Robust decision trees for the multi-mode project scheduling problem with a resource investment objective and uncertain activity duration

Tom Portoleau, Christian Artigues, Romain Guillaume

Robust decision trees for the multi-mode project scheduling problem with a resource investment objective and uncertain activity duration

Tom Portoleau[1,2], Christian Artigues[1], Romain Guillaume[2]

[1] LAAS-CNRS, Université de Toulouse, CNRS, Toulouse, 31400 France
[2] ADRIA, Université de Toulouse - IRIT, Toulouse Cedex 1, France

Abstract

In this paper, we advocate the use of robust decision trees for the problem of assembly line scheduling problem, which is modeled as a multi-mode resource constrained project scheduling Problem, with uncertainty about activity duration and a resource investment objective. The idea of a robust decision tree is that at each node, the decision maker has access to some information about the ongoing scenario. Depending on the different information they could obtain, a partial solution is proposed. Considering that the level of uncertainty is lowered, the new partial solution is less conservative and improve the robustness guarantee. However, since all accessible information may not be relevant, we turned the information selection part into an optimization problem. We first introduce the industrial context and the problem at stake. Then we propose algorithms to solve the information selection problem, using constraint programming, and then to build such robust decision trees. Finally we provide experimental results for benchmarks instances and industrial instances.

1 Introduction

In real-world scheduling problems, such as in aeronautical assembly line scheduling problems, several parameters are subject to uncertainty. Moreover, the knowledge the decision maker has about these parameters takes place in a dynamic context, i.e. actual parameter values are revealed progressively over time. Taking into account of both uncertain and dynamic aspects in scheduling problems solution approaches is necessary to control the risk and the flexibility of the schedule. Historically, two general approaches emerged to deal with uncertainty in scheduling problems [27]. On the first hand there are the proactive methods. The goal is to compute a robust baseline schedule, that ensures the feasibility and a worst-case objective value. However, it is known that such

approaches tend to produce overly conservative solutions [21]. On the second hand, reactive methods aim at compute quickly new solutions to deal with the unexpected and adapt the schedule to the scenario that emerges. Nevertheless these approaches do not have in general worst-case guarantees, which does not necessarily fit in practice industrial cases where a bad management of the worst case is not conceivable. In order to make a compromise between conservatism and too much optimism, hybrid methods were developed, and the term proactive-reactive scheduling emerged [27]. For instance, in [10, 9] the authors propose an integrated proactive-reactive approach, where the goal is to find the best policy, which in their case corresponds to a couple (initial solution, set of reactions), given a transition cost between solutions. With a different perspective the authors of [11] proposed what they call the Just In Case approach, in which they compute a multiple contingent schedule with the application of action planning for an autonomous telescope scheduling problem, where the worst cases are critical in the sense that the telescope does not carry out any of the observations that had been planned. Inspired by this method, and borrowing ideas from multi-stage robust approaches [3], we propose here an approach based on the computation of a decision tree. The idea behind decision trees is that they are easily usable for a decision-maker, and make it possible to deal with small but identified uncertainties by dynamically proposing solutions according to the arrival of new information, while guaranteeing the quality of solutions in the worst case. We use this model on an aeronautical assembly line scheduling problem which will be modelled as a multi-mode resource-constrained project scheduling problem (MMRCPSP) problem with a resource investment objective. Then, we compare our model to a proactive-reactive algorithm with computational experiments carried out on MMRCPSP benchmark instances from the PSPLIB [16] and real industrial instances from an aeronautical assembly line scheduling problem [6].

The outline of the paper is the following. Section 2 introduces the industrial context of our approach. In section 3 we go deeper into the literature review concerning multi-stage scheduling techniques from which our approach is inspired. Section 4 formally introduces the MMRCPSP, uncertainty and information modelling. In section 5, we present an information-based model of the MMRCPSP and formulate the robust partition problem, which is the core problem of our approach. Section 6 introduces the robust decision tree. In Section 7 we assess the practical appeal our approach comparing it to an information-based proactive-reactive scheduling algorithm.

2 Industrial Context

In this paper we are interested on aeronautical assembly line scheduling problem. The optimisation of assembly lines forms a family of problems that have been extensively studied for decades. Two distinct types of problems have emerged, assembly line balancing problems and assembly line scheduling problems. Line

balancing problems are more of a general assembly-line design nature, and can have two objectives: at a fixed number of workstations, to maximise the production rate, or conversely at a fixed production rate, to minimise the number of workstations. We will not focus on these problems in this paper, but an extensive literature review can be found in [26]. Before presenting the scheduling problems, it should be noted that an attempt has been made to integrate the two problems into a single one. For instance, in [1] and [25] the authors consider problems where the duration of activitiess depends on the way they are scheduled, with respectively sequence-dependant setup-time for the former, and taking into account the deterioration effect for the latter. A more concrete approach, rather engineering oriented, has been proposed in [6] where the authors develop a methodology for Airbus assembly lines so that their tools integrate the aspects of the two previous problems. Regarding the scheduling aspect on assembly lines, a wide range of techniques have been developed. In [24], the authors propose to improve the "beam search" method, a heuristic based on the Branch and Bound algorithm, and then applied it to an assembly line scheduling problem. In [29] the authors present an industrial problem from the assembly lines of the car manufacturer Toshiba, and propose to solve it by Lagrangian relaxation taking advantage of the geometrical characteristics of their problem.

Now let us introduce more specifically the problem at stake. This problem consist in building a schedule that satisfies a fixed makespan while minimising the peak resource usage, which may correspond to the number of operators to be employed during the scheduling period, also called the resource investment problem [12]. In this paper, inline with our target application on aeronautical assembly lines, we consider moderate hazards, which only impact the duration of the activities. Given an initial schedule, any activity duration increase may degrade the quality of this schedule. If we know the maximal duration of each activity, a robust initial schedule can be built as in the standard robust optimization scheme, i.e. by seeking a good performance on the worst case scenario, in this case obtained by setting each activity to its maximal duration.

On an assembly line, the same scheduling problem with small variations repeats itself time after time (as similar aircraft are assembled) whereas the risks are constantly changing, which can cause the quality of the initial robust schedule to fluctuate greatly over these assembly cycles. Moreover, over-estimating the needs of certain resources, as it is necessary in traditional robust optimization, can be very costly. This is the well-known conservatism issue of standard robust optimization.

An other important characteristic of the considered industrial context, is that decision makers have often access to information about the course of the planning only at specific times, for instance every evening before a team change when the leaving team reports its actions during the day. At this specific time, the decision makers can use their know-how to adapt quickly the schedule on the fly. However, having access to these information can be costly, and all information may not be particularly relevant.In addition, the decision maker may not want to change the schedule too often to ensure a certain stability in the operators planning. For these reasons, one wishes to look for the most relevant

information, i.e. the information that, if taken into account for rescheduling, would best improve the worst case performance.

activities	i	1	2	3	4
maximal duration	$p_i^{\max}$	6	10	2	4
Operators 1	$b_{i,1}$	1	1	3	0
Operators 2	$b_{i,2}$	1	2	0	4

Table 1: A small RCPSP example

Example 2.1. *To exemplify this, let us consider a toy instance of a (single-mode) resource-constrained project scheduling problem (RCPSP): we are given a set of activities $\mathcal{J}$, a makespan $C_{\max}$ and two decision time points $t_0 = 0$ and t_1. There is a set of renewable resources $\mathcal{R}$ of limited capacities and each activity $i \in \mathcal{J}$ requires during its processing a given non-negative amount $b_{i,k}$ of each resource $k \in \mathcal{R}$ (see the instance data in Table 2). In this example, we have four activities and two resources, each corresponding to an operator type (e.g. electrician and mechanic). Activity 1 requires one operator of each type during its processing, i.e. during at most 6 hours. Our goal is to find a schedule that ends before $C_{\max}$ while minimising the sum of the maximum resource usages from t_0 to t_1 and from t_1 to the end of the schedule. The rationale behind this objective is that we assume that operators are hired depending on the maximum resource use of the computed schedule during the interval defined by two consecutive decision time points. Let us now consider a fixed makespan $C_{\max} = 12$ and, additionally, the precedence constraints $1 \to 3$ and $1 \to 4$. A schedule s is the assignment of integer start times S_i in $[0, C_{\max} - 1]$ to each activity $i \in \mathcal{J}$. A duration scenario is a vector $p = (p_i)_{i \in \mathcal{J}}$ with $p_i \in [p_i^{\min}, p_i^{\max}] \cap \mathbb{N}$ where $p_i^{\min} = 1$ and $p_i^{\max}$ are known parameters. A schedule is feasible for a scenario p if it satisfies the precedence constraints $S_3 \geq S_1 + p_1$ and $S_4 \geq S_1 + p_1$. For a scenario p, the use of resource $k \in \mathcal{R}$ at time $t \in [0, C_{\max} - 1]$ by schedule S is given by*

$$R_{k,t} = \sum_{i \in \mathcal{J}, t \in [S_i; S_i + p_i - 1]} b_{i,k}$$

An optimal schedule for scenario p is the one that minimises the objective function

$$z = \max_{t \in [0, t_1 - 1]} [R_{1,t} + R_{2,t}] + \max_{t \in [t_1, C_{\max} - 1]} [R_{1,t} + R_{2,t}]$$

The optimal schedule S^0 for scenario $p = (p_i^{\max})_{i \in \mathcal{J}}$, with an objective value of 13, is given by the Gantt diagrams displayed in Figures 1 and 2:

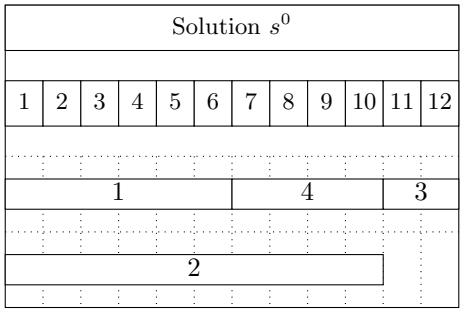

Figure 1: Gantt Diagramm of s^0

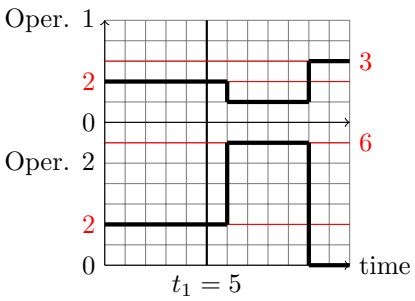

Figure 2: Resource profile of s^0

We assume that schedule s^0 has been computed before time $t_0 = 0$ where no information on the planning state was available since the schedule was not started. The schedule is followed as is from t_0 to $t_1 - 1$ and the 2 operators of each type are hired in this interval. Now suppose that at time $t_1 = 5$ the decision maker has the opportunity to ask two questions to increase the knowledge of parameters $p^{\min}$ and $p^{\max}$, but wonders which one is more pertinent: A)"Is activity 1 already completed ? (i.e. $p_1^{\max} \leq 5$)" or B)"Will activity 2 last more or less than 8 units of time ? (i.e. $p_2^{\min} \geq 8$ or $p_2^{\max} \leq 8$)". Let us review both cases:

- *The decision maker chooses to ask A). Then, no matter what is the answer to this question, we can not compute a better solution, because even if the answer is yes, this new information only allows the decision maker to shift activities 3 and 4 to the right, which does not improve the objective.*
- *The decision maker chooses to ask B). If the answer to this question is yes, the decision maker can switch at t_1 to solution s^1 (see Fig. 3 and 4), which has an objective value of 12 which makes it a strictly better solution than S^0. Remark that to implement such a solution, the schedule (and operator use) must of course be kept unchanged (frozen), i.e. in $[t_0, t_1 - 1]$ independently of the information obtained at t_1 on what happened in this interval. This is why the schedule before time $t_1 = 5$ is displayed in gray.*

It can be clearly seen from that example that the answer to question A) is of no use, unlike the answer to question B). From the point of view of the decision-maker this kind of information is interesting, since it suggests that if it is possible to monitor activity 2 in particular so that it runs smoothly, then it is possible to improve the schedule. Note that we are only interested here in the selection of questions from a fixed set.

In this case by selecting in advance the question B) to be asked at t_1, a decision tree can be defined. At the root node corresponding to time t_0, activities 1 and 2 are started. This node has a unique child node corresponding to time t_1. From this node, a branch corresponding to the no answer to question B)

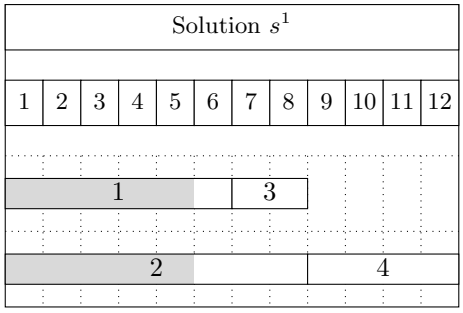

Figure 3: Gantt Diagram of s^1

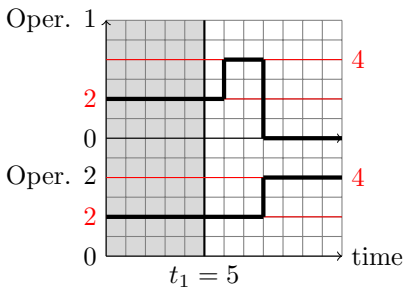

Figure 4: Resource use of s^1

will keep on following schedule s^0 while another branch corresponding to the yes answer to question B) will switch to schedule s^1.

Given the hypothesis that at some point during the schedule some information about the progress of the planning is available and that the schedule can be changed, such a decision tree computed in advance allows decision makers to promptly retrieve high quality solutions while using as little information as possible.

In the rest of this paper, we will consider that the question asked by the decision maker about the duration of a activity is: "Will this activity last less than 80% of its estimated duration ?". The 80% is arbitrary and has no major impact on our experimental results. After a literature review, we will more formally describe the robust decision tree generation process.

3 Litterature Review/Robust Optimization and Scheduling under Uncertainty

Another way to see the Just In Case approach presented in section 1 is to see it as a very specific case of multi-stage robust adaptable optimization [3]. In those approaches a part of the uncertain data is revealed sequentially -at each stage-, and some recourses, or alternative solutions, are pre-computed to adapt to the known realization of the ongoing scenario. The ideal goal of robust adaptable optimization is to compute a fully adaptable solution, that is to say that an optimal schedule is known for every possible scenario realization. However this is untractable in practice. In [2], the authors show that even an uncertain two-stage problem is NP-Hard to solve in the general case. Several methods to approximate the fully adaptable solution have been explored yet. A first family of methods aims at limiting the possible recourses. In [13] the authors introduce K-adaptability where only a finite number K of policies are available at each stage. Another way of restricting the possible recourses is to consider only affine -or linear- decision rules [20, 19]. The second family of methods approximate

the fully adaptable solution by clustering the scenarios into a finite partition. Unfortunately, it was shown in [4] that even computing a partition of size 2 is NP-Hard in the general case. Several ways of calculating the partition have been studied. In [5] a small number of reference scenario is computed, and each scenario is placed in the same subset as its closest reference scenario. In [28], the authors consider the specific case where the subsets of the partition are all hyper-rectangles in the set of scenarios.

The method we present in this paper is close to the methods of the second family, in the sense that we seek to sequentially calculate efficient partitions using information as the scenario is revealed. However, there are some key differences with the methods mentioned above. Firstly, in these methods, the authors consider a robust objective (such as minmax) on the subsets of the partition they calculate. However, this type of objective does not apply to problems where the objective is monotonous with respect to uncertain variables (for example, the makespan for a scheduling problem) and where the set of scenarios is not discrete. Indeed, in this case one can easily find a worst-case scenario, which will "absorb" the scenarios present in the same subset of the partition as it. Moreover, since it is the worst-case scenario, the minmax value remains the same.

Let us illustrate the first difference again Example 2.1. At decision time point $t_1 = 5$, a multi-stage adjustable robust optimization approach would ask itself how to partition the set of scenarios so as to propose reactions that lower the worst-case makespan. Suppose that the choice is between partitions A) and B) as already described. None of these partitions would results in a decrease in the global worst case. In case A), which partitions the scenario set in the two subsets defined by $p_1 \in [1, 5]$ on one hand and $p_1 = 6$ on the other hand, we have seen that the worst-case $C_{\max}$ will inevitably remain equal to 13 for both subsets. Case B) partitions the scenario set in the two subsets defined by $p_2 \in [1, 8]$ on one hand and $p_2 \in [8, 10]$ on the other hand. Even if the first subset decreases the worst case makespan to 12, the second subset still includes the worst case scenario with all durations equal to their maximum value, so the global worst-case schedule remains equal to 13.

However partition B) is clearly a better one from a practical point of view as switching to S^1 as soon as the duration realization is revealed at t_1 to belong to the second subset will allow resource savings. For this reason, we will introduce a new criterion to compare scenario partitions with each other. It is not focused on exploiting the obtained information to lower the global worst case but rather in identifying the information that will allow to opportunistically switch to a better solution when possible and so reduce the price of robustness in practice.

Secondly, the notion of "cost of knowledge" is sometimes considered in the literature, but when this is the case, it is introduced in the objective function of the partitioning problem as a penalty. We preferred to add it as a second objective, from a lexicographical point of view. Indeed, we are more interested in avoiding to react to useless information than to consider actual information costs.

Lastly, in the majority of works the multistage uncertain problems are stud-

ied from the angle of mathematical programming. In this paper, motivated by a specific industrial context -aeronautical assembly lines-, we propose an approach based on constraint programming which we can extend naturally to integrate the information selection problem.

As stated in the introduction, our work is partly inspired by the proactive and reactive resource-constrained project scheduling problem considered in [10, 9] in the context of uncertain durations. They propose to define a baseline schedule and a set of reactions in front of a realization of the durations. A reaction is a transition from one schedule to an other one. The activity durations follow discrete distribution and the transitions take place inside a pre-computed set of schedules The goal is to minimise an expected cost of the reactions given the distributions. The cost measures fixed reaction costs and also the distance from the baseline to the realized schedule. In this paper, we assume that probability distributions on the activity durations are not available in accordance with the target aeronautical applications but we are aware of the worst-céase scenario. Hence we can follow a baseline worst case schedule and react to improve it for some realizations using as less information as possible, thus minimising indirectly the schedule stability.

The general model we propose was already introduced, formally, in [22] and applied to a simple one-machine scheduling problem. However in that work we dealt only with the case where the problem at stake can be solved in polynomial time, even in its uncertain version. As the MMRCPSP is a NP-hard problem even in its deterministic form, the core algorithm of the approach is different than the one presented in our previous work. If the main idea behind the robust decision tree remains the same, the goal of the present paper is to show how it can be applied to an industrial problem.

4 Uncertainty, information and scheduling models

4.1 Uncertainty, robustness and information

In this paper and for the considered problem we suppose that the activities duration are uncertain. As mentioned previously, the goal of our model is to take into account small and frequent uncertainties, which occur with each repetition of the problem. Moreover, since building an accurate probabilistic model requires a large amount of data, it is easier and more convenient to ask the decision maker for bounds over activity duration. For these reasons, we consider interval uncertainty. More formally, it means that for any activity $i \in \mathcal{J}$ of the problem, its duration $p_i \in [p_i^{min}, p_i^{max}]$. We call any realization of all the uncertainties a scenario. We denote by Ω the set of all possible scenarios and ω any of its elements. We can then write the robustness criterion we consider in

this paper, introduced as the absolute robustness criterion in [18] :

$$s^* = \underset{s \in \mathcal{X}}{\operatorname{argmin}} \max_{\omega \in \Omega} f(\omega, s) \tag{1}$$

where $\mathcal{X}$ is the set of solutions, and $f(\omega, s)$ the objective value of the solution s under scenario ω. We refer to s^* (resp. to the maximum value of $f(\omega, s^*)$) as the minmax or robust solution (resp. the minmax or robustness value) over the set of scenarios Ω.
We can notice that in our case, the objective function that we consider is increasing with the duration of the activities. Thus, to find a robust solution for the MMRCPSP problem with uncertainties, it is sufficient to solve it in its worst case scenario, where all durations are at their maximum value.

Now that uncertainties are clearly set out, we can rigorously define what we call information. There is a set T of predefined time points (e.g. ends of shifts), at which the decision maker can ask several questions, the answer to one question bringing an information about a activity. Let $\mathcal{J}_q$ denote the set of activities that are in process during interval $[t_q, t_q + \Delta] \subset \mathcal{T}$ in the baseline schedule, i.e.

$$\mathcal{J}_q = \{i \in \mathcal{J} | S_i - \Delta \leq t_q < S_i + p_i\}$$

In the considered information model, we assume that only questions on the activities in $\mathcal{J}_q$ are allowed. The rationale behind this is that if a activity is completed at time t_q in the baseline schedule is is necessary completed in the realized scenario, as the worst case has been considered to build the schedule. If an activity is not scheduled to start in interval $[t_q, t_q + \Delta]$, we assume that the decision maker does not have enough feedback from the assembly line to obtain reliable information on this activity.

In our model, having access to an information about an activity i means that the decision maker is able to know the answer to the question "Does the activity i last more or less than X_i unit of time ?", with $X_i \in [p_i^{min}, p_i^{max}]$. Consequently, one information allows us to split in two the set of scenarios : the scenarios such that $p_i \in [p_i^{min}, X_i]$ and the scenarios such that $p_i \in [X_i, p_i^{max}]$.

Let $K \subseteq \mathcal{J}_q$ a set of activities on which questions can be asked at time point t_q. Having the answers to $|K|$ questions, i.e. having access to $|K|$ information, allows the decision maker to distinguish between $2^{|K|}$ subsets of scenarios corresponding to the $2^{|K|}$ possible subsets of activities getting a "yes" answer. Let $K_+ \subseteq K$ denote one of the possible yes-answer subsets. We denote by Ω_{K_+} the subset of scenarios issued from the usage of the information obtained from K_+ to reduce uncertainty. More formally,

$$\Omega_{K_+} = \prod_{i \in K_+} [p_i^{min}, X_i] \times \prod_{i \in \mathcal{J} \setminus K_+} [p_i^{min}, p_i^{max}]$$

In our case as the worst-case objective value can always be obtained when each activity has its maximum duration, having for an activity i the information that $p_i \in [X_i, p_i^{max}]$ or having no information on i is the same as this does not change

the worst case. One could note that using these notations we have that if K_+ and K'_+ denote two sets of information about uncertainties over a set of scenarios Ω then $K_+ \subset K'_+ \Leftrightarrow \Omega_{K'_+} \subset \Omega_{K_+}$ and, more generally, $\Omega_{K_+ \cup K'_+} = \Omega_{K_+} \cap \Omega_{K'_+}$.

In the examples and experiments, we will take $\Delta = 0$ and $X_i = \lceil 0.8 p_i^{max} \rceil$.

Let us illustrate these notions on example 2.1 at time $t_1 = 5$ and the baseline schedule S^0. The decision maker is allowed to ask questions on $\mathcal{J}_1 = \{1, 2\}$ as these two activities are in process in t_1 according to S^0. Taking as mentioned $X_i = \lceil 0.8 p_i^{\max} \rceil$, we obtain $X_1 = 5$ and $X_2 = 8$. Let us take $K = \{1, 2\}$. Considering the possible yes-answer subsets $K_+ = \{1\}$, $K'_+ = \{2\}$, $K''_+ = \{1, 2\}$, we can distinguish between the 4 scenarios $\Omega_{\{1\}} = [1, 5] \times [1, 10]$, $\Omega_{\{2\}} = [1, 8] \times [1, 8]$, $\Omega_{\{1,2\}} = [1, 5] \times [1, 8]$ and $\Omega_{\emptyset} = \Omega = [1, 8] \times [1, 10]$.

We can check that $\Omega_{\{1,2\}} \subset \Omega_{\{1\}}$, $\Omega_{\{1,2\}} \subset \Omega_{\{2\}}$ and that $\Omega_{\{1,2\}} = \Omega_{\{1\}} \cap \Omega_{\{2\}}$.

4.2 Scheduling model: the robust MMRCPSP

The problem we address in this paper deals with the optimisation of resource utilisation for activity scheduling on an aeronautical assembly line [7, 8]. In this problem each activity is subject to precedence constraints, time-lag constraints and has several possible modes, which can make its use of the different resources vary. Several resources are at stake: the resources of areas that will be represented by limited cumulative constraints, and human resources, which correspond to operators with different skills. As already explained, the objective is to minimise the sum of the maximum use of each type of operators inside each interval defined by two consecutive decision time points. As stated in [8], the industrial problem under study can be modeled as a multi-mode resource-constrained project scheduling problem (MMRCPSP) with a resource investment objective, such as in [12]. This problem is known to be NP-Hard [15].

There is a set of activities $\mathcal{J}$ and a set of resources $\mathcal{R}$. Each activity $i \in \mathcal{J}$ is associated with a set $\mathcal{M}_i$ of modes such that a mode $m \in \mathcal{M}_i$ for an activity $i \in \mathcal{J}$ defines the activity duration $p_{i,m}$ and the amount $b_{i,k,m}$ that the activity requires on resource $k \in \mathcal{R}$. We have a discrete time horizon H with special so-called decision time points $T \subset H$ with $H = \{0, \ldots, |H|\}$ and $T = \{t_0 = 1, \ldots, t_1, t_{|T|} = |H|\}$. The set of resources $\mathcal{R}$ comprise two distinct subsets: the subset of investment resources $\mathcal{R}^I$, for which the peak usage must be minimised inside each decision interval and the subset of standard renewable resources $\mathcal{R}^\rho$, each being associated with a capacity B_k that cannot be exceeded. There is a set of precedence constraints E.

By defining decision variable S_i, $\forall i \in \mathcal{J}$, which gives the start time of each activity $i \in \mathcal{J}$ and $m_i \in \mathcal{M}_i$, which gives the mode selected for activity $i \in \mathcal{J}$, the MMRCPSP with resource investment objective can be formally stated as follows.

$$\min \quad z = \sum_{k \in \mathcal{R}^I} \sum_{q=1}^{|T|} \max_{\tau \in t_{q-1}}^{t_q - 1} R_{k,\tau} \tag{2}$$

$$\text{s. t.} \quad S_j - S_i \geq p_{i,m_i} \qquad \forall (i,j) \in E \tag{3}$$

$$R_{k,\tau} = \sum_{i \in \mathcal{J}, \tau \in [S_i; S_i + p_{i,m_i} - 1]} b_{i,k,m_i} \qquad \forall k \in \mathcal{R}, \forall \tau \in H \tag{4}$$

$$R_{k,\tau} \leq B_k \qquad \forall k \in \mathcal{R}^\rho, \forall \tau \in H \tag{5}$$

$$S_i \in H \qquad \forall i \in \mathcal{J} \tag{6}$$

$$m_i \in \mathcal{M}_i \qquad \forall i \in \mathcal{J} \tag{7}$$

The objective (2), to be minimised is equal to the sum on each resource of the maximal peaks inside each interval. Constraints (3) are standard precedence constraints. Constraints (4) computes, for each resource $k \in \mathcal{R}^I$ the total usage $R_{k,\tau}$ at each time $\tau \in H$. For standard renewable resources, constraints (5) prevent the resource usage to exceed the resource capacity. Constraints (6,7) give the variable domains.

About uncertainty modeling and given the fact that activities have multiple modes, we now consider that $p_{i,m}^{min}$ is the minimum activity duration in mode m while $p_{i,m}^{max}$ is its maximum duration in mode m. Consequently, the set of scenarios Ω is the set of matrices $(p_{i,m})_{j \in \mathcal{J}, m \in \mathcal{M}_i}$ where $p_{i,m} \in [p_{i,m}^{min}, p_{i,m}^{max}]$.

Consider now an MMRCPSP instance I_Ω obtained by fixing $p_{i,m} = p_{i,m}^{max}$, for each activity $i \in \mathcal{J}$ and each mode $m \in \mathcal{M}_i$. As stated in Section 4.1, the optimal solution to the deterministic MMRCPSP with I_Ω as input is the same as for the absolute robust problem (1) with $\mathcal{X} = (3-7)$ on instance I subject to uncertainties Ω. Let RobustMMRCPSP denotes a procedures that solves the problem taking as input the instance I_Ω and outputs the solution $s = ((S_i)_{j \in \mathcal{J}}, (m_i)_{j \in \mathcal{J}}, z)$.

5 Robust scenario partition based on information selection

5.1 Robust scheduling with limited information selection

As explained in Section 2, the industrial context assumes that the decision maker has access to information -which may come with a cost- during the schedule, but the problem of information selection might not be trivial. We propose a way of formalising the information selection problem as an extension of the MMRCPSP model introduced in the previous section. We suppose that we are given an instance of the MMRCPSP, a decision time t_q, and some characteristics $\bar{s}$ of the solution computed at the previous decision time point, the set of activities K from which we can get information at time t_q, and an integer Q. The objective is to compute a robust solution with a better worst-case guarantee value than $\bar{s}$

using at most Q information from K, i.e. building a yes-answer subset $K_+ \subseteq K$ with $|K_+| \leq Q$. To achieve that we introduce a new scheduling problem. The model extends the MMRCPSP as follows.

For each activity i in K we define a set of duplicate modes $\mathcal{M}'_i$ such that $m' \in \mathcal{M}'_i$ has the same resource requirement as is duplicate $m \in \mathcal{M}_i$ but a duration equal to $p_{i,m'} = X_i$, which is the reduced worst case duration in case of the yes-answer to the question asked for i. Hence selecting mode m' for activity $i \in K$ means that $i \in K_+$.

As the previous solution must be fixed before t_q, we need to that purpose the start times and the resource investment cost before t_q, given by $\bar{s}$, where

$$\bar{s} = \left((\bar{S}_i)_{i \in \mathcal{J}}, \bar{R}, \bar{z}\right)$$

where $\bar{S}_i$ denote the previous start time of activity i, $\bar{R}$ denotes the contribution to the objective function of all intervals in $[t_0, t_q - 1]$, and $\bar{z}$ denote the total objective function value of the previous solution. The problem can now be modeled as the following MMRCPSP:

$$\begin{aligned}
\min \quad & leximin\,(z, |K_+|) && && (8)\\
\text{s. t.} \quad & (3-6) && &&\\
& z \geq \bar{R} + \sum_{k \in \mathcal{R}^I} \sum_{\rho=q}^{|T|} \max_{\tau \in t_{\rho-1}}^{t_\rho - 1} R_{k,\tau} && && (9)\\
& z \leq \bar{z} - 1 && && (10)\\
& K_+ = \{i \in K | m_i \in \mathcal{M}'_i\} && && (11)\\
& |K_+| \leq Q && && (12)\\
& S_i = \bar{S}_i && \forall i \in \mathcal{J}, \bar{S}_i \leq t_q - 1 && (13)\\
& S_i \geq t_q && \forall i \in \mathcal{J}, \bar{S}_i \geq t_q && (14)\\
& m_i \in \mathcal{M}_i \cup \mathcal{M}'_i && \forall i \in K && (15)\\
& m_i \in \mathcal{M}_i && \forall i \in \mathcal{J} \setminus K && (16)
\end{aligned}$$

The goal of objective (8) is to choose the smallest subset of information $|K_+|$ (i.e. the smallest number of selected duplicate modes) that produces the solution with the best objective value z. Constraints (9) defines the main objective z with a constant part equal to $\bar{R}$ and a variable part equal to the resource investment for all intervals in $[t_q, H]$. Constraint (10) state that the worst-case objective must be improved by getting information from K. Constraint (11) defines the set K_+ as the set of selected duplicate modes. Constraint (12) limits the selected number of duplicate modes, i.e. the information K_+ used from K under limit Q. Constraints (13) prevent the start times of activities that start before the decision point t_q to be changed. Constraints (14) prevent an activity that was not planned before t_q in $\bar{s}$ to be shifted before decision time t_q. Constraints (7) in the MMRCPSP are replaced by constraints (15) that state that activities in

K have duplicate modes, and constraints (16) that keep the original mode set of the other activities.

Another way to model constraint (10) is to add a nonrenewable resource that is used exclusively by modes in $\mathcal{M}'_i$, $i \in K$. Each mode m requires one unit of this resource, and only Q of it is available. In both way we can note that the problem is not fundamentally different from the original MMRSPCP and extends it naturally.

We suppose that we have an algorithm, called 1SubsetSolution, to solve this problem, taking as input an MMRCPSP instances I_Ω associated with a scenario set Ω, the set of activities K available for questions, the maximal number of questions Q, the previous solution $\bar{s}$ and the time point q. The algorithm outputs the new solution s and the used information K_+.

An example of how to apply mode duplication to an instance is given in Example 5.1.

Example 5.1. *We consider once again the instance from Example 2.1. We recall that at time $t_1 = 5$ the decision maker has the possibility to ask two questions, about the duration of activities 1 and 2 ($K = \{1, 2\}$). It follows that activities 1 and 2 have duplicate modes:*

$$\mathcal{M}_1 = \{1\}, \mathcal{M}'_1 = \{2\}, \mathcal{M}_2 = \{1\}, \mathcal{M}'_2 = \{2\}$$

The problem of information selection can be solved by solving, using the described previously, the following instance:

activities	1		2		3	4
modes	1	2	1	2	1	1
maximal duration	6	5	10	8	2	4
Operators 1	1	1	1	1	3	0
Operators 2	1	1	2	2	0	4

If we run our model with this instance, the solver will return the solution S^1, with a total of 1 information used ($K_+ = \{2\}$)

In the solution of this problem, there are two things of interest: the subset of information that induces a subset of scenarios, and the solution which is robust for this subset of scenarios.

5.2 Robust Partition

The algorithm just presented makes it possible to answer the problem of information selection at a given decision time.

Actually, solving this problem at time point t_q, we obtain a partition of the set of scenarios Ω in two subsets Ω_{K+}, on which the obtained solution s minimises the worst-case objective given that at most Q information are uses

and $\Omega \setminus \Omega_{K+}$ that still contains the global worst case scenario on which previous solution $\bar{s}$ still minimises the worst-case objective.

However, it could be objected that one answer of this algorithm is far too optimistic, in the sense that the proposed solution is only feasible if the answer to all the selected questions is yes. To overcome this, we propose to use this algorithm several times to enlarge the partition size, adding each time new constraints preventing the use of a subset of questions already selected. Each successive call to this algorithm makes it possible to calculate a different subset of questions and solutions. Now let us suppose that the precedent algorithm is called $L-1$ times, with L an integer greater than 1. We denote by $(K_+^l, s^l)_{i=1}^{L-1}$ the set of information and the robust solution returned by the l-th call of the precedent algorithm. Let s^0 denote the initial robust solution for Ω. Then each scenario in $\bigcup_{l=1}^{L-1} \Omega_{K_+^l}$ is covered by at least one solution from $\{s^l | l = 1, \ldots, L-1\}$. Moreover, in order to make sure that the same subset of information (or any set it is included in) is not selected at each iteration, the already used sets of information K_+^l are stored in a list $\mathcal{K}_+$.

The constraint

$$\bigvee_{\kappa_+ \in \mathcal{K}_+} |\{i \in \kappa_+ | m_i \in \mathcal{M}_i'\}| \leq |\kappa_+| \tag{17}$$

is added to the model (8–16) to guarantee that property, which means that the solution procedure 1SubsetSolution takes also as input set $\mathcal{K}_+$.

To build a partition of Ω, we have to adjust the $(\Omega_{K_+^l})_l$ family to satisfy these conditions. By construction, if $l' < l$ then the solution $s_{l'}$ has a better worst-case objective value than s_l. Then the ordered family of subset $(\Omega_l)_{l=1}^{L-1}$ defined by

$$\Omega_l = \Omega_{K_+^l} \setminus \bigcup_{l'=1}^{l} \Omega_{K_+^{l'}} \forall l = 1, \ldots, L-1$$

ensures that for any $l \leq L-1$, every scenario in Ω_l is covered by exactly one solution –namely s^l–, and that no other solution $s^{l'}$ has a proven better worst-case guarantee than s^l. Finally, we can extend the $(\Omega_l)_l$ family with $\Omega_L = \Omega \setminus \bigcup_{i=1}^{L-1} \Omega_l$, and covers its scenarios by the solution s^0, which is the solution that was implemented until the moment decision t_q. Now, the family $(\Omega_m)_{i=1,\ldots,L}$ is clearly a partition of Ω, and each scenario is covered by one and only one solution from $(s^l)_{l=0,\ldots,L-1}$. One could note that the parameter L bounds the size of the partition. The pseudo code of this algorithm, returning partition $\mathcal{P}$ and the associated set of robust solutions $\mathcal{S}$ computed for each subset of scenarios in the partition, is shown in Algorithm 1.

Example 5.2. *Let us consider the instance from the running example once again, in the same context as in Example 2.1 and 5.1, but this time, we want to compute a robust partition. For the sake of the example, we set* $L = 2$ *and* $Q = 1$. *First we solve* 1*SubsetSolution*. *The solver finds a solution, and returns* $(K_+^1 = \{2\}, s^1)$. *Consequently, the first subset of the partition* $\mathcal{P}$ *is*

$\Omega_{K_+^1} = [1,6] \times [1,8] \times [1,2] \times [1,4]$ *and the solution that covers it is* s^1*. Since* $L = 2$*, we have to complete the partition with the subset of scenarios* $\Omega \backslash \Omega_{K_1}$*, and cover it with the solution* s^0 *(the initial robust solution). Finally the algorithm returns* $\mathcal{P} = (\Omega_{K_1}, \Omega \setminus \Omega_{K_1})$*, which is clearly a partition of* Ω*, and* $\mathcal{S} = (s^1, s^0)$ *which were proven to have the best worst case guarantee over the subsets of* $\mathcal{P}$ *in the same order.*

Algorithm 1 Robust Partition

Require: an instance I, Ω a set of scenarios, s^0 a robust solution on Ω, K a set of activities, a time t_q, $\bar{R}$ the resource investment cost before t_q and two integers L and Q.

$\mathcal{P} \leftarrow \{\}$, $\mathcal{S} \leftarrow \{\}$, $\mathcal{K}_+ \leftarrow \{\}$, $\bar{s} \leftarrow \left((S_i^0)_{i \in \mathcal{J}}, \bar{R}, z^0\right)$

for $1 \leq l < L - 1$ **do**

  $s^l, K_+^l \leftarrow 1\textsc{SubsetSolution}(I_\Omega, K, Q, \mathcal{K}_+, \bar{s}, t_q)$

  if No solution found **then**

    break

  end if

  $\mathcal{S} \leftarrow \mathcal{S} \cup \{s^l\}$, $\mathcal{K}_+ \leftarrow \mathcal{K}_+ \cup \{K_+^l\}$, $\mathcal{P} \leftarrow \mathcal{P} \cup \{\Omega_{K_+^l} \setminus \bigcup_{\Omega_{K_+} \in \mathcal{P}} \Omega_{K_+}\}$

end for

$\mathcal{P} \leftarrow \mathcal{P} \cup \{\Omega \setminus \bigcup_{\Omega_{K_+} \in \mathcal{P}} \Omega_{K_+}\}$

$\mathcal{S} \leftarrow \mathcal{S} \cup \{s_0\}$

return $\mathcal{P}, \mathcal{S}$

6 Robust Decision Tree

In this section we present the Robust Decision Tree. The idea of such a tree is that at each node, corresponding to a decision point t_q, we suppose that some information are accessible. With those information we are able to split the set of scenarios and compute a new robust solution for every subset of the partition. The way we compute a "good" partition is introduced in Section 5.2. Figure 5 shows an example of a Robust Decision Tree for the example from Section 2. We assume that the moments of decision, that is to say, the moments when information are available are given. The tree is to be read from top to bottom, and each moment corresponds to a level in the tree, so that going down the tree means going forward in time. This makes it very easy to use for a decision-maker.

The planning starts with the solution from the root. At each decision moment, the decision maker asks the operators the questions (selected in the tree) about the state of the planning. Then they are able to choose the branch corresponding to the answers to the questions, offering a robust continuation to the planning and leading to another node, and so on.
As one can imagine, since building such a tree require to solve multiple times a NP-Hard problem, it takes a very long time. But as the scenarios encapsulate

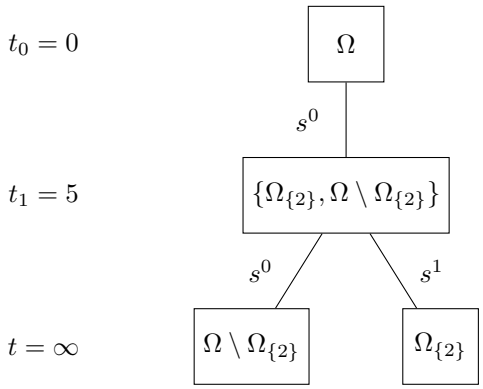

Figure 5: Robust Decision Tree for the example from Section 2.

all the uncertainties, the tree has to be computed only once, in a offline phase. Once it is done, the tree is reusable as long as the uncertainties, and the data of the scheduling problem at stake are unchanged.
It can be noted that the initial robust solution is always in the tree, because the worst case scenario is always "contained" in one of the branches of the tree, and is always feasible, regardless of the current scenario. In the same way, the solution proposed after choosing a branch becomes a new reference solution, which remains equally feasible in the corresponding sub-tree. Thus, if several branches are available after a decision moment, the local robust solution computed at the previous decision node will be the "default" solution. Therefore, whatever the level in the tree, it is necessary that the solutions proposed in the different branches are strictly more interesting than the local robust solution. In our case, "more interesting" means with a better robustness value (i.e. a better minmax value) as explained in section 4.2. By doing this, it follows that by construction, the worst case value can only improve as we go down the tree. In sum, robust decision trees break the conservatism inherent to classical robust approaches while keeping bounds over worst case scenarios.

Moreover, we can derive from these trees a new way to define how critical an activity is. In the classical definition in project scheduling an activity is critical if an increase of its duration necessarily increases the total duration of the project [14]. Here we can adapt this definition to our context: an activity is critical if a decrease of its duration allows the calculation of a better planning. Since it is possible to visualise in advance the questions to be asked at the next decision moment, it is interesting for the decision maker to monitor - if possible - precisely the activities concerned. If these activities are completed faster than the initial estimate, this increases the chances of being in a subset of the scenarios covered by a branch of the tree that significantly improves the worst case of the objective function.

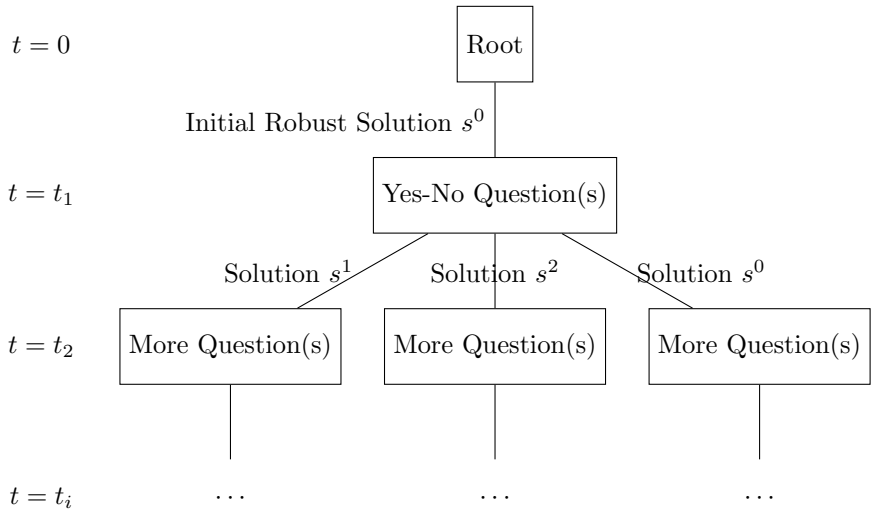

Figure 6: General Idea of a Robust Decision Tree

As explained in Algorithm 2, the construction of the tree is done level by level, each level corresponding to a decision time point t_q. The robust decision tree algorithm takes as parameters the list $(t_q) \in T$ of the decision moments, Q the maximum number of information that can be used at each node and L the maximum size of the partition calculated at each node by solving the Robust Partition Problem. These parameters limit the size of the tree. The depth of the tree is $|T|$, and L is the maximum branching factor of the tree which makes the total number of nodes in the tree is bounded by $\frac{L^{|T|+1}-1}{L-1}$, and the total number of leaves is bounded by $L^{|T|}$. Moreover, one could note that since asking a question splits in two the set of scenarios (and more generally if m questions are asked, the set of scenarios is split in 2^m subsets), the maximum size of the partition L cannot be greater than 2^Q.

Example 6.1. *We illustrate the robust decision tree generation process on a larger example with* $|\mathcal{J}|= 10$ *activities,* 2 *investment resources* $\mathcal{R}^I = \{1, 2\}$ *and* 2 *renewable resources* $\mathcal{R}^\rho = \{3, 4\}$ *with capacities* $B = (20, 19)$*. Finally we have the precedence constraints:*

$$E = \{(1, 7), (1, 10), (2, 4), (2, 5), (2, 7), (3, 5), (4, 6), (4, 9), (5, 6), (6, 8), (7, 8, (7, 9)\}.$$

The other instance characteristics are displayed in Table 2.

At the root node, a robust schedule s^0 *is computed by taking the maximal duration of each mode as the worst case scenario. A schedule of cost 74 is obtained with activity start times* $(2, 1, 0, 3, 5, 9, 8, 16, 12, 9)$ *and modes* $(1, 1, 1, 1, 1, 1, 1, 2, 2, 1)$*. The Gantt diagram and the corresponding peaks on the two investment resources are displayed in Figure 7. For sake of conciseness the renewable resource usage is not displayed.*

Activity	mode	$p_{i,m}^{min}$	$p_{i,m}^{max}$	$b_{i,1,m}$	$b_{i,2,m}$	$b_{i,3,m}$	$b_{i,4,m}$
	1	1	1	6	6	6	0
1	2	2	4	4	3	5	0
	3	4	9	4	1	1	0
	1	1	1	6	6	0	9
2	2	3	6	5	6	0	7
	3	4	9	4	5	0	2
	1	1	1	8	5	7	0
3	2	3	6	5	4	5	0
	3	4	9	4	3	0	6
	1	2	5	5	5	8	0
4	2	4	8	5	4	0	3
	3	4	8	3	4	0	6
	1	2	4	6	7	0	7
5	2	2	5	6	5	7	0
	3	4	9	5	4	0	5
	1	3	7	3	4	7	0
6	2	4	9	3	3	5	0
	3	5	10	2	1	0	6
	1	2	4	5	4	7	0
7	2	3	6	5	4	0	7
	3	3	7	4	3	6	0
	1	3	7	2	7	0	1
8	2	3	7	3	6	4	0
	3	4	9	2	5	3	0
	1	2	5	10	7	2	0
9	2	2	5	10	7	0	6
	3	5	10	10	6	2	0
	1	1	2	2	6	0	7
10	2	4	8	2	5	0	4
	3	5	10	1	5	6	0

Table 2: A 10-activity MMRCPSP instance

Algorithm 2 Robust Decision Tree

Require: an instance I, Ω a set of scenarios, and two integers L and Q.

```
s ← RobustRCPSP(I_Ω)
create robust decision tree 𝒯 with root node (Ω, s, 0)
initialize node queue 𝒬 ← (Ω, s, 0)
for 1 ≤ q < |T| do
  𝒬' ← {}
  while 𝒬 is not empty do
    dequeue node (Ω, s, R̄) from 𝒬
    K ← {j ∈ 𝒥 | S_i < t_q < S_i + p_{i,m_i}}
    R̄' ← R̄ + Σ_{j∈R^I} max_{τ=t_{q-1}}^{t_q-1} R_{k,τ}
    𝒫, 𝒮 ← RobustPartition(I, Ω, s, K, t_q, R̄', L, Q)
    while 𝒫 is not empty do
      pop scenario set Ω' from 𝒫 and solution s' from 𝒮
      add node (Ω', s', R̄') to the children of (Ω, s, R̄) in 𝒯
      enqueue (Ω', s', R̄') in 𝒬'
    end while
  end while
  𝒬 ← 𝒬'
end for
return 𝒯
```

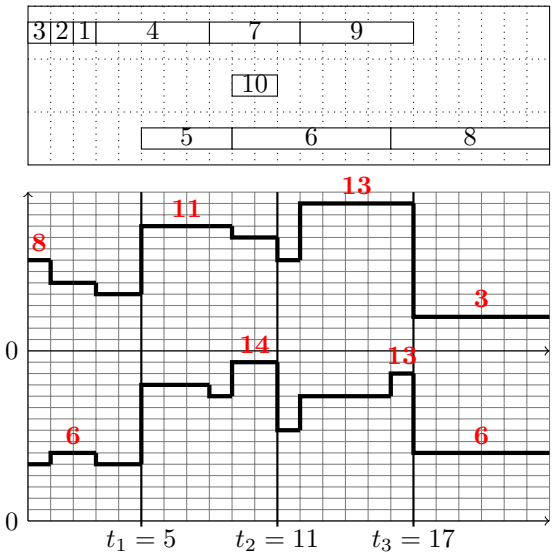

Figure 7: Robust solution s^0 of value 74 at root node Ω ($t_0 = 0$)

Jumping to the first decision point $t_1 = 5$, the set of activities on which information can be obtained is $K = 4, 5$. Taking again $Q = 1$ and $L = 2$, all

the modes of these activities are duplicated and the robust partition problem is solved. The best way found of improving the solution is to obtain information from activity set $K_+^1 = \{5\}$ *with a reduced maximal solution* $p_{5,2}^{max} = 2$ *in mode 2 (scenario set* Ω_5*), which allows to switch to the solution* s^1 *of value 70 represented in Figure 8. The updated mode assignment is* $(1, 1, 1, 1, 2, 1, 1, 1, 1)$*. Note that the initial mode of activity 5 can be still changed (from mode 1 to mode 2) as the activity is not already started. This gives the left child node identified by* Ω_5*,* $t_1 = 5$ *and* s^1*. As* $L = 2$*, the right child considers all scenarios for which* $p_{5,1}^{min} = 3$*,* $p_{5,2}^{min} = 3$ *and* $p_{5,3}^{min} = 6$*, i.e. activity 5 will not be finished at 80% of its initial planned duration in any of its modes (*$\Omega \setminus \Omega_{\{5\}}$*), which is covered by solution* s^0*.*

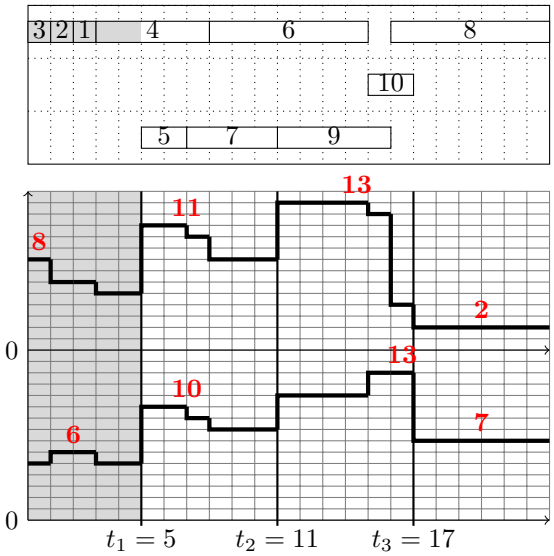

Figure 8: Robust solution s^1 of value 70 at node Ω_5 ($t_1 = 5$)

The left and right nodes are pushed in $\mathcal{Q}$ *and node (*$\Omega_{\{5\}}$*,* t_1*,* s^1*) is popped for decision point* $t_2 = 11$*. The set of activities on which information can be obtained is* $K = \{6, 9\}$ *and the robust partition problem selects* $K_+^1 = \{9\}$ *with a reduced duration of* $p_{9,2}^{max}$ *in mode 2 (scenario set* $\Omega_{\{5,9\}}$*). As previously, we had* $S9 = t_2$ *so mode of 9 can be changed. The obtained solution* s^2 *is displayed in Figure 9. The new assigned modes are* $(1, 1, 1, 1, 2, 1, 1, 2, 2, 1)$*. This gives left child* $(\Omega_{\{5,9\}}, t_2, s^2)$*. The right child considers the remaining scenarios for which the duration of activity 9 is larger than 80% of its initial duration* $(\Omega_{\{5\}} \setminus \Omega_{\{9\}}, t_2, s^1)$*. Both nodes are pushed into* $\mathcal{Q}$

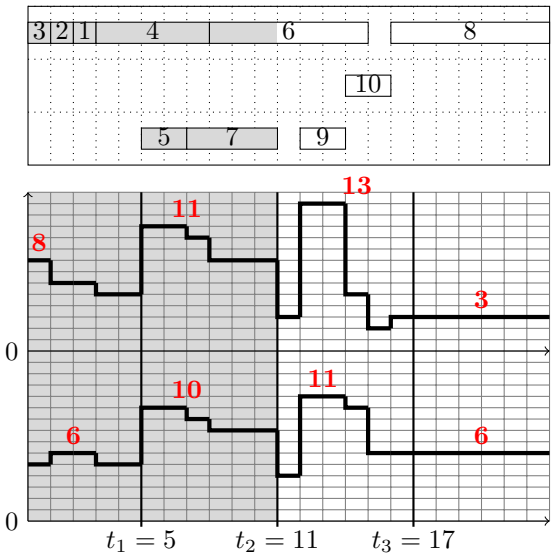

Figure 9: Robust solution s^2 of value 68 at node $\Omega_{5,9}$ ($t_2 = 11$)

The process is iterated, generating the robust decision tree displayed in Figure 10. When node $(\Omega_{\{5,9\}}, t_2, s^2)$ *is popped at* $t_3 = 17$*, no scenario partition can improve the current solution as only activity 8 starts its processing at* t_3*. So only one child is generated. Similarly for all popped right nodes, no other solution improvement can be reached. Finally the robust decision tree has 9 nodes, allowing to switch to better solutions at* t_1 *and* t_2 *according to scenario partition* $\Omega_{\{5,9\}}, \Omega_{\{5\}} \setminus \Omega_{\{9\}}, \Omega \setminus \Omega_{\{5\}}$*. Using only information from 2 activities, the robust decision tree reaches, for the considered scenario partitions, a gain of up to 8.10% on the objective function value.*

7 Experimentation

7.1 Instances

In order to test our decision trees, we built our own instances based on the MMRCPSP instances from the PSPLIB [23]. In particular we were interested in the J10 and J30 instances. These two sets differ from each other by two parameters, shown in the following table:

Set	J_{10}	J_{30}
Number of act. (N)	10	30
Network Compl. (C)	1,8	1,5

The number of activities is self explanatory, and the network complexity is the average number of non-redundant arcs per node in the precedence graph.

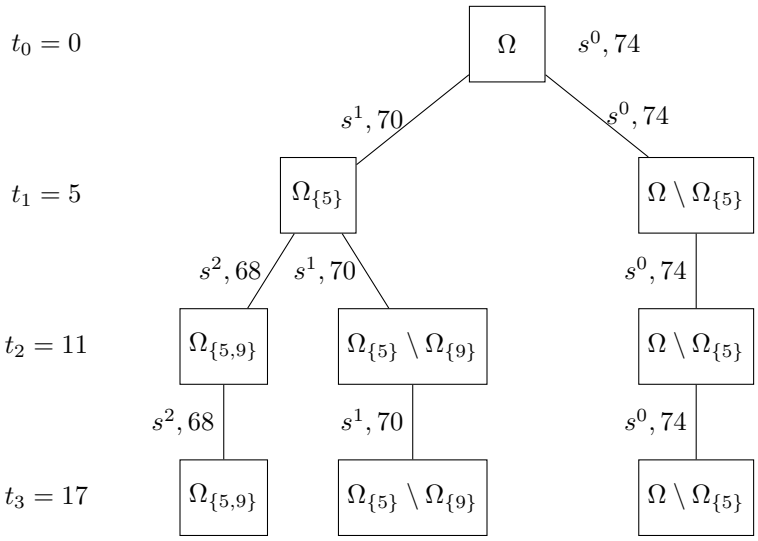

Figure 10: Robust Decision Tree for the 10-activities example

One could note that the overall problem complexity is not monotonous with respect to this parameter: when this parameter is very high (very few feasible solutions) or very low (many feasible solutions) search trees based methods tend to give good results, whereas an average value of this parameter tends to create difficult instances. All the details concerning the parameters used to generate these instances can be found in [17] and in [23]. To build our instances, we had to slightly modify those from the PSPLIB. These instances considered two types of resources, renewable and non-renewable. We chose to consider renewable resources as our $\mathcal{R}^\rho$ resources, keeping the same capacities as in the original instance. As for the non-renewable resources of the original instance, we change their semantics and consider them them as the investment resources $\mathcal{R}^I$ for our problem. The idea behind the structure of the PSPLIB instances is that activities have several modes, and the more a mode uses the non-renewable resource, the shorter the duration of the activity. So to minimise the makespan (makespan) of the project, one of the difficulties of the problem is to choose which activities will be executed with a mode using a lot of non-renewable resources. In our case, the makespan of the project is constrained. We therefore use the best solutions uploaded to PSPLIB to fix our makespan. Finally, to generate the uncertainty intervals we considered that if the duration of an activity i was initially $\hat{p}_i$, the uncertainty interval for this duration is $[0.6\hat{p}_i, \hat{p}_i]$. Thus, even in the worst case scenario where all activities last their maximum duration, we are certain that there exist a feasible solution.
For each instance, we generated 100 scenarios, under each of the following probability distributions:

- Uniform distribution (U): all scenarios are equally possible.

- Neutral Normal distribution (NN): normal distribution with the parameters $\mu = 0.8\hat{p}_i, \sigma^2 = \frac{0.2\hat{p}_i}{3}$.
- Optimistic Normal distribution (ON): normal distribution with the parameters $\mu = 0.7\hat{p}_i, \sigma^2 = \frac{0.1\hat{p}_i}{3}$.
- Pessimistic Normal distribution (PN): normal distribution with the parameters $\mu = 0.9\hat{p}_i, \sigma^2 = \frac{0.1\hat{p}_i}{3}$.

The interest of these distributions is to observe the quality of the trees according to the draws of the scenarios. Thus, if the maximum durations of the activities of an instance are underestimated, the drawing of the scenarios in practice will resemble our pessimistic normal distribution. Conversely, if the estimate of the maximum duration of the activities is overestimated, the actual scenarios will resemble those generated according to the optimistic normal distribution.

7.2 Reactive Algorithm

In order to evaluate the performance of the decision tree approach, it seemed relevant to compare the various evaluation criteria with other methods. Unfortunately, we are not aware of other approaches that use information in the same way as we do. We therefore decided to implement a relatively simple reactive algorithm, which would use information in the same way. Since we want to evaluate the impact of information selection, this algorithm is parameterised by $r_{info} \in \{0, 1\}$ which defines the proportion of information used at each decision time. It can be seen that by choosing $r_{info} = 0$, the reactive algorithm simply follows the initial robust schedule. The pseudo code of the algorithm is shown in Algorithm 3.

Basically given an instance and a scenario, the algorithm simulates the scheduling process according to the scenario. At each decision time, $r_{info} * 100\%$ information are randomly, or blindly, selected from the information available at that time. The scenario "answers" the corresponding questions, and then computes a new robust solution within a limited time, until the next decision moment or the end of the schedule. In this section, the notation R_p denotes the information-based reactive algorithm with the parameter $r_{info} = p$.

7.3 Implementation details

In what precedes, we often refer to solutions procedure for the MMRCPSP with the resource investment objective. This is the case for the RobustMMRCPSP procedure that solves the MMRCPSP with each activity duration is set to its maximal value and for the 1SubsetSolution that solves the MMRCPSP with additional modes for activities that can be included in the information set K_+. For these two procedures, we use the C++ API of IBM ILOG CP Optimizer v12.9, a constraint programming solver tailored for scheduling problem. We use standard CP Optimizer global scheduling constraint. A description of CP

Algorithm 3 Information-Based Reactive Algorithm

Require: an instance I, Ω a set of scenarios, ω a hidden scenario from Ω and $r_{info} \in \{0, 1\}$ the proportion of information selected at each moment.
$\mathcal{K} \leftarrow \{\}$
$s_{init} \leftarrow$ RobustMMRCPSP(I_Ω)
for $0 \leq q < |T|$ **do**
  $K \leftarrow$ set of information available at time t_q
  $K_+ \leftarrow$ set of information where $100r_{info}\%$ of the information from K were randomly selected
  $\Omega_q \leftarrow$ the largest subset of Ω_{K_+} such that $\omega \in \Omega_q$
  solve : RobustMMRCPSP(I_{Ω_q})
  if No solution **then**
    $s_q \leftarrow s_{q-1}$ or s_{init} if $q = 0$
  else
    $s_q \leftarrow$ the solution found by the solver
  end if
end for
return $s_{|T|-1}$

Optimizer model for the MMRCPSP with the resource investment objective can be found in [12, 8]. The remaining algorithms (RobustPartition, RobustDecisionTree and InformationBasedReactiveAlgorithm are coded in C++.

All experiments were run on a Intel Xeon CPU E5-2695 v4 2.10GHz cores running Linux Ubuntu 16.04.4.

7.4 Trees Computations Parameters Impact

As stated in section 6 the computation of the tree takes several parameters as input, in addition to the MMRCPSP instance, and these parameters have a great impact on the size of the tree . Given the number of instances tested, we arbitrarily set $L = 3$ and $|T|= 4$ for the experimentation concerning the sets J_{10} and J_{30}, so the trees can be computed within a reasonable amount of time. The total maximum duration of the project $C_{\max}$ is divided into 5 equal parts, which makes the list of moment of decision $T = [0.2C_{\max}, 0.4C_{\max}, 0.6C_{\max}, 0.8C_{\max}]$. Concerning the parameter Q, the maximum number of questions asked at each node of the tree, we noticed that with $Q > 3$ (resp. $Q > 5$) most of the information were actually used for the instances and the solutions returned by the trees no longer improve, for the set J_{10} (resp. J_{30}). These results are displayed in Figure 11. While this might be surprising, since the difference in the parameter C in the generations of the sets induces that on average more information should be available at a given time for the J_{10} instances, this amount of information is counterbalanced by the difference in the number of activities between the two sets. In order to make the information selection quality easier

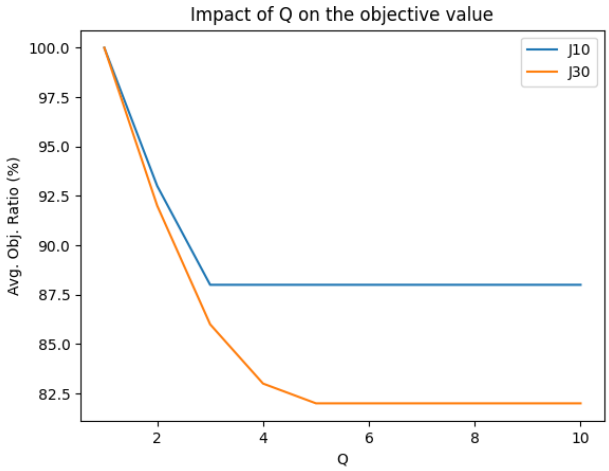

Figure 11: Normalized average objective value as a function of Q.

to observe, the parameter Q is set to 2 (resp. 4) for all the other experimentation involving the set J_{10} (resp. J_{30}).

As for the robust reactive algorithms, they were run with r_{info} in $\{0, 0.5, 1\}$, on the same instances as the decision trees, with the same scenarios, the same decision moments T, and with a computation time at each moment of one minute, and with the current schedule as starting point for the solver. We will denote by R_0, R_{50}, and R_{100} each of these algorithms. One could note that since R_0 do not use any information, the solution returned by this algorithm is the initial robust solution.

7.5 Experimental Results

First we study the impact of the distributions followed by the scenarios on the quality of the solutions proposed by the robust decision trees. Figure 12 shows the relative gap of the average objective value (over all generated scenarios) between the uniform distribution and all the normal distributions for both sets J_{10} and J_{30}. We can first note that in all cases the boxes are entirely below 0, which means that for both sets, the trees obtain better results on 75% of the instances when the scenarios follow one of the normal laws than when they follow the uniform law. We can also notice that for both sets, the average relative gap respects the order $ON < NN < PN$, which is not surprising, since our partitioning criterion favours optimism in the information selection.

Second we look at the amount of information used by each algorithm. The results are shown in Figure 13. Firstly, it can be observed that the trees use

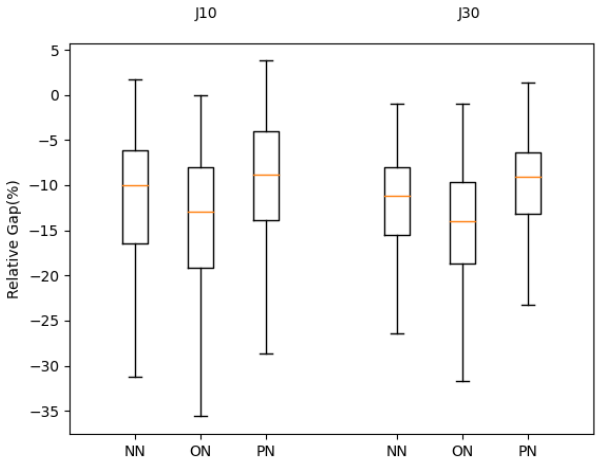

Figure 12: Normalized relative gap of average objective value between the uniform distribution and all the others distributions.

much less information than R_{100} for both instances, and much more than R_0 (which by definition uses none). We can also notice that globally the number of used/available information is higher for the instances of the set J_{30} than for the instances of the set J_{10}. This confirms the intuition of the study of the impact of the parameter Q in the previous section. Then, if we look at the difference of the average of the total number of information used by the trees compared to R_{50}, we see that for the J_{10} instances, the trees use less information on more than 75% of the instances. On the other hand, for the instances of the set J_{30}, this number is quite similar.
In view of the analysis of the quality of the solutions, we can conclude that the trees use the information better than the reactive algorithms, manage to do better (on average) even with less information.

Now that we know that the trees tend to use less information than the R_{50} algorithm, we can asses the quality of the solution offered by the robust decision trees compared to the ones yielded by the reactive algorithms. Figure 14 shows the relative gap of the average (over all the scenarios for a given instance) objective value of the solution returned by the trees and each reactive algorithm between each instances for both sets. In this case, a positive value means that the tree beats the corresponding reactive algorithm, while negative value means the opposite. First of all, it can be noted that in all cases, the reactive algorithm using 0 information never beats the robust decision tree on the same instance. The opposite thing can be noticed concerning the reactive algorithms using all available information: only in best cases the robust decision trees match with the reactive algorithm. However one can observe that algorithm R_{100} only beats

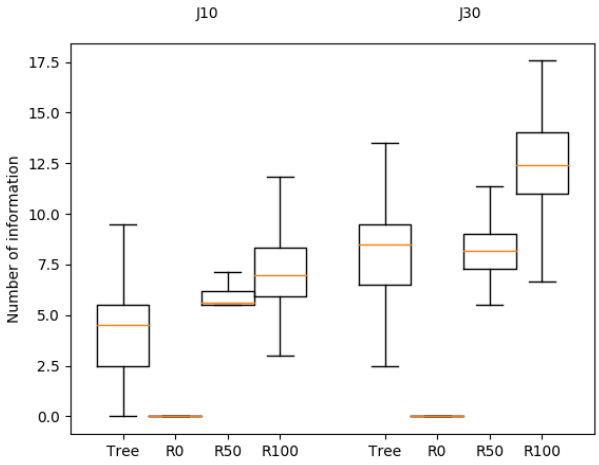

Figure 13: Average total number of information used by trees and reactive algorithms.

the robust decision by only a few percents for more than 75% of the instances, and that the trees beat R_{50} for more than 75% of the instance.

Finally, we study the number of solutions returned by the different algorithms. From a practical point of view, it is important for the operators that the schedules vary as little as possible between two repetitions of the problem. Two solutions are considered to be different if the order induced by the start dates of each activity in each solution is different. Figure 15 displays the average number of different solution found by each algorithm for each instance over all scenarios. Obviously, since algorithm R_0 do not adapt to the scenario, it returns the initial robust solution. However, for both sets the robust decision trees return around 10 different solution per instance, whereas R_{50} and R_{100} return around 30 different solutions for instances from J_{10}, and around 60 solutions for instances from J_{30}. Thus, from the point of view of the number of solutions, trees are preferable to reactive algorithms.

7.6 Results on Industrial Instances

In this section, we present experimental results on two industrial instances provided by Airbus [6]. These instances are denoted by instance A and instance B, with respectively 721 and 486 activities. These instances were not given with interval uncertainties or scenarios, so we generated them the same way as explained in section 7.1. For both of these instances, we generated robust decision trees with the following parameters:

- Number of moment of decision $T = 10$.

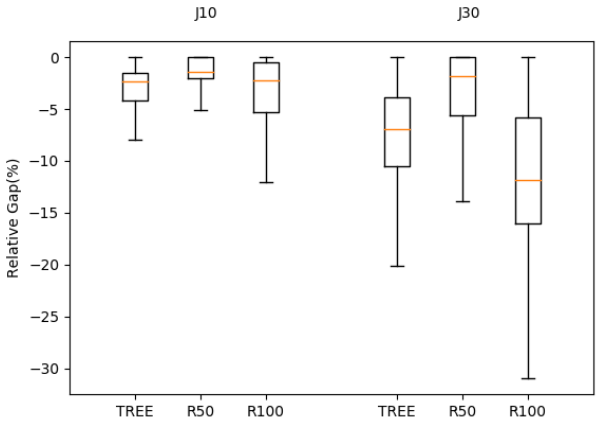

Figure 14: Relative gap (in percent) of average objective value between the solutions returned by the trees, R_{50} R_{100} and the initial robust solution returned by R_0.

- Maximum number of information at each moment $Q \in [1, .., 10]$.
- Maximum size of partitions at each moment $L \in [2, .., 4]$.

Concerning the reactive algorithms, we made r_{info} vary in $[0, 0.1, ..., 1]$. The results are displayed in Figure 16 and 17. To be more precise, each point of these figures is the value (average objective value, average number of information per decision) for a given scenario distribution and a parameterisation for a tree (with a pair Q, L) or a robust reactive algorithm (with r_{info}). Among all the points generated, we kept the one shaping the Pareto front. In this case, since we want to minimise both the average objective value and the average number of information, the "best" points are the ones in the bottom left of each figure. There are several things to note about these figures. First, the impact of the scenario distributions seems to be the same as that noted in Section 7.5. Secondly, we can observe that there are few points forming Pareto fronts, especially for instance A. A first reason is that there are many overlapping points: as we saw in section 7.4, increasing the parameter Q does not improve the quality of the solutions from a certain point. Moreover, given the size of the instances and the fact that the algorithms were run in a limited time (60 sec per decision moment), it sometimes happens that no solution is computed within the given time, leaving the node "empty" in the sense that no question is asked, and the previous robust solution is automatically selected for further tree construction. Nevertheless, it can be seen from the figures that the points of the decision trees are generally a little further to the left than those of the reactive algorithms on the same distribution, with the exception of the instance B, when the tree uses little information.

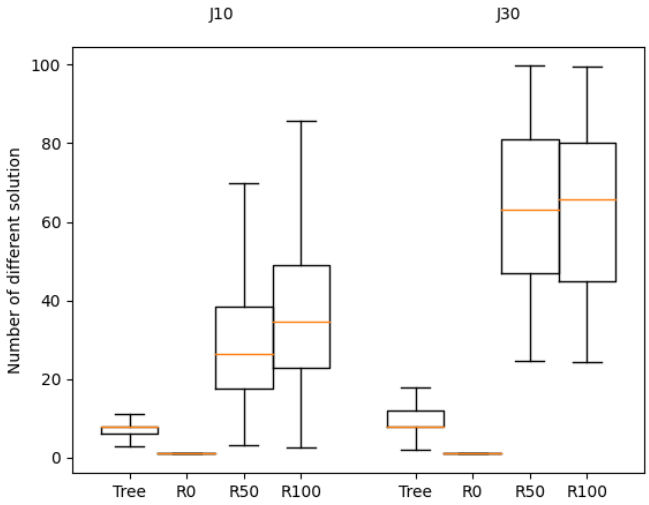

Figure 15: Number of different solutions over 100 scenarios for each instance for both sets.

8 Conclusion

In this work, we have presented a framework based on decision trees that in practice breaks the conservatism inherent in the usual robust approaches.
We show how to compute such trees in the particular case of the scheduling problem for aeronautical assembly lines, modelled as an MMRSPSP. We then evaluated the robust decision trees first on benchmark instances, then on real industrial instances provided by Airbus Madrid by varying the tree construction parameters and the distributions according to which the scenarios were drawn. The results of these experiments showed satisfactory results on the benchmark instances, and promising results on the industrial instances.
One major flaw of our approach is that such trees take a very long time to compute (even if it can be controlled by parameters). Nevertheless we believe that the computation time can be significantly improved for instance, by merging equivalent nodes of the tree as they are being computed. In addition, when the tree is computed, one could imagine using Knowledge Compilation techniques to reduce its size.
It can be noted that the framework proposed in this paper can be extended to any problem as long as it includes a temporal component. However, not all objective functions may be suitable. Objectives whose value can be fixed "early" in the solution (for example an L_{max} objective for scheduling problems) will be more difficult to improve as time progresses. On the contrary, for objective functions based on sums and whose value varies with time (most of the classical objective functions for lot-sizing problems for instance) the use of robust

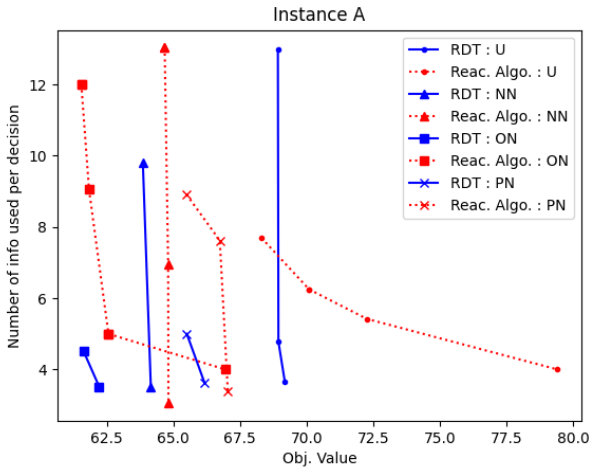

Figure 16: Experimental results on instance A

decision trees is quite appropriate. Moreover, one can imagine several ways of adapting this framework to the problem one wants.
For instance, we have chosen to start with a very -and too much- conservative solution and improve it as we go along. However, we can imagine building the tree the other way round: starting with a super-optimistic solution, and trying to degrade it as little as possible by choosing the best information as the scenario unfolds. Intuitively, one might think that this approach would be more appropriate in cases where optimistic scenarios are very likely.
In addition, the discussions with Airbus have constrained us with regard to access to information. One could extend the approach by assuming that it is possible to have information about the future, not just about the piece of the scenario that has already happened. This would make sense for scheduling problems with uncertainty about the due date of activities for example.

Acknowledgements

This work has been partially funded by the French National Research Agency (ANR) project ANR-18-CE10-0007 “PER4MANCE”.
Crhistian Artigues and Romain Guillaume have benefitted from the AI Interdisciplinary Institute ANITI funding. ANITI is funded by the French ”Investing for the Future – PIA3” program under the Grant agreement n°ANR-19-PI3A-0004.

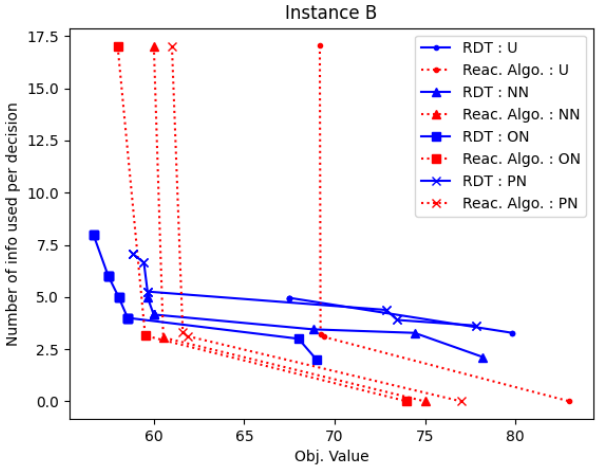

Figure 17: Experimental results on instance B

References

[1] Carlos Andres, Cristobal Miralles, and Rafael Pastor. "Balancing and scheduling tasks in assembly lines with sequence-dependent setup times". In: *European Journal of Operational Research* 187.3 (2008), pp. 1212–1223.

[2] Aharon Ben-Tal, Alexander Goryashko, Elana Guslitzer, and Arkadi Nemirovski. "Adjustable robust solutions of uncertain linear programs". In: *Mathematical programming* 99.2 (2004), pp. 351–376.

[3] Dimitris Bertsimas, David B Brown, and Constantine Caramanis. "Theory and applications of robust optimization". In: *SIAM review* 53.3 (2011), pp. 464–501.

[4] Dimitris Bertsimas and Constantine Caramanis. "Finite adaptability in multistage linear optimization". In: *IEEE Transactions on Automatic Control* 55.12 (2010), pp. 2751–2766.

[5] Dimitris Bertsimas and Iain Dunning. "Multistage robust mixed-integer optimization with adaptive partitions". In: *Operations Research* 64.4 (2016), pp. 980–998.

[6] Tamara Borreguero, Fernando Mas, Jose Luis Menéndez, and MA Barreda. "Enhanced assembly line balancing and scheduling methodology for the aeronautical industry". In: *Procedia engineering* 132 (2015), pp. 990–997.

[7] Tamara Borreguero, Fernando Mas, Jose Luis Menéndez, and MA Barreda. "Enhanced assembly line balancing and scheduling methodology for the aeronautical industry". In: *Procedia engineering* 132 (2015), pp. 990–997.

[8] Tamara Borreguero, Tom Portoleau, Alvaro García Sánchez, Miguel Ortega Mier, and Pierre Lopez. *Exact and heuristic methods for an aeronautical assembly line time-constrained scheduling problem with multiple modes and a resource leveling objective*. Tech. rep. Working paper – hal-03344445. URL: `https://hal.laas.fr/hal-03344445/document`.

[9] Morteza Davari and Erik Demeulemeester. "Important classes of reactions for the proactive and reactive resource-constrained project scheduling problem". In: *Annals of Operations Research* 274.1-2 (2019), pp. 187–210.

[10] Morteza Davari and Erik Demeulemeester. "The proactive and reactive resource-constrained project scheduling problem". In: *Journal of Scheduling* (2017), pp. 1–27.

[11] Mark Drummond, John Bresina, and Keith Swanson. "Just-in-case scheduling". In: *AAAI*. Vol. 94. 1994, pp. 1098–1104.

[12] Patrick Gerhards. "The multi-mode resource investment problem: a benchmark library and a computational study of lower and upper bounds". In: *OR Spectrum* 42.4 (2020), pp. 901–933.

[13] Grani A Hanasusanto, Daniel Kuhn, and Wolfram Wiesemann. "K-adaptability in two-stage robust binary programming". In: *Operations Research* 63.4 (2015), pp. 877–891.

[14] James E Kelley Jr and Morgan R Walker. "Critical-path planning and scheduling". In: *Papers presented at the December 1-3, 1959, eastern joint IRE-AIEE-ACM computer conference*. 1959, pp. 160–173.

[15] Rainer Kolisch and Andreas Drexl. "Local search for nonpreemptive multi-mode resource-constrained project scheduling". In: *IIE transactions* 29.11 (1997), pp. 987–999.

[16] Rainer Kolisch and Arno Sprecher. "PSPLIB-a project scheduling problem library". In: *European journal of operational research* 96.1 (1997), pp. 205–216.

[17] Rainer Kolisch, Arno Sprecher, and Andreas Drexl. "Characterization and generation of a general class of resource-constrained project scheduling problems". In: *Management science* 41.10 (1995), pp. 1693–1703.

[18] Panos Kouvelis and Yu Gang. *Robust Discrete Optimization and Its Applications*. Kluwer Academic Publishers, 1997.

[19] Murari Mani, Ashish K Sing, and Michael Orshansky. "Joint design-time and post-silicon minimization of parametric yield loss using adjustable robust optimization". In: *Proceedings of the 2006 IEEE/ACM international conference on Computer-aided design*. 2006, pp. 19–26.

[20] Supakom Mudchanatongsuk, Fernando Ordóñez, and Jie Liu. "Robust solutions for network design under transportation cost and demand uncertainty". In: *Journal of the Operational Research Society* 59.5 (2008), pp. 652–662.

[21] Yury Nikulin. *Robustness in combinatorial optimization and scheduling theory: An extended annotated bibliography*. Tech. rep. Manuskripte aus den Instituten für Betriebswirtschaftslehre der Universität Kiel, 2006.

[22] Tom Portoleau, Christian Artigues, and Romain Guillaume. "Robust Predictive-Reactive Scheduling: an Information-Based Decision Tree Model". In: *International Conference on Information Processing and Management of Uncertainty in Knowledge-Based Systems*. Springer. 2020, pp. 479–492.

[23] *PSPLIB PROJECT SCHEDULING PROBLEM LIBRARY*. 2020. URL: `http://www.om-db.wi.tum.de/psplib/main.html` (visited on 04/20/2020).

[24] İhsan Sabuncuoğlu, Yasin Gocgun, and Erdal Erel. "Backtracking and exchange of information: Methods to enhance a beam search algorithm for assembly line scheduling". In: *European Journal of Operational Research* 186.3 (2008), pp. 915–930.

[25] Kamran Shahanaghi, Abdolmajid Yolmeh, and Unes Bahalke. "Scheduling and balancing assembly lines with the task deterioration effect". In: *Proceedings of the Institution of Mechanical Engineers, Part B: Journal of Engineering Manufacture* 224.7 (2010), pp. 1145–1153.

[26] Panneerselvam Sivasankaran and Peer Mohamed Shahabudeen. "Literature review of assembly line balancing problems". In: *The International Journal of Advanced Manufacturing Technology* 73.9-12 (2014), pp. 1665–1694.

[27] Stijn Van de Vonder, Erik Demeulemeester, and Willy Herroelen. "A classification of predictive-reactive project scheduling procedures". In: *Journal of scheduling* 10.3 (2007), pp. 195–207.

[28] Phebe Vayanos, Daniel Kuhn, and Berç Rustem. "Decision rules for information discovery in multi-stage stochastic programming". In: *Proceedings of the IEEE Conference on Decision and Control* (Dec. 2011), pp. 7368–7373. DOI: `10.1109/CDC.2011.6161382`.

[29] Yuanhui Zhang, Peter B Luh, Kiyoshi Yoneda, Toshiyuki Kano, and Yuji Kyoya. "Mixed-model assembly line scheduling using the Lagrangian relaxation technique". In: *Iie Transactions* 32.2 (2000), pp. 125–134.