Decision trees for robust scheduling

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

[1] LAAS-CNRS, Universite de Toulouse, CNRS, Toulouse, France
tom.portoleau@laas.fr,artigues@laas.fr
[2] IRIT, Toulouse, France
romain.guillaume@irit.fr

Keywords: proactive-reactive scheduling, robust optimization, decision tree

1 Introduction

In the scheduling literature, approaches that mix a proactive phase aiming at issuing a robust baseline schedule and a reactive phase that adapts the baseline scheduling in case of major disturbances have been widely studied under the proactive-reactive scheduling terminology [3]. Recently Davari and Demeulemeester [2] remarked that in this series of approaches, the proactive and the reactive phases were rather treated separately while they should mutually influence each other. In particular the author propose to associate a reaction cost to the reactive scheduling part. As a matter of fact, over-reacting to disturbance, although generally beneficial for the objective function may lead to a destabilization of the scheduling environment. In this paper, we aim also at addressing this issue together with another one arising with the huge increasing amount of available data. We consider that we have a scheduling problem where a part of parameters is known with precision while another part is known under interval-based scenarios. A robust schedule, e.g. minimizing the min max objective function over the scenario set, can be computed and then followed in run time. During the schedule execution, information becomes regularly available that allows to reduce the uncertainty set and possibly react to this change. However the question arise whether the decision maker should use of not the information, or which part of the information should be used and for which reaction. As already mentioned, over-reacting can be costly but there may be also a cost of getting information, especially when the amount of information that can be obtained is large. We proposed an approach for dealing with the information selection and reaction decision issues, inspired by contingent planning approaches that determines alternative schedules at some events where a high probability of schedule break occur [1]. However we assume that contingent schedules can only be computed at some predetermine decision points, corresponding to particular times such as the end of a work day or a shift change. At each such time point, we select a subset of the information that can be available to partition the set of scenarios. For each element of the partition, we compute a new robust schedule compatible to the one followed up to the decision point. This yields what we call a robust decision tree, that prescribes the path to follow in response to a particular scenario. In the remaining of the paper we present the different models and algorithms we propose as well as experimental results. To illustrate our approach we focus on the simple robust $1||L_{\max}$ model with uncertainty on due dates and knwon processing times.

2 Uncertainty, Information and Robust Decision Tree Models

2.1 Uncertainty model

In practice, it is often easier for a decision-maker to establish bounds over uncertain parameters, like processing times or due dates, than to build an accurate probabilistic model

which often requires a large amount of data. In this paper, we consider interval uncertainty. Given an uncertain parameter x we denote by $X = [x_{min}, x_{max}]$ the interval in which it takes its value. We make no assumption about which probabilistic law is followed by x in this interval. Given a set of uncertain parameters $(x_i)_{i\in I}$ we define the set of possible assignments of parameters by $\Omega = \prod_{i\in I} X_i$ (i.e. it is assumed that there is no correlation between them, all realisation x_i are independent). We call a scenario an assignment of each parameter $x_i, \forall i \in I$ such that $(x_i)_{i\in I} \in \Omega$.
From now on, when ω is a scenario, s a schedule and f the objective , we denote by $f(\omega, s)$ the objective value of s in scenario ω (note that for our application case, $f = L_{max}$).

Example 1 *We consider a small instance of the problem $1||L_{max}$ with three tasks : task 1 : $p_1 = 10$, $d_1 \in D_1 = [10, 11]$, task 2 : $p_2 = 6$, $d_2 \in D_2 = [11, 17]$, task 3 : $p_3 = 4$, $d_3 \in D_3 = [13, 20]$. The set of scenarios for this instance is $\Omega = D_1 \times D_2 \times D_3$ and $\omega = (10, 12, 19)$ is a scenario. We will keep this instance all along this paper to exemplify the notions and algorithms we introduce.*

2.2 Information Model

As explained in Section **??** we suppose that at some time during the execution of the schedule, some information become accessible. In our model, an information allows the scheduler to tighten the interval of uncertainty of a future realisation.

Definition 1 *For a given uncertain parameter $x \in X$ and a moment of decision t during the execution of the schedule, we call an information about x a value k_x^t and an operator in $\{\leq, \geq\}$.*

For instance, an information is $x \leq k_x^t$. So the decision maker, from time t on, is able to reduce the set of possible assignments by updating the interval X, $x \in X_{k_x^t}^{inf} = [x_{min}, k_x^t]$ or $x \in X_{k_x^t}^{sup} =]k_x^t, x_{max}]$. Note that the information depends on t, so the scheduler may have to ask for an information about the same data several times during the execution of the planning. Our model aims to make the best use of available information, and select the more relevant ones.
For the problem considered in this paper we suppose that for a given moment of decision t and a task i, we have an information k_d^t if $\min(t + p_i, 2t) \in D_i$. If so, $k_d^t = \min(t + p_i, 2t)$. Otherwise we consider that we have no information about the task i . This hypothesis on the availability of information is arbitrary, but in fact it expresses two natural questions a scheduler may ask about uncertain due dates : "If task i is started now, can it be completed without being late ? If so, is the due date d_i far from now ?" In any case, an answer to these questions allows the scheduler to bound the uncertainty of a due date d_i.

Example 2 *Let us look again at the instance from Example 1. We suppose that we are at a moment of decision $t = 10$ and that task 1 has been scheduled first. Given the hypothesis we made about information availibility, we have:*

- *task 1 is completed, so there is no relevant information about it.*
- $\min(t + p_2, 2t) = 16 \in D_2$, *so* $k_{d_2}^t = 16$.
- $\min(t + p_3, 2t) = 14 \in D_3$, *so* $k_{d_3}^t = 14$.

Therefore, for any $\omega \in \Omega$, the scheduler is able to determine, from time $t = 10$, if $\omega_2 \leq 16$ or if $\omega_2 > 16$, and if $\omega_3 \leq 14$ or if $\omega_3 > 14$.

2.3 Robust Decision Tree

Definition 2 *A robust decision tree T is a tree where the nodes are labeled with a subset of Ω and a partition of this subset, and the arcs are labeled with a partial schedule. If n is a node of T, Ω^n and P^n denote respectively a subset of Ω and a partition of Ω^n, both associated to n. A robust decision tree satisfies the following properties:*

(i) *The node n has exactly $|P^n|$ children.*
(ii) *Let us denote by $(n_j)_{j \leq |P^n|}$ the children of node n. Each n_j is associated with one element of P^n, i.e for all $j \leq |P_n|$, $\Omega^{n_j} \in P^n$ and $\bigcup_{j \leq |P_n|} \Omega^{n_j} = \Omega^n$.*
(iii) *For any path $(n_0, ..., n_m)$ where n_0 is the root of T and n_m is a leaf, the schedule obtained by concatenating all the partial schedules on the arcs along the path is feasible.*
(iv) *Let n and n' be two nodes of T. The partial schedule s' on the arc (n, n') is robust:*

$$s' = \arg\min_{s \in S} \max_{\omega \in \Omega^{n'}} f(\omega, s)$$

where S is the set of admissible partial solutions.

Example 3 *Based on the instance from Example 1, Figure 1 is an illustration of a Robust Decision Tree. In this case, at the first node after the root, Ω is splitted in Ω_1 and Ω_2. Finally, the tree may lead to two distinct solutions according to the ongoing scenario ω : the sequence of task $[1, 2, 3]$ if $\omega \in \Omega_1$, or $[1, 3, 2]$ if $\omega \in \Omega_2$.*

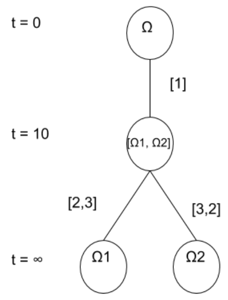

Fig. 1. Illustration of small robust decision tree, for Example 3.

2.4 Partitioning the Scenarios

The core problem of our method is, for a given node n, computing a robust partition P^n, but how do one compare the robustness of two different partitions ? We propose the following criterion. We define the Robustness Score (RS) of a partition P as the sorted vector of the worst case objective values of the optimal robust solution (considering absolute robustness) on each element of P :

$$RS_f(P) = (\min_{s \in S} \max_{\omega \in p} f(\omega, s))_{p \in P}$$

where S is the set of feasible solutions.

As we supposed that we have no information about data distribution, it would be inconsistent to look at the "size" of the element of P. Now, given two partitions P and

P', we say that P is a better partition than P' if $RS(P) <_{lexi} RS(P')$ where $<_{lexi}$ is the lexicographical order. We call Robust Partition Problem (RPP) the problem of finding the best partition with that criterion.

3 Algorithms

Using the previous definitions we now propose a method to build a robust decision tree (see Definition 2). In this paper, we consider that the moments of decision (i.e. moments when the scheduler is able to access new information and change the schedule), denoted by $(t_j)_{j \in J}$ are fixed in advance. This may correspond in practice to special times, such as the end of a working day, or a shift change where the planning can be updated. Every decision moment corresponds to a level in the decision tree, such that t_1 corresponds to the first level, t_2 to the second one, etc... In that respect, the depth of the tree is controlled by the number of decision moments. At each fork at a level j in the tree, a new partial solution, consistent with the partial schedule that has been accomplished until t_j, is proposed according to the current set of scenarios. The root of the tree, that we consider being the level 0 corresponds to the time $t_0 = 0$, the beginning of the schedule. At this point no information is known, so only one robust solution is proposed. Thus, a single node is created at level 1. At this node, we retrieve all the information available at time t_1. Using up to K information, we split the set of scenarios into -at most 2^k- subsets forming a partition P. and obtain a robust partition P'. For each subsets in P' a new solution is proposed and a new branch is set up, leading to a new node at the next level. The set of scenarios considered in this node is the one from which it originated in P'. These steps are repeated until the last decision moment is reached. to solve the the RPP we propose an exhaustive algorithm, that enumerate each combination of information and, provided that the problem admits a global worst case scenario, extracts a dominant partition from this combination. This algorithm has an exponential complexity since that at each iteration of the for loop, we compute a partition P such that $|P| = 2^{|K|}$.

4 Results

The objective of the carried out experiments is to evaluate the robustness of our robust decision tree model, the quality of the selected information used for its construction and its stability in terms of number of reactions for a simulation over 500 scenarios. As our approach uses the notion of information to reduce uncertainties to provide more robust solution, we compare it to a more standard proactive-reactive algorithms. We observe that the robust decision trees provide better solutions (for a given number of tasks) when uncertainty intervals are larger. Intuitively, this can be explained by the fact that decision trees are very constrained by the moments of decision while the reactive algorithms is not. So, larger uncertainties allow robust decision trees to acquire more new information than it does with robust reactive algorithms. Overall our approach obtains better solutions using less information and reactions.

References

1. Davari, M., and Demeulemeester, E. 2017. The proactive and reactive resource-constrained project scheduling problem. *Journal of Scheduling* 1–27.
2. Dearden, R.; Meuleau, N.; Ramakrishnan, S.; Smith, D. E.; and Washington, R. 2003. Incremental contingency planning. In *ICAPS-03 Workshop on Planning under Uncertainty.*
3. Van de Vonder, S.; Demeulemeester, E.; and Herroelen, W. 2007. A classification of predictive-reactive project scheduling procedures. *Journal of scheduling* 10(3):195–207.