – 142, 1995.

[17] Boris Detienne. A mixed integer linear programming approach to minimize the number of late jobs with and without machine availability constraints. *European Journal of Operational Research*, 235:540–552, 2014.

[18] Elizabeth D. Dolan and Jorge J. Moré. Benchmarking optimization software with performance profiles. *Mathematical Programming*, 91(2):201–213, Jan 2002.

[19] A. Garcia-Piquer, J. C. Morales, I. Ribas, J. Colomé, J. Guàrdia, M. Perger, J. A. Caballero, M. Cortés-Contreras, S. V. Jeffers, A. Reiners, P. J. Amado, A. Quirrenbach, and W. Seifert. Efficient scheduling of astronomical observations - Application to the CARMENES radial-velocity survey. *Astronomy & Astrophysics*, 604:A87, August 2017. Publisher: EDP Sciences.

[20] Angelos Georghiou, Angelos Tsoukalas, and Wolfram Wiesemann. Robust Dual Dynamic Programming. *Operations Research*, April 2019. Publisher: INFORMS.

[21] R.L. Graham, E.L. Lawler, J.K. Lenstra, and A.H.G.Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. In P.L. Hammer, E.L. Johnson, and B.H. Korte, editors, *Discrete Optimization II*, volume 5 of *Annals of Discrete Mathematics*, pages 287 – 326. Elsevier, 1979.

[22] Grani A. Hanasusanto, Daniel Kuhn, and Wolfram Wiesemann. K-adaptability in two-stage robust binary programming. *Operations Research*, 63(4):877–891, 2015.

[23] J.R. Jackson. *Scheduling a production line to minimize maximum tardiness.* Research report. Office of Technical Services, 1955.

[24] Ruiwei Jiang, Muhong Zhang, Guang Li, and Yongpei Guan. Two-stage network constrained robust unit commitment problem. *European Journal of Operational Research*, 234(3):751–762, 2014.

[25] Hiroshi Kise, Toshihide Ibaraki, and Hisashi Mine. A solvable case of the one-machine scheduling problem with ready and due times. *Oper. Res.*, 26(1):121–126, February 1978.

[26] Nikolaos H. Lappas and Chrysanthos E. Gounaris. Multi-stage adjustable robust optimization for process scheduling under uncertainty. *AIChE Journal*, 62(5):1646–1667, 2016. _eprint: https://aiche.onlinelibrary.wiley.com/doi/pdf/10.1002/aic.15183.

[27] Rym M'Hallah and R.L. Bulfin. Minimizing the weighted number of tardy jobs on a single machine with release dates. *European Journal of Operational Research*, 176(2):727 – 744, 2007.

[28] J v Neumann. Zur theorie der gesellschaftsspiele. *Mathematische annalen*, 100(1):295–320, 1928.

[29] Artur Pessoa, Ruslan Sadykov, Eduardo Uchoa, and François Vanderbeck. Automation and combination of linear-programming based stabilization techniques in column generation. *INFORMS Journal on Computing*, 30(2):339–360, 2018.

[30] Michael L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer Publishing Company, Incorporated, 6th edition, 2016.

[31] Laurent Péridy, Eric Pinson, and David Rivreau. Using short-term memory to minimize the weighted number of late jobs on a single machine. *European Journal of Operational Research*, 148(3):591 – 603, 2003.

[32] Ruslan Sadykov. A branch-and-check algorithm for minimizing the weighted number of late jobs on a single machine with release dates. *European Journal of Operational Research*, 189(3):1284 – 1304, 2008.

[33] Alexander Shapiro. Minimax and risk averse multistage stochastic programming. *European Journal of Operational Research*, 219(3):719–726, June 2012.

[34] Anirudh Subramanyam, Chrysanthos E. Gounaris, and Wolfram Wiesemann. K-adaptability in two-stage mixed-integer robust optimization. *Mathematical Programming Computation*, 12(2):193–224, November 2019.

[35] Marjan van den Akker, Han Hoogeveen, and Judith Stoef. Combining two-stage stochastic programming and recoverable robustness to minimize the number of late jobs in the case of uncertain processing times. *Journal of Scheduling*, 21(6):607–617, December 2018.

[36] Ruby van Rooyen, Deneys S. Maartens, and Peter Martinez. Autonomous observation scheduling in astronomy. In Alison B. Peck, Robert L. Seaman, and Chris R. Benn, editors, *Observatory Operations: Strategies, Processes, and Systems VII*, volume 10704, pages 393 – 408. International Society for Optics and Photonics, SPIE, 2018.

[37] F. Vanderbeck. Bapcod – a generic branch-and-price code, 2005.

[38] Jian Yang and Gang Yu. On the robust single machine scheduling problem. *Journal of Combinatorial Optimization*, 6(1):17–33, Mar 2002.

[39] Long Zhao and Bo Zeng. Robust unit commitment problem with demand response and wind energy. In *Power and Energy Society General Meeting, 2012 IEEE*, pages 1–8. IEEE, 2012.

Table 1: CPU execution times for solving problem ($\mathcal{P}$)

		Unsolved within $T = 1$ hour (%)			Average CPU time (s.)			Fastest approach (%)		
$\lvert\mathcal{J}\rvert$	Γ	KAdapt1-a	Kadapt1-b	ColGen1	KAdapt1-a	KAdapt1-b	ColGen1	KAdapt1-a	KAdapt1-b	ColGen1
5	1	0	0	0	0	0	2	10	90	0
	2	0	1	0	0	0	2	4	96	0
	3	0	0	0	0	0	2	1	99	0
Avg. $\lvert\mathcal{J}\rvert = 5$		0	0	0	0	0	2	5	95	0
10	1	6	12	0	85	59	13	11	76	12
	2	11	24	0	49	30	15	18	64	19
	3	5	10	0	17	1	11	5	85	10
	4	1	2	0	0	1	8	1	96	2
	5	1	2	0	7	0	7	1	96	2
	6	0	0	0	0	0	7	1	99	0
	7	0	0	0	0	0	7	1	99	0
Avg. $\lvert\mathcal{J}\rvert = 10$		4	7	0	22	13	10	6	88	7
15	1	35	28	0	443	207	43	8	50	42
	2	57	69	0	452	27	69	4	28	69
	3	46	49	0	16	1	64	1	50	49
	4	29	29	0	55	0	47	4	68	29
	5	12	12	0	0	0	29	6	81	12
	6	10	10	0	0	0	23	2	88	10
	7	2	2	0	0	0	21	4	94	2
	8	0	0	0	0	0	17	4	96	0
	9	0	0	0	0	0	16	1	99	0
	10	0	0	0	0	0	16	1	99	0
Avg. $\lvert\mathcal{J}\rvert = 15$		19	20	0	97	24	35	3	75	21
20	1	66	45	0	693	184	118	2	44	54
	2	88	86	0	171	39	190	4	12	84
	3	86	91	0	306	42	273	5	6	89
	4	71	76	0	132	20	332	6	19	75
	5	55	56	0	20	1	346	5	39	56
	6	35	35	0	0	0	269	16	49	35
	7	20	20	0	0	0	188	20	60	20
	8	5	5	0	0	0	127	31	64	5
	9	0	0	0	0	0	67	34	66	0
	10	0	0	0	0	0	46	28	72	0
Avg. $\lvert\mathcal{J}\rvert = 20$		43	42	0	132	29	196	15	43	42
25	1	82	59	9	384	345	375	8	30	62
	2	95	95	12	78	45	623	11	9	79
	3	91	92	16	5	0	629	15	16	69
	4	78	78	19	0	0	613	18	25	56
	5	66	66	21	0	1	538	25	29	46
	6	57	57	20	0	1	534	29	29	41
	7	39	39	18	0	0	442	33	38	30
	8	31	31	12	0	1	442	38	37	26
	9	15	15	10	0	1	426	50	37	13
	10	6	6	5	0	2	286	59	35	6
Avg. $\lvert\mathcal{J}\rvert = 25$		56	54	14	47	40	491	29	29	43

From left to right : the number of jobs, the uncertainty budget, the average computation time (when less than $T = 1$ hour) for the K-adaptability approach, the convexification-based branch-and-price algorithm, the percentage of times one method was found to be the most efficient and the percentage of instances which could not be solved within $T = 1$ hour. For the sake of readability, all numbers are rounded to the closest integer.

Table 2: Feasible solutions found for $(\mathcal{P})$, over instances that could not be solved to optimality by the method within the time limit $T = 1$ hour

		Feasible solutions found (%)		
$\|\mathcal{J}\|$	Γ	KAdapt1-a	KAdapt1-b	ColGen1
25	1	100	100	29
	2	100	100	30
	3	100	100	15
	4	100	100	7
	5	100	100	6
	6	100	100	6
	7	100	100	14
	8	100	100	10
	9	100	100	38
	10	100	100	25

From left to right : the number of jobs, the uncertainty budget and the percentage of instances for which a feasible solution could be found within $T = 1$ hour over the instances which could not be solved optimally within $T = 1$ hour. Numbers are rounded to the closest integer.

Table 3: The cost of approximating with finite adaptability for problem $(\mathcal{P})$

$\hat{K}^*$	K	# Instances	Approximation gap (%)	Time ratio
1	1	1989	0	0.03
2	1	587	0.00	0.03
	2	110	0	5.23
3	1	368	5.86	0.01
	2	368	1.43	6.00
	3	90	0	14.59
4	1	123	6.3	0.01
	2	123	2.06	0.09
	3	123	0.6	14.23
	4	32	0	13.27
5	1	16	6.85	0.01
	2	16	2.25	0.04
	3	16	0.67	0.82
	4	16	0.24	34.76
	5	3	0	29.28
$\geq$6	1	3	1.92	0.01
	2	3	1.7	0.02
	3	3	0.58	0.05
	4	3	0.13	0.31
	5	3	0.02	5.87
	6	2	0	2.06

From left to right : the value of $\hat{K}^*$ (*i.e.* the value of K required for K-adaptability to be equivalent to $(\mathcal{P})$ or a lower bound on this value), the value of K which was used, the number of instances with this value of $\hat{K}^*$ which were solved using Kadapt1 with that value of K within $T = 1$ hour, the approximation gap computed as $|KAdapt1^* - ColGen1^*|/|ColGen1^*|$, the computation time ratio computed as $t(KAdapt1)/t(ColGen1)$.

Table 4: Computation times for solving problem $(\widetilde{\mathcal{P}})$

		Unsolved within 1 hour (%)			Average CPU time (s.)			Fastest approach (%)		
$\lvert\mathcal{J}\rvert$	Γ	KAdapt2-a	KAdapt2-b	ColGen2	KAdapt2-a	KAdapt2-b	ColGen2	KAdapt2-a	KAdapt2-b	ColGen2
5	1	0	0	0	0	0	1	12	87	1
	2	0	0	0	0	0	1	14	86	0
	3	0	0	0	0	0	1	7	93	0
Avg. $\lvert\mathcal{J}\rvert = 5$		0	0	0	0	0	1	11	88	0
	1	0	11	0	14	27	28	22	72	5
	2	1	22	0	22	18	24	28	59	14
	3	1	9	0	9	4	11	16	78	6
	4	0	2	0	43	2	8	6	91	2
	5	0	1	0	37	40	4	12	86	1
	6	0	0	0	0	0	3	6	94	0
	7	0	0	0	0	0	3	8	92	0
Avg. $\lvert\mathcal{J}\rvert = 10$		0	7	0	18	13	12	14	82	4
15	1	11	18	8	191	118	243	14	61	25
	2	36	55	16	212	280	219	25	27	48
	3	25	38	12	154	0	158	19	46	35
	4	19	26	8	131	0	169	21	56	23
	5	9	10	4	24	0	77	18	74	9
	6	5	6	4	8	0	47	20	76	5
	7	2	2	2	0	0	61	23	74	2
	8	0	0	0	0	0	50	19	81	0
	9	0	0	0	0	0	15	12	88	0
Avg. $\lvert\mathcal{J}\rvert = 15$		12	17	6	80	44	115	19	65	16

From left to right : the number of jobs, the uncertainty budget, the average computation time (when less than $T = 1$ hour) for each method, the percentage of times one method was found to be the most efficient and the percentage of instances which could not be solved within the time limit $T = 1$ hour. All numbers are rounded to the closest integer.

Table 5: The cost of approximating with finite adaptability for problem ($\widetilde{\mathcal{P}}$)

$\hat{K}^*$	K	# Instances	Approximation gap (%)	Time ratio
1	1	1165	0	0.23
2	1	87	4.88	0.06
	2	87	0	0.11
3	1	79	6.37	0.02
	2	76	0.97	0.09
	3	65	0	3.74
4	1	93	7.15	0.04
	2	93	1.65	0.09
	3	73	0.4	2.01
	4	59	0	5.79
5	1	40	7.93	0
	2	40	2.72	0.03
	3	40	0.83	0.53
	4	40	0.19	8.75
	5	14	0	14.17
6	1	14	5.84	0.01
	2	14	2.98	0.02
	3	14	1.12	0.09
	4	14	0.41	0.81
	5	14	0.09	13.24
	6	6	0	35.19
$\geq$7	1	2	0.37	0.01
	2	2	0.37	0.02
	3	2	0.37	0.14
	4	2	0.37	0.65
	5	2	0.14	5.52
	6	2	0.03	78.13

From left to right : the value of $\hat{K}^*$ (*i.e.* the value of K required for K-adaptability to be equivalent to ($\mathcal{P}$) or a lower bound on this value), the value of K which was used, the number of instances with this value of $\hat{K}^*$ which were solved using Kadapt2 with that value of K within $T = 1$ hour, the approximation gap computed as $|KAdapt2^* - ColGen2^*|/|ColGen2^*|$, the computation time ratio computed as $t(KAdapt2)/t(ColGen2)$.

Table 6: Fixed-order solutions cost analysis

$\|\mathcal{J}\|$	Γ	Objective cost Free (ColGen1)	Objective cost Fixed order (ColGen2)	Gap (%)	N. Instance
5	1	70.39	70.40	0.00	80
5	2	75.27	75.28	0.01	80
5	3	75.94	75.94	0.00	80
10	1	144.76	145.14	0.26	80
10	2	165.92	166.21	0.18	80
10	3	171.73	171.80	0.04	80
10	4	173.24	173.26	0.01	80
10	5	173.61	173.61	0.00	80
10	6	173.70	173.70	0.00	80
10	7	173.70	173.70	0.00	80
15	1	192.83	193.32	0.25	74
15	2	232.46	233.06	0.26	67
15	3	248.97	249.57	0.24	70
15	4	253.39	253.78	0.15	74
15	5	254.77	254.91	0.05	77
15	6	255.35	255.37	0.01	77
15	7	255.74	255.74	0.00	78
15	8	256.98	256.98	0.00	80
15	9	256.98	256.98	0.00	80
15	10	256.98	256.98	0.00	80

From left to right: the number of jobs, the uncertainty budget, the average objective costs of free solutions and fixed-ordered solutions, the relative gap between the two, the number of instances which where accounted for in the computation (i.e., instances which could be solved within the time limit $T = 1$ hour for both problems).