Robust scheduling with budgeted uncertainty

Marin Bougeret, Artur Alves Pessoa, Michael Poss

Noname manuscript No.
(will be inserted by the editor)

Robust scheduling with budgeted uncertainty

Marin Bougeret · Artur Alves Pessoa · Michael Poss

the date of receipt and acceptance should be inserted later

Abstract In this work we study min max robust scheduling problems assuming that the processing times can take any value in the budgeted uncertainty set introduced by Bertsimas and Sim (2003, 2004). We consider problems on a single machine that minimize the (weighted and unweighted) sum of completion times and problems that minimize the makespan on parallel and unrelated machines. We provide approximation algorithms: constant factor, average non-constant factor, (fully or not) polynomial time approximation schemes. In addition, we prove that the robust version of minimizing the weighted completion time on a single machine is $\mathcal{NP}$-hard in the strong sense.

Keywords approximation algorithms, robust optimization, scheduling

1 Introduction

Scheduling is a very wide topic in combinatorial optimization with applications ranging from production and manufacturing systems to transportation and logistics systems. Stated generally, the objective of scheduling is to allocate optimally scarce resources to activities over time. The practical relevance and the difficulty of solving the general scheduling problem have motivated an intense research activity in a large variety of scheduling environments. Scheduling problems are usually defined in the following way: given a set of n jobs represented by $\mathcal{J}$, a set of m machines represented by $\mathcal{M}$, and processing times represented by the tuple p, we look for a schedule σ of the jobs on the machines

Production Engineering Department, Fluminense Federal University, Rua Passo da Pátria 156, 24210-240 Niteroí, RJ, Brazil
artur@producao.uff.br
· UMR CNRS 5506 LIRMM, Université de Montpellier, 161 rue Ada, 34392 Montpellier Cedex 5, France
{marin.bougeret,michael.poss}@lirmm.fr

that satisfies the side constraints, represented by the set S of feasible schedules, and minimize objective function $f(\sigma, p)$. Formally, this amounts to solve optimization problem $\min_{\sigma \in S} f(\sigma, p)$. These problems has been intensively studied in the past decades, see [6,18] among many others.

Various sources of uncertainty affect real scheduling problems, among which machine breakdowns, working environment changes, worker performance instabilities, tool quality variations and unavailability. Ignoring these uncertainties usually yields schedules that perform poorly under real conditions. Hence, researchers have introduced frameworks where the uncertainty is directly taken into account either by considering random variables as input or in a worst-case approach where the uncertainty parameters are constrained in a set. These frameworks are respectively denoted by Stochastic Programming and Robust Optimization (RO). We disregard the former in this paper because of its requirement for a probabilistic distribution of the random inputs, which is very difficult to obtain in practice. We focus instead on Robust Scheduling, which models the uncertainty on the processing times by a finite set $U \subset \mathbb{N}^n$. [1] In the robust problem, the maximum value of $f(\sigma, p)$ over all $p \in U$ should be minimized. Formally, this amounts to solve optimization problem $\min_{\sigma \in S} \max_{p \in U} f(\sigma, p)$, or equivalently, $\min_{\sigma \in S} F(\sigma, U)$ where $F(\sigma, U) = \max_{p \in U} f(\sigma, p)$ represents the robust objective function. We say that a schedule $\sigma^* \in S$ is robust if it solves the associated scheduling problem $\min_{\sigma \in S} F(\sigma, U)$.

Robust schedules are desirable from a practical perspective because they hedge against adverse conditions of the system. In spite of its practical relevance, robust scheduling has hardly become a practical tool since differents papers [2,7,27] have shown that very simple scheduling problems become $\mathcal{NP}$-hard as soon as U contains more than one scenario. This is the case, for instance, for minimizing the sum of completion times on a single machine, which is polynomially solvable in the deterministic context but does not admit a Polynomial Time Approximation Scheme (PTAS) in the robust context, unless $\mathcal{P} = \mathcal{NP}$ (see [19]). These negative results are not surprising since [16] had proved that robust combinatorial optimization problems are, more often than not, harder than their deterministic counterpart, even if U contains only two scenarios. In view of these negative results, researchers willing to provide solutions to robust scheduling address them with heuristics or linear programming approaches, see for instance [20,15], rather than using purely combinatorial algorithms.

In this context, the positive results of Bertsimas and Sim [4] opened a new avenue of research in combinatorial robust optimization. Given two positive vectors $\overline{p}$ and $\hat{p}$, that respectively represent the nominal value of and the deviation of p, and a positive integer Γ, they define the following uncertainty

[1] For the sake of clarity, we consider that p is a n-uple. One readily extends the definition to more complex problems, such as those defined on unrelated machines where p is instead a nm-uple.

set:

$$U^{\Gamma} \equiv \left\{ p \in \mathbb{R}^n : p_j = \overline{p}_j + \delta_j \hat{p}_j, j \in \{1,\dots,n\}, \delta_j \in \{0,1\}, j \in \{1,\dots,n\}, \sum_{j=1}^{n} \delta_j \leq \Gamma \right\}.^{2}$$

As the previous set is more structured than an arbitrary uncertainty set U, one could expect better positive results. Bertsimas and Sim proved indeed that, for a large class of combinatorial optimization problems, the robust counterparts defined through U^{Γ} (denoted U^{Γ}-robust for short) belong to the complexity classes of their deterministic versions:

Theorem 1 (Bertsimas and Sim [4]) *Let $\mathcal{X} \subseteq \{0,1\}^n$ and $p \in \mathbb{R}^n$ characterize the combinatorial optimization problem $\min_{x\in\mathcal{X}} \sum_i p_i x_i$. The optimal solution to the robust problem $\min_{x\in\mathcal{X}} \max_{p\in U^{\Gamma}} p^T x$ can be obtained by solving problem $\min_{x\in\mathcal{X}} \sum_i p_i^{\ell} x_i$ for each $\ell = 1,\dots,n+1$ and taking the minimum, where $p_j^{\ell} = \overline{p}_j + \max(\hat{p}_j - \hat{p}_{\ell}, 0)$ for each $\ell = 1,\dots,n+1$ and $\hat{p}_{n+1} = 0$.*

They provided a similar result for the approximation ratio of robust combinatorial optimization problems and [9] have extended these positive results to larger classes of U^{Γ}-robust combinatorial optimization problems, many of them relying on dynamic programming algorithms, see [1,14,24]. The impact of Theorem 1 is such that, prior to this paper, there was no known example of polynomial combinatorial optimization problem having a $\mathcal{NP}$-hard U^{Γ}-robust counterpart; only an example of a weakly $\mathcal{NP}$-hard problem turning strongly $\mathcal{NP}$-hard in the U^{Γ}-robust case has been exhibited [21]. In addition to its theoretical tractability, U^{Γ}-robust problems can often be solved numerically by applying the classical dualization approach to robust optimization [3]. This approach holds because set U^{Γ} can be described as the extreme points of a polytope described by a linear number of inequalities. Hence, there are some papers solving numerically U^{Γ}-robust scheduling problems, e.g. [8]. The interest for set U^{Γ} is also motivated by its link with probabilistic constraints studied in [5,23]. In the context of combinatorial optimization problems with cost uncertainty, the results from [5,23] imply that $\min_{x\in\mathcal{X}} \max_{p\in U^{\Gamma}} p^T x$ provides a conservative approximation of minimizing the value-at-risk over $\mathcal{X}$, see also [24]. In spite of the tremendous success of set U^{Γ} in the robust combinatorial optimization literature, we are not aware of previous work studying the theoretical complexity of U^{Γ}-robust scheduling problems, apart from [26], which was carried out in parallel with the present work. Therein the authors study some one-machine scheduling problems with uncertainty set U^{Γ} and two other variants, however, assuming that there exists a constant $K > 0$ such that $\hat{p}_j = K\overline{p}_j$ for each $j \in \mathcal{J}$.

Recall the three-field notation $\alpha|\beta|\gamma$ from [10] where α describes the machine environment, β the job characteristics, and γ the objective function. In this first work on U^{Γ}-robust scheduling, we focus on the following classical

[2] Notice that [4] originally considers both upwards and downwards deviations. However, the latter are irrelevant for the class of problems studied in this manuscript.

scheduling problems. Let $C_j(\sigma, p)$ denote the completion time of task j for schedule σ and processing times represented by p. The first type of problems studied herein concerns the minimization of the weighted sum of completion times on a single machine ($1||\sum_j w_j C_j$), defined by letting S contain all orders for the n tasks and setting $f(\sigma, p) = \sum_{j\in\mathcal{J}} w_j C_j(\sigma, p)$. We pay a particular attention to the case where $w_j = 1$ for each $j \in \mathcal{J}$, which is denoted $1||\sum_j C_j$. The second type of problems studied herein considers a set of m machines, which can be identical (P), uniform (Q) or unrelated (R), and minimize the makespan $f(\sigma, p) = C_{max}(\sigma, p) = \max_{j\in\mathcal{J}} C(\sigma, p)$. In the first case, processing job j on machine i is given by p_j. In the second case, the processing time is given by $p_{ij} = p_j/s_i$ where s_i is the speed of machine i. In the last case, the processing times are given by an arbitrary matrix $p \in \mathbb{N}^{m\times n}$. The resulting problems are denoted by $P||C_{max}$, $Q||C_{max}$, and $R||C_{max}$, respectively. We also consider the special case $Rm||C_{max}$ where the number of machines m is not considered part of the instance.

Contributions and structure of the paper Let us extend Graham's notation to $\alpha|\beta|U_p^\Gamma|\gamma$ to specify that the cost of any feasible schedule is obtained for the worst processing times in U^Γ. We refer the reader to the next paragraphs which provide two examples of robust scheduling problems. In Section 2 we consider one machine problems minimizing the sum of completion times. We prove that $1||U_p^\Gamma|\sum C_j$ is polynomial by extending Theorem 1, thus complementing the polynomial algorithm provided in [26] for the case where $\hat{p}_j = K\overline{p}_j$ for each $j \in \mathcal{J}$. Comparing with [2,7,27], the result illustrates how U^Γ-robust scheduling can lead to more tractable problems than robust scheduling with arbitrary uncertainty sets. We show then that $1||U_p^\Gamma|\sum w_j C_j$ is weakly $\mathcal{NP}$-hard if $\Gamma = 1$ and strongly $\mathcal{NP}$-hard if $\Gamma > 1$. To our knowledge, this is the first example of a polynomial scheduling problem having a $\mathcal{NP}$-hard U^Γ-robust counterpart. Our hardness proof is inspired on the classical reduction from the 3-partition problem to the decision version of $1||\sum w_j T_j$ [22] where the jobs can be divided into partition jobs, that represent the 3-partition elements, and separation jobs, that have fixed positions in any valid certificate. In the case of $1||U_p^\Gamma|\sum w_j C_j$, to force the separating jobs to have fixed positions, we use a third class of jobs whose processing time deviations impose the same increase to the overall scheduling cost. The remaining deviations are then designed in such a way that a separating job has its processing time deviated in the worst case scenario if and only if it is scheduled before its fixed position, which allows to prove that these fixed positions lead to a strictly minimum overall cost. In Section 3 we show that $P||U_p^\Gamma|C_{max}$ is 3-approximable and admits a PTAS if Γ is constant. Section 4 is dedicated to $R||U_p^\Gamma|C_{max}$. We first show how the classical FPTAS of [12] for $Rm||C_{max}$ can be adapted for $Rm||U_p^\Gamma|C_{max}$. We then focus on the general case and provide an average $O(\log m)$-approximation based on an extended formulation of the problem. The formulation is solved in polynomial time by combining column generation with an approximately feasible solution for the pricing problem. Finally, a

classical randomized rounding is applied, which is carefully analyzed to provide the required approximation factor.

Example for $1||U_p^\Gamma|\sum C_j$ Let us consider the following instance of $1||U_p^\Gamma|\sum C_j$ with $n = 3$ jobs and $\Gamma = 1$. Let $\overline{p}_1 = 3, \overline{p}_2 = 1, \overline{p}_3 = 2, \hat{p}_1 = 1, \hat{p}_2 = 10, \hat{p}_3 = 5$. By definition we have $U_p^\Gamma = \{(3,1,2),(4,1,2),(3,11,2),(3,1,7)\}$. Notice that a schedule is completely characterized by a permutation of the job. For $\sigma = (2,1,3)$, we have $f(\sigma,(3,1,2)) = 11$, $f(\sigma,(4,1,2)) = 13$, $f(\sigma,(3,11,2)) = 41$, $f(\sigma,(3,1,7)) = 16$, and thus $F(\sigma) = 41$.

Example for $P||U_p^\Gamma|C_{max}$ Let us now consider the following instance of $P||U_p^\Gamma|C_{max}$ with $n = 4$ jobs, $m = 2$ machines and $\Gamma = 1$. Let $\overline{p}_1 = 5, \overline{p}_2 = 3, \overline{p}_3 = 2, \overline{p}_4 = 2, \hat{p}_1 = 1, \hat{p}_2 = 2, \hat{p}_3 = 12, \hat{p}_4 = 8$. By definition we have $U_p^\Gamma = \{(5,3,2,2),(6,3,2,2),(5,5,2,2),(5,3,14,2),(5,3,2,10)\}$. Notice that a schedule is completely characterized by the set of jobs scheduled on each machine. For σ that schedules jobs $\{1,2\}$ on machine 1, and jobs $\{3,4\}$ on machine 2, we have $f(\sigma,(5,3,2,2)) = 8$, $f(\sigma,(6,3,2,2)) = 9$, $f(\sigma,(5,5,2,2)) = 10$, $f(\sigma,(5,3,14,2)) = 16$, $f(\sigma,(5,3,2,10))) = 12$, and thus $F(\sigma) = 16$.

Notations used throughout the paper A schedule is denoted by σ and $\sigma_i \subseteq \mathcal{J}$ denotes a schedule restricted to machine i. An optimal schedule is denoted by σ^* and its value is denoted by opt. For any integer n, $[n] = \{0,\ldots,n\}$ and $[n]^* = [n] \setminus \{0\}$.

2 Minimizing sum of completion times

2.1 Unitary weights

This is one of the simplest scheduling problems, yet it is $\mathcal{NP}$-hard in the weak sense for arbitrary uncertainty sets U, even for two scenarios [27]. In contrast, we show below that the U^Γ-robust version of the problem can be solved in polynomial time.

Let x_{ij} be equal to 1 iff job j is scheduled in position i. Problem $1||U_p^\Gamma|\sum C_j$ can be cast as

$$\left\{ \min_x \max_{p\in U^\Gamma} \sum_{(i,j)\in[n]^*\times\mathcal{J}} p_j(n+1-i)x_{ij} : \sum_{i\in[n]^*} x_{ij} = 1, j\in\mathcal{J}, \sum_{j\in\mathcal{J}} x_{ij} = 1, i\in[n]^* \right\}. \tag{1}$$

Observation 1 *Theorem 1 cannot be applied to problem* (1) *because its cost function is defined by a product of parameters* $p_j(n+1-i)$ *where only* $p \in U^\Gamma$. *Hence, the change of* p_i *from the scenario set affects the coefficients for several decision variables.*

We provide below a straightforward extension of Theorem 1, which encompasses problem (1). Its proof, following essentially the lines of the proof of Theorem 1 from [4], is deferred to the appendix.

Proposition 1 *Let* $\mathcal{X} \subseteq \left\{\{0,1\}^{I\times J} : \sum_{i=1}^{I} x_{ij} = 1,\ j = 1,\dots,J\right\}$ *and let* $q \in \mathbb{R}^I$ *and* $p \in U^\Gamma$ *be cost vectors. The optimal solution to problem*

$$\min_{x\in\mathcal{X}} \max_{p\in U^\Gamma} \sum_{i,j} p_j q_i x_{ij} \tag{2}$$

can be obtained by solving the problems $\min_{x\in\mathcal{X}} \sum_{i,j} (\overline{p}_j + \hat{p}_j) q_i x_{ij}$ *and*

$$\Gamma \hat{p}_l q_k + \min_{x\in\mathcal{X}} \sum_{i,j} \left(\overline{p}_j + \tilde{p}_{ij}^{kl}\right) q_i x_{ij},$$

for each $k \in I, l \in J$ *and taking the minimum, where* $\tilde{p}_{ij}^{kl} = \max(0, \hat{p}_j - \frac{\hat{p}_l q_k}{q_i})$.

Applying Proposition 1 to (1) by setting $q_i = n + 1 - i$, we obtain that $1||U_p^\Gamma|\sum C_j$ can be solved by solving $O(n^2)$ assignment problems. We point out that, although the robust problem can be solved in polynomial time, the modified cost coefficients $\overline{p}_j + \tilde{p}_{ij}^{kl}$ break the structure of $1||U_p^\Gamma|\sum C_j$, i.e., the deterministic problems with cost vector $\overline{p} + \tilde{p}^{kl}$ are not instances of $1||\sum C_j$.

2.2 General weights

It is well known that problem $1||\sum w_j C_j$ can be solved in polynomial time by applying Smith's rule [25] (i.e., scheduling jobs by non-decreasing $\frac{p_j}{w_j}$). However, it does not seem easy to extend that simple rule to the robust problem $1||U_p^\Gamma|\sum w_j C_j$. In fact, we show that the problem is $\mathcal{NP}$-hard in the weak sense for $\Gamma = 1$ and strongly $\mathcal{NP}$-hard for arbitrary Γ. For that, we need the following two lemmas characterizing the structure of an optimal solution. First we show that for any pair of consecutive jobs j, ℓ, if $\frac{w_j}{\overline{p}_j} < \frac{w_\ell}{\overline{p}_\ell + \hat{p}_\ell}$, then j must be scheduled after ℓ.

Lemma 1 *Given* $X \subset \mathcal{J}$ *such that* $\frac{w_j}{\overline{p}_j} < \frac{w_\ell}{\overline{p}_\ell + \hat{p}_\ell}$, $\forall j \in X$, *and* $\ell \in \mathcal{J} \setminus X$, *in any optimal solution for* $1||U_p^\Gamma|\sum w_j C_j$ *the jobs in* X *are the last* $|X|$ *in the schedule.*

Proof We prove the proposition by contradiction. Assume that there is an optimal solution σ^* for $1||U_p^\Gamma|\sum w_j C_j$ where two consecutive jobs j and ℓ have

$$\frac{w_j}{\overline{p}_j} < \frac{w_\ell}{\overline{p}_\ell + \hat{p}_\ell}. \tag{3}$$

Let $c^*(p)$ denote the solution cost for the specific vector $p \in U^\Gamma$ and c^* be the solution cost for worst deviations; that is, $c^* = \max_{p\in U^\Gamma} c^*(p)$. By swapping j

and ℓ in σ^*, we obtain an alternative schedule σ' with cost denoted c', which satisfies

$$\begin{aligned} c' &= \max_{p \in U^\Gamma} (c^*(p) + p_\ell w_j - p_j w_\ell) \\ &\leq \max_{p \in U^\Gamma} c^*(p) + \max_{p \in U^\Gamma} (p_\ell w_j - p_j w_\ell) \\ &\leq c^* + (\overline{p}_\ell + \hat{p}_\ell) w_j - \overline{p}_j w_\ell. \end{aligned} \tag{4}$$

This swap operation is illustrated by Figure 1, where rectangles depict jobs, with widths proportional to the corresponding processing times. From (3) and (4), we obtain that $c' < c^*$, which contradicts the optimality of σ^*. □

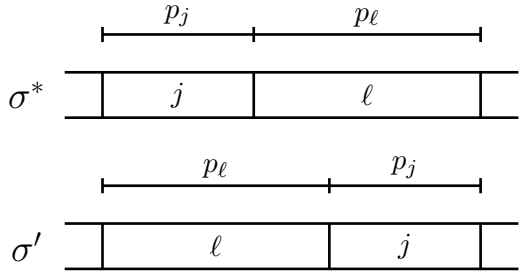

Fig. 1 Swap between jobs j and ℓ applied in Lemma 1.

Now we show that if two jobs only differ by their value of $\hat{p}_j$, then the one having the smallest value must be scheduled first.

Lemma 2 *There exists an optimal solution for* $1||U_p^\Gamma| \sum w_j C_j$ *where, for any two jobs* j *and* ℓ *with* $\overline{p}_j = \overline{p}_\ell$, $w_j = w_\ell$, *and* $\hat{p}_j < \hat{p}_\ell$, j *is scheduled before* ℓ.

Proof Let j and ℓ be two jobs satisfying the conditions of this proposition such that ℓ precedes j in an optimal solution σ^*. We show that swapping ℓ and j does not increase the robust cost c^* of σ^*. Let σ' be the resulting schedule. Let also W_k be the sum of weights of all jobs that do not precede k in σ^*, for all $k \in \mathcal{J}$. Clearly $W_j < W_\ell$. Moreover, the total cost due to mean processing times is the same for σ^* and σ', and the total cost due to deviations is calculated by selecting the Γ jobs with maximum $\hat{p}_k W_k$ among all $k \in \mathcal{J}$, and summing up these values. After the swap, $\hat{p}_j W_j$ and $\hat{p}_\ell W_\ell$ are replaced by $\hat{p}_\ell W_j$ and $\hat{p}_j W_\ell$, and the remaining values are kept unchanged. Since $W_j < W_\ell$ and $\hat{p}_j < \hat{p}_\ell$, we have that $\hat{p}_\ell W_j + \hat{p}_j W_\ell < \hat{p}_j W_j + \hat{p}_\ell W_\ell$. As a result, the total cost due to deviations cannot increase after the swap. □

For the hardness proof, we define the k-PARTITION problem.

Definition 1 *Given* kN *positive numbers* $a_1, \ldots, a_{kN}$ *satisfying* $\sum_{j=1}^{kN} a_j = NA$, *k-PARTITION asks if there exists a partition of* $\{a_1, \ldots, a_{kN}\}$ *into* N *subsets* $S_1, \ldots, S_N$ *such that* $\sum_{j \in S_i} a_j = A$, *for* $i = 1, \ldots, N$.

The decision version of $1||U_p^\Gamma|\sum w_jC_j$, denoted by $(1||U_p^\Gamma|\sum w_jC_j, K)_{dec}$, asks for a schedule whose robust cost is not greater than a given integer K.

Theorem 2 *There is a polynomial reduction from k-PARTITION to* $(1||U_p^\Gamma|\sum w_jC_j, K)_{dec}$ *with* $\Gamma = N-1$.

Proof First, we describe a reduction allowing that $\hat{p}$ is a vector of rational numbers. Later, we show how the proposed reduction can be modified to use only integer numbers, and still satisfy the conditions of this theorem. We create three types of jobs. For $j = 1, \ldots, kN$, job j, referred to as a *partition* job, has $w_j = \overline{p}_j = a_j$, and $\hat{p}_j = 0$; for $j = kN+1, \ldots, (k+1)N-1$, job j, referred to as a *tail* job, has $w_j = 1$, $\overline{p}_j = 2N$, and $\hat{p}_j = \frac{4NA}{(k+1)N-j}$; for $j = (k+1)N, \ldots, (k+2)N-2 = n$, job j, referred to as a *separating* job, has $w_j = 2$, $\overline{p}_j = 1$, and $\hat{p}_j = \frac{4NA}{W_j+\beta(j)A}$, where $\beta(j) = j-(k+1)N+1$, and $W_j = N-1+2\beta(j)$. Moreover, $\Gamma = N-1$.

We restrict our analysis to schedules that satisfy Lemma 1 and 2 since they necessarily include an optimal solution to the optimization version of the problem. Thus, we can conclude that the last $N-1$ scheduled jobs are exactly the tail jobs, which are sorted in an increasing order by their indices, and that the separating jobs are sorted in a decreasing order by their indices. In the following, we construct a schedule σ for the proposed instance which is feasible if and only if the corresponding k-PARTITION instance is feasible. The schedule is illustrated by Figure 2, following the same convention of Figure 1 but representing only the mean processing times. In this figure, partition, tail and separating jobs are colored in white, light gray and dark gray, respectively. For the sake of easy notation, we consider only the special case where the partition jobs are sorted by their indices in the figure.

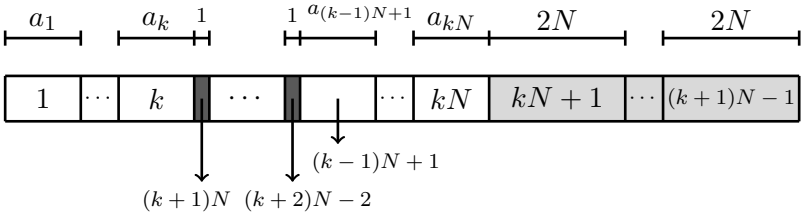

Fig. 2 Schedule σ constructed in Theorem 2 for the special case where partition jobs are sorted by their indices.

For a given schedule σ, let $\sigma(\ell)$ denote the ℓ-th job to be executed, for $\ell = 1, \ldots, kN$, and define $\sigma^{-1}(j)$ such that $\sigma(\sigma^{-1}(j)) = j$ for each $j \in \mathcal{J}$. Define also p^σ as the worst vector $p \in U^\Gamma$ for the schedule σ. In the objective function $\sum_{j\in\mathcal{J}}\sum_{\ell=\sigma^{-1}(j)}^{n} p_j^\sigma w_{\sigma(\ell)}$, the term $p_j^\sigma w_{\sigma(\ell)}$ is referred to as the cost from job j to job $\sigma(\ell)$. Let also $\sigma^\Delta(\ell)$ denote the ℓ-th partition job to be

executed according to σ, and define $(\sigma^{\Delta})^{-1}$ analogously to σ^{-1}. Finally, let

$$A_j^{\sigma} = \sum_{\substack{\ell=1 \\ \sigma^{-1}(\ell) \geq \sigma^{-1}(j)}}^{kN} a_\ell$$

be the sum of the weights of the partitions jobs scheduled after job j (and including the weight of job j if it is a partition job).

We divide the cost of a schedule σ for the created instance of $(1||U_p^{\Gamma} \,|\, \sum w_j C_j, K)_{dec}$ into six terms:

- the cost from partition jobs to partition jobs, given by

$$c_1 = \sum_{j=1}^{kN} \sum_{\ell=(\sigma^{\Delta})^{-1}(j)}^{kN} \overline{p}_j w_{\sigma^{\Delta}(\ell)} = \sum_{j=1}^{kN} \sum_{\ell=1}^{kN} a_j a_\ell;$$

- the cost from partition jobs to tail jobs, given by

$$c_2 = \sum_{j=1}^{kN} \sum_{\ell=kN+1}^{(k+1)N-1} \overline{p}_j w_\ell = NA(N-1);$$

- the cost from tail jobs excluding deviations, given by

$$c_3 = \sum_{j=kN+1}^{(k+1)N-1} \sum_{\ell=j}^{(k+1)N-1} \overline{p}_j w_{\sigma(\ell)} = N^2(N-1);$$

- the cost from partition jobs to separating jobs, given by

$$c_4 = \sum_{j=1}^{kN} \sum_{\substack{\ell=(k+1)N \\ \sigma^{-1}(\ell) > \sigma^{-1}(j)}}^{(k+2)N-2} \overline{p}_j w_\ell = 2NA(N-1) - 2 \sum_{j=(k+1)N}^{(k+2)N-2} A_j^{\sigma};$$

- the cost from separating jobs excluding deviations, given by

$$c_5 = \sum_{j=(k+1)N}^{(k+2)N-2} \sum_{\substack{\ell=1 \\ \sigma^{-1}(\ell) > \sigma^{-1}(j)}}^{(k+2)N-2} \overline{p}_j w_\ell = N(N-1) + (N-1)^2 + \sum_{j=(k+1)N}^{(k+2)N-2} A_j^{\sigma};$$

- the cost due to deviations from both the separating jobs and the tail jobs, given by

$$\begin{aligned} c_6 &= \sum_{j=kN+1}^{(k+2)N-2} \sum_{\substack{\ell=1 \\ \sigma^{-1}(\ell) > \sigma^{-1}(j)}}^{(k+2)N-2} (p_j^{\sigma} - \overline{p}_j) w_\ell \\ &= \sum_{j=(k+1)N}^{(k+2)N-2} \max \left\{ 4NA, \sum_{\substack{\ell=1 \\ \sigma^{-1}(\ell) > \sigma^{-1}(j)}}^{(k+2)N-2} \hat{p}_j w_\ell \right\} \end{aligned}$$

$$= \sum_{j=(k+1)N}^{(k+2)N-2} \max\left\{4NA, \frac{4NA}{W_j + \beta(j)A}(W_j + A_j^\sigma)\right\}$$

where the second equality holds because for each tail job, the cost due to its deviation is equal to $4NA$.

The total cost is given by $c_1 + c_2 + c_3 + c_4 + c_5 + c_6$. Note that only c_6, the third term of c_5 and the second term of c_4 depend on the schedule σ. All remaining terms are constant. Summing up the non-constant terms, we obtain

$$\tilde{c}(\sigma) = \sum_{j=(k+1)N}^{(k+2)N-2} \max\left\{4NA - A_j^\sigma, \frac{4NA}{W_j + \beta(j)A}W_j + \left(\frac{4NA}{W_j + \beta(j)A} - 1\right)A_j^\sigma\right\}.$$

Now, let us consider the previous expression as a function $\tilde{c}(A^\sigma)$ of the vector A^σ whose components A_j^σ are relaxed to non-negative integer numbers. Assuming that $A > 3$, we have that $\frac{4NA}{W_j+\beta(j)A} > 2$. Hence, the value of $\tilde{c}(A^\sigma)$ is minimized (and thus $\tilde{c}(\sigma)$ and the total cost) when $A_j^\sigma = \beta(j)A$, for $j = (k+1)N, \ldots, (k+2)N-2$. This only occurs when each sum of processing times of partition jobs scheduled between two consecutive separating jobs is exactly A. Otherwise, by the coefficients to A_j^σ in the two arguments of the maximum function, $\tilde{c}(\sigma)$ increases by at least one unit. Thus, setting $K = c_1 + c_2 + c_3 + N(N-1) + (N-1)^2 + 5.5NA(N-1) + 0.5$, we have that a positive answer to k-PARTITION yields a schedule of cost $K - 0.5$, and that any schedule costs at least $K + 0.5$ otherwise.

To ensure that the constructed instance contains only integer numbers on the input, we multiply all processing times and K by $2(N-1)\sum_{j\in\mathcal{J}} w_j$. This yields a solution of cost $K - (N-1)\sum_{j\in\mathcal{J}} w_j$ in the case of a positive answer to k-PARTITION, and no solution of cost less than $K + (N-1)\sum_{j\in\mathcal{J}} w_j$ otherwise. By rounding up the values of $\hat{p}_j$, for $j = kN+1, \ldots, (k+2)N-2$, the cost of each solution may increase by at most $(N-1)\sum_{j\in\mathcal{J}} w_j$, still allowing to answer k-PARTITION. Moreover, if A is polynomially bounded for the k-PARTITION, so are all input data for the constructed instance. □

The next corollary proves the desired hardness results.

Corollary 1 *$(1||U_p^\Gamma|\sum w_jC_j, K)_{dec}$ is $\mathcal{NP}$-complete in the weak sense for $\Gamma = 1$ and strongly $\mathcal{NP}$-complete when Γ is part of the input.*

Proof For $N = 2$ and arbitrary k, k-PARTITION corresponds to PARTITION, which is weakly $\mathcal{NP}$-complete, and, for $k = 3$ and arbitrary N, k-PARTITION generalizes 3-PARTITION, which is $\mathcal{NP}$-complete in the strong sense. Hence, the corollary follows directly from the reduction given by Theorem 2. □

3 Minimizing makespan on identical machines

We introduce the following notations. For any set of jobs $X \subseteq \mathcal{J}$, we use $\overline{p}(X) = \sum_{j \in X} \overline{p}_j$ and $\hat{p}(X) = \sum_{j \in X} \hat{p}_j$. We let $\Gamma(X)$ contain the Γ jobs from X with highest deviations, $\hat{p}^{\Gamma}(X) = \hat{p}(\Gamma(X))$, and $\hat{p}^{\Gamma}(\sigma) = \sum_{i \in \mathcal{M}} \hat{p}^{\Gamma}(\sigma_i)$. We also use $C(\mathcal{J}) = \overline{p}(\mathcal{J}) + \hat{p}^{\Gamma}(\mathcal{J})$ and $C(\sigma) = \max_{i \in \mathcal{M}} C(\sigma_i)$. Remark that $F(\sigma, U^{\Gamma}) = C(\sigma)$, which can be computed in polynomial time.

We say that an algorithm A is a ρ-dual approximation if for any ω and instance $\mathcal{I}$, either $A(\mathcal{I}, \omega)$ builds a schedule σ such that $C(\sigma) \leq \rho\omega$ or fails, which implies then that $\omega < \text{opt}$. Notice that any ρ-dual approximation can be converted to a ρ-approximation algorithm by performing a binary search on ω to find the smallest ω that is not rejected.

3.1 General case: 3-approximation

Let us first review some simple approaches that do not work. First, applying a PTAS on a classical instance of $P||C_{max}$ where $p_j = \overline{p}_j + \hat{p}_j$ does not help as the gap between optimal values of the new instance and the original one (for the robust problem) can be large. Defining only $p_j = \overline{p}_j$ (i.e. ignoring deviations) and applying a PTAS would lead to a $2(1+\epsilon)$ ratio if $\hat{p}_j \leq \overline{p}_j$, but does not work in the general case. Finally, it seems also tempting to apply a PTAS on a first instance where $p_j = \overline{p}_j$ to get a schedule σ_1, apply a PTAS on a second instance where $p_j = \hat{p}_j$ to get a schedule σ_2, and try to merge σ_1 and σ_2 to get a $2(1+\epsilon)$ algorithm, but again finding such a merge does not seem straightforward.

Let us now design an algorithm A for $P||U_p^{\Gamma}|C_{max}$, and prove the following theorem.

Theorem 3 *For any $\epsilon > 0$, if for any $j \in \mathcal{J}$, $\overline{p}_j \leq \epsilon\omega$ and $\hat{p}_j \leq \epsilon\omega$, then A is a $\min(2+2\epsilon, 3)$ dual approximation algorithm for $P||U_p^{\Gamma}|C_{max}$. This implies that $P||U_p^{\Gamma}|C_{max}$ admits a 3-approximation in the general case.*

Before presenting algorithm A, we point out an important obstacle faced when designing dual algorithms for the problem. As usual, fixing the value of ω is suitable as it defines the size of bins in which we can schedule the jobs. Thus, a natural way to design a dual approximation algorithm would be to take the jobs in an arbitrary order and schedule as many of them as possible into each machine, moving to the next machine whenever $C(\sigma_i) > \omega$, and rejecting ω if there remain some jobs after filling m machines. If (as in Theorem 3) for any $j \in \mathcal{J}$, $\overline{p}_j \leq \epsilon\omega$ and $\hat{p}_j \leq \epsilon\omega$, this algorithm would not exceed $(1+2\epsilon)\omega$, thus improving over Theorem 3. However, this algorithm is not correct, as the existence of a σ with $C(\sigma_i) > \omega$ for any i does not imply that $\omega < \text{opt}$, even if the algorithm selects jobs by non-increasing $\hat{p}_j$. Indeed, consider the input where $m = \Gamma = 2$, $\omega = 15$ and $p_1 = (0, 10)$, $p_2 = p_3 = p_4 = (0, 6)$, $p_5 = (5, 0)$, $p_6 = (3, 0)$ (where $p_j = (\overline{p}_j, \hat{p}_j)$). The previous algorithm woud create $\sigma_1 = \{1, 2\}$, $\sigma_2 = \{3, 4, 5\}$ and rejects as $C(\sigma_i) > \omega$ for any i and not all

jobs are scheduled, whereas there exists a schedule $\sigma_1^* = \{1, 5\}, \sigma_2^* = \{2, 3, 4, 6\}$ that fits in ω. This explains the design of Algorithm 1. The validity of the algorithm is shown in the rest of this section.

Algorithm 1 Algorithm A

```
// Given a set of jobs J, A(J) either schedules J on m machines, or fails.
i = 1
while J ≠ ∅ AND i ≤ m do
   σ_i ← ∅
   while J ≠ ∅ AND p̄(σ_i) ≤ ω AND p̂^Γ(σ_i) ≤ ω do
      assign to σ_i the largest job (in term of p̂_j) of J;
   end while
   i ← i + 1;
end while
if J ≠ ∅ then
   fails
end if
```

Observation 2 *For any σ, $C(\sigma) \leq \omega \Rightarrow \hat{p}^{\Gamma}(\sigma) + \overline{p}(\mathcal{J}) \leq m\omega$.*

Lemma 3 *If for any $j \in \mathcal{J}$, $\overline{p}_j \leq \epsilon\omega$ and $\hat{p}_j \leq \epsilon\omega$, then for any i, $C(\sigma_i) \leq \min((2 + 2\epsilon)\omega, 3\omega)$*

Proof In the worst case, before adding the last job j in the interior while loop we had $\overline{p}(X) = \omega$ and $\hat{p}^{\Gamma}(X) = \omega$, and thus $C(\sigma_i) \leq 2\omega + \overline{p}_j + \hat{p}_j$ with $\overline{p}_j + \hat{p}_j \leq \min(2\epsilon\omega, \omega)$ (if there is a job with $\overline{p}_j + \hat{p}_j > \omega$, we can immediatly reject ω). □

Lemma 4 *If A fails, then* opt $> \omega$.

Proof Let us suppose that A fails and suppose by contradiction that opt $\leq \omega$. We say that machine i is of type 1 iff $\hat{p}^{\Gamma}(\sigma_i) > \omega$, and is of type 2 otherwise. Notice that a schedule on a machine of type 1 contains at most Γ jobs (as jobs are added by non-increasing $\hat{p}_j$), and a schedule σ_i on a machine of type 2 verifies $\overline{p}(\sigma_i) > \omega$. Let $\mathcal{M}_1$ be the set of machines of type 1, and let $\mathcal{J}_1$ be the set of jobs scheduled by A in machines $\mathcal{M}_1$. Let $\mathcal{M}_2$ and $\mathcal{J}_2$ be defined in the same way. We have

- $\overline{p}(\mathcal{J}_2) > |\mathcal{M}_2|\omega$ by definition of type 2
- $\hat{p}(\mathcal{J}_1) > |\mathcal{M}_1|\omega$ by definition of type 1
- $\hat{p}(\sigma^*) \geq \hat{p}(\mathcal{J}_1)$

Let us prove the last item. Notice first that for any schedule σ' of $\mathcal{J}_1$ on m machines such that $C(\sigma') \leq \omega$, $\hat{p}(\sigma') = \hat{p}(\mathcal{J}_1)$. Indeed, let i be the last machine in $\mathcal{M}_1$ and let $x = |\sigma_i|$. Notice that as we select the jobs by non-increasing order of $\hat{p}_j$ in the interior while loop, σ_i contains the x smallest jobs (in terms of $\hat{p}_j$) of $\mathcal{J}_1$. As i is type 1 we get $\hat{p}^{\Gamma}(\sigma_i) > \omega$, and we deduce that in any schedule of $\mathcal{J}_1$ that fits in ω, there is at most $x \leq \Gamma$ jobs on every machine. Thus, all jobs deviate in σ', and $\hat{p}(\sigma') = \hat{p}(\mathcal{J}_1)$.

Then, notice that $\hat{p}(\sigma^*) \geq \hat{p}(\sigma^*_{|\mathcal{J}_1|})$ where $\sigma^*_{|\mathcal{J}_1|}$ is the schedule we obtain by starting from σ^* and only keeping jobs of $\mathcal{J}_1$ on each machine (and removing idle time). Thus, as $\text{opt}_{|\mathcal{J}_1|}$ is a schedule of $\mathcal{J}_1$ on m machines that fits in ω, we know that $\hat{p}(\sigma^*_{|\mathcal{J}_1|}) = \hat{p}(\mathcal{J}_1)$, concluding the proof of the last item.

Thus, we get $\overline{p}(\mathcal{J}) + \hat{p}(\sigma^*) \geq \overline{p}(\mathcal{J}_2) + \hat{p}(\mathcal{J}_1) > m\omega$, and thus according to Observation 2 we deduce $\text{opt} > \omega$, a contradiction. □

Lemmas 3 and 4 directly imply Theorem 3.

3.2 Fixed Γ: PTAS

Let $\epsilon > 0$. Our objective is to design an $(1+\epsilon)$-dual approximation algorithm A. Let $\mathcal{I}$ be an instance of $P||U_p^{\Gamma}|C_{max}$ with m machines and a job set $\mathcal{J}$. Let ω be the current value of the guess. Consider a small positive number δ whose value will be specified later according to Γ and ϵ .

The key observations leading to the algorithm can be summarized as follows. Small deviations ($\hat{p}_j \leq \delta w$) lead to an additive error of $\Gamma\delta w$, which can be neglected when Γ is constant (see Observation 5 below). Large deviations ($\hat{p}_j > \delta w$) must be addressed carefully as the number of jobs with large deviations on a machine may not be constant, which we handle by using partial profiles (see Definition 2 below).

Let us partition $\mathcal{J}$ into $\mathcal{J} = \widehat{\overline{B}} \cup \overline{B} \cup \widehat{B} \cup S$, where $\widehat{\overline{B}} = \{j | \hat{p}_j > \delta\omega \text{ and } \overline{p}_j > \delta\omega\}$, $\overline{B} = \{j | \hat{p}_j \leq \delta\omega \text{ and } \overline{p}_j > \delta\omega\}$, $\widehat{B} = \{j | \hat{p}_j > \delta\omega \text{ and } \overline{p}_j \leq \delta\omega\}$, and $S = \{j | \hat{p}_j \leq \delta\omega \text{ and } \overline{p}_j \leq \delta\omega\}$. Call a job j small if $j \in S$, and big otherwise.

Let us define $\mathcal{I}'$ where we geometrically round down the size of all big jobs. More formally, let $k = \lceil \log_{1+\delta} \frac{1}{\delta} \rceil$. For any j, if $\overline{p}_j \in]\omega\delta(1+\delta)^r, \omega\delta(1+\delta)^{r+1}]$ for some $r \in [k]$ then $\overline{p}'_j = \omega\delta(1+\delta)^r$ (otherwise we define $\overline{p}'_j = \overline{p}_j$), and if $\hat{p}_j \in]\omega\delta(1+\delta)^r, \omega\delta(1+\delta)^{r+1}]$ for some $r \in [k]$ then $\hat{p}'_j = \omega\delta(1+\delta)^r$ (otherwise we define $\hat{p}'_j = \hat{p}_j$). Notice that a job j in $\widehat{\overline{B}}$ has been rounded twice (i.e., $\hat{p}_j$ and $\overline{p}_j$ have been rounded), whereas a job j in $\overline{B}$ or $\widehat{B}$ has been rounded only once. Then, we define $\mathcal{I}''$ from $\mathcal{I}'$ by setting to zero any small deviation: for any j, if $\hat{p}'_j \leq \delta\omega$ we define $\hat{p}''_j = 0$, and otherwise we define $\hat{p}''_j = \hat{p}'_j$.

Observation 3 $\text{opt}(\mathcal{I}'') \leq \text{opt}(\mathcal{I}') \leq \text{opt}(\mathcal{I})$.

Observation 4 *From any schedule σ' of $\mathcal{I}'$, we can deduce a schedule σ of $\mathcal{I}$ (by simply defining $\sigma = \sigma'$) such that $C(\sigma) \leq (1+\delta)C(\sigma')$.*

Observation 5 *From any schedule σ'' of $\mathcal{I}''$, we can deduce a schedule σ' of $\mathcal{I}'$ (by simply defining $\sigma' = \sigma''$) such that $C(\sigma') \leq C(\sigma'') + \Gamma\delta\omega$.*

The idea behind $\mathcal{I}''$ is that neglecting the small deviation is possible as the extra additive factor $\Gamma\delta\omega$ can be set to a negligeable amount by setting δ sufficiently small ($\approx \frac{\epsilon}{\Gamma}$) as Γ is constant. We will show next how to solve $\mathcal{I}''$ approximately. We partition the jobs of $\mathcal{I}''$ using the same partition as in $\mathcal{I}$ (i.e., according to threshold $\delta\omega$) obtaining four subsets of jobs $\widehat{\overline{B}}''$, $\overline{B}''$, $\widehat{B}''$, and S''. Notice that:

- for any job $j \in \widehat{\overline{B}}''$, $\overline{p}''_j$ and $\hat{p}''_j$ can take at most k different values;
- for any job $j \in \overline{B}''$, $\overline{p}''_j$ can take at most k different values, and $\hat{p}''_j = 0$;
- for any job $j \in \widehat{B}''$, $\overline{p}''_j$ can take arbitrary values (but below $\delta\omega$), and $\hat{p}''_j$ can take at most k different values;
- for any job $j \in S''$, $\overline{p}''_j$ can take arbitrary values (but below $\delta\omega$), and $\hat{p}''_j = 0$.

Theorem 4 *There exists a polynomial $(1+\Gamma\delta)$-dual approximation algorithm for $\mathcal{I}''$.*

The key elements of the algorithm used to prove Theorem 4 follow.

Definition 2 *Given a schedule σ''_i of a machine i in $\mathcal{I}''$, let us define $\tilde{\sigma}''_i$, the* partial profile *associated to σ''_i as follows. For any $r_1, r_2 \in [k]$, let $\hat{\overline{n}}_i^{r_1,r_2} = |\{j \in \sigma''_i \cap \widehat{\overline{B}}'' \ |\overline{p}''_j = w\delta(1+\delta)^{r_1}$ and $\hat{p}''_j = w\delta(1+\delta)^{r_2}\}|$, let $\overline{n}_i^{r_1} = |\{j \in \sigma''_i \cap \overline{B}''|\overline{p}''_j = w\delta(1+\delta)^{r_1}\}|$, and let $\hat{n}_i^{r_1} = |\{j \in \Gamma(\sigma''_i) \cap \widehat{B}''|\hat{p}''_j = w\delta(1+\delta)^{r_1}\}|$. We define $\tilde{\sigma}''_i = (t_i^1, t_i^2, t_i^3)$, where $t_i^1 = (\hat{\overline{n}}_i^{r_1,r_2}, r_1 \in [k], r_2 \in [k])$, $t_i^2 = (\overline{n}_i^{r_1}, r_1 \in [k])$, and $t_i^3 = (\hat{n}_i^{r_1}, r_1 \in [k])$.*

Informally, the partial profile gives us the exact size all jobs of $\widehat{\overline{B}}''$ and $\overline{B}''$ and only the value $\hat{p}''_j$ for the jobs in $\widehat{B}''$ that deviate (which are the Γ jobs with largest value $\hat{p}''_j$). We extend the notion to $\tilde{\sigma}'' = \{\tilde{\sigma}''_i, i \in [m]\}$.

Observation 6 *The number of possible partial profiles on a machine is bounded by a constant k'. Indeed, notice that $\hat{\overline{n}}_i^{r_1,r_2} \leq \frac{1}{\delta}$ and $\overline{n}_i^{r_1} \leq \frac{1}{\delta}$ (as all these jobs have $\overline{p}''_j > \delta w$), and $\hat{n}_i^{r_1} \leq \Gamma$ (as we only keep the deviating jobs to define $\hat{n}_i^{r_1}$). Thus, we can take for example $k' = \frac{1}{\delta}^{k^2} \frac{1}{\delta}^{k} \Gamma^k$.*

Let us define a polynomial algorithm A that either schedules $\mathcal{I}''$ in $(1+\Gamma\delta)\omega$ or fails, implying that $\omega < \text{opt}(\mathcal{I}'')$. Let σ^* be an optimal solution of $\mathcal{I}''$. For each $l \in [k']^*$, where k' is defined in Observation 6, we enumerate on how many machines have a partial schedule of type l. Thus, we will run A on the $\mathcal{O}(m^{k'})$ possible $\tilde{\sigma}$, and we now assume that A takes $\tilde{\sigma}^*$ as an input. For any machine, let $\hat{p}_i^*$ be the size of the smallest deviating job on machine i. More formally, using $r_i^{min} = \min\{r|(\exists r_1$ such that $\hat{\overline{n}}_i^{r_1,r} \neq 0)$ or $\hat{n}_i^r \neq 0\}$, we define $\hat{p}_i^* = w\delta(1+\delta)^{r_i^{min}}$. Let us assume w.l.o.g. that $\hat{p}_i^* \leq \hat{p}_{i+1}^*$. In Algorithm 2, we explain in details how A creates a schedule σ'' of jobs of $\mathcal{I}''$ given this partial profile. We recall that for any integer x, $[x]^* = [x] \setminus \{0\} = \{1, \ldots, x\}$.

Observation 7 *$\hat{p}^\Gamma(\sigma'') = \hat{p}^\Gamma(\sigma^*)$. Indeed, if two schedules $\sigma^{(\prime\prime,1)}$ and $\sigma^{(\prime\prime,2)}$ of $\mathcal{I}''$ have the same partial profile, then $\hat{p}^\Gamma(\sigma^{(\prime\prime,1)}) = \hat{p}^\Gamma(\sigma^{(\prime\prime,2)})$.*

For any i, let $\sigma^*_{i|big} = \sigma^*_i \setminus S''$. Let us also define $\mathcal{M}_1 \cup \cdots \cup \mathcal{M}_{m'}$ as the partition of $[m]^*$ of minimal cardinality m' such that $\hat{p}_i^* = \hat{p}_j^*$ for each pair $i, j \in \mathcal{M}_g$ and each $g \in [m']^*$ (for example with $m = 4$ if $\hat{p}_1^* = 10, \hat{p}_2^* = 15, \hat{p}_3^* = 15$ and $\hat{p}_4^* = 20$, we have $\mathcal{M}_1 = \{1\}, \mathcal{M}_2 = \{2, 3\}$ and $\mathcal{M}_3 = \{4\}$). For any $g \in [m']^*$, let us also denote by x_g the value of $\hat{p}_i^*$ for any $i \in \mathcal{M}_g$.

Algorithm 2 $A(\mathcal{I}'', \tilde{\sigma}^*, \omega)$

Phase 1 A schedules jobs in $X_1 = \widehat{\widehat{B}}'' \cup \overline{B}''$ by scheduling all jobs of $\widehat{\widehat{B}}''$ and $\overline{B}''$ according to $\tilde{\sigma}^*$. Notice that we have $|X_1| = \sum_{i\in[m]^*, r_1\in[k], r_2\in[k]} \hat{n}_i^{r_1,r_2} + \sum_{i\in[m]^*, r\in[k]} \overline{n}_i^r$

Phase 2 A schedules jobs in $X_2 \subseteq \widehat{B}''$ (X_2 are the deviating jobs of $\widehat{B}''$) as follows. For i from 1 to m, for any r, A schedules the $\hat{n}_i^r$ remaining job $j \in \widehat{B}''$ with $\hat{p}_j'' = w\delta(1+\delta)^r$ having the largest $\overline{p}_j$, and schedules them on machine i. Notice that we have $|X_2| = \sum_{i\in[m]^*, r\in[k]} \hat{n}_i^r$.

Phase 3 A schedules jobs in $X_3 = \widehat{B}'' \setminus X_2$ as follows. Let $\widehat{A}_i = \{j \in \widehat{B}'' \setminus X_2$ such that $\hat{p}_j'' \leq \hat{p}_i^*\}$ be the set of remaining jobs of $\widehat{B}''$ that are authorized on i. For i from 1 to m, while $\widehat{A}_i \neq \emptyset$ and $C(\sigma_i'') \leq \omega$, A schedules arbitrary jobs of $\widehat{A}_i$ on i .At the end of Phase 3, if there remains some unscheduled jobs in $\widehat{B}''$, then A fails, otherwise it goes to Phase 4.

Phase 4 A schedules jobs of $X_4 = S''$ by picking any $j \in X_4$ and any i such that $C(\sigma_i'') \leq \omega$ and scheduling j on i. At the end of Phase 4, if there remains some unscheduled jobs in S'', then A fails.

Lemma 5 *If $C(\sigma^*) \leq \omega$, then during Phase 3, for any $g \leq m'$, after scheduling the last machine of $\mathcal{M}_g$ we have $\overline{p}(\bigcup_{i\in\mathcal{M}_1\cup\ldots\cup\mathcal{M}_g} \sigma^*_{i|big}) \leq \overline{p}(\bigcup_{i\in\mathcal{M}_1\cup\ldots\cup\mathcal{M}_g} \sigma_i'')$. Informally, the total (non-deviating) processing time scheduled by A on machines from $\mathcal{M}_1 \cup \ldots \cup \mathcal{M}_g$ is greater than the one scheduled in $\sigma^* \setminus S''$.*

Proof Let us suppose this is true for $g-1$ and prove it for g. Let us first consider the case where there exists $i \in \mathcal{M}_g$ such that A schedules all jobs of $\widehat{A}_i$ on i. This implies that at the end of Phase 3, A did not schedule any other job from X_3 on machines $\{(i+1), \ldots, i_{max}\}$ where i_{max} is the last machine of $\mathcal{M}_g$. Let us partition $Z^* = \bigcup_{l\in\mathcal{M}_1\cup\ldots\cup\mathcal{M}_g} \sigma^*_{l|big}$ into $\widehat{\widehat{Z}}^* = Z^* \cap \widehat{\widehat{B}}''$, $\overline{Z}^* = Z^* \cap \overline{B}''$ and $\hat{Z}^* = Z^* \cap \widehat{B}''$, and define the same partition for $Z = \bigcup_{l\in\mathcal{M}_1\cup\ldots\cup\mathcal{M}_g} \sigma_l''$. Notice that Phase 1 garantees that $\widehat{\widehat{Z}}^* \cup \overline{Z}^* = \widehat{\widehat{Z}} \cup \overline{Z}$. Thus, it remains to prove that $\overline{p}(\hat{Z}^*) \leq \overline{p}(\hat{Z})$. Let $J_{>\hat{p}_i^*} = \{j \in \mathcal{I}'' \setminus S''$ with $\hat{p}_j > \hat{p}_i^* = x_g\}$ and $J_{\leq\hat{p}_i^*} = \{j \in \mathcal{I}'' \setminus S''$ with $\hat{p}_j \leq \hat{p}_i^* = x_g\}$. Notice that all jobs of $\hat{Z} \cap J_{>\hat{p}_i^*}$ deviate, and thus have been scheduled in Phase 2. Thus, as Phase 2 chooses at each step the largest remaining $\overline{p}_j$ we get $\overline{p}(\hat{Z}^* \cap J_{>\hat{p}_i^*}) \leq \overline{p}(\hat{Z} \cap J_{>\hat{p}_i^*})$. Now, we will prove that $\hat{Z}^* \cap J_{\leq\hat{p}_i^*} \subseteq \hat{Z} \cap J_{\leq\hat{p}_i^*}$ (which together with the previous inequality implies the desired result $\overline{p}(\hat{Z}^*) \leq \overline{p}(\hat{Z})$). To prove this, it is sufficient to remark that any $j \in \hat{Z}^* \cap J_{\leq\hat{p}_i^*}$ is either in $\widehat{A}_i$ (implying immediately the claim) or in X_2 (implying that j was scheduled in Phase 2). In the last case, observe that whenever Phase 2 schedules j on a machine i', then $\hat{p}_j'' \geq \hat{p}_{i'}^*$, and thus j must have been scheduled by Phase 2 on a machine $i' \leq i_{max}$, and thus $j \in \hat{Z} \cap J_{\leq\hat{p}_i^*}$.

It remains to analyse the case where for any $i \in \mathcal{M}_g$, A stops filling i as $C(\sigma_i'') > \omega$. As $\hat{p}^\Gamma(\sigma_i'') = \hat{p}^\Gamma(\sigma_i^*)$, we have $\overline{p}(\sigma_i'') > \omega - \hat{p}^\Gamma(\sigma_i'') \geq \overline{p}(\sigma^*_{i|big})$, and using the induction hypothesis the result is immediate. □

Corollary 1 (of Lemma 5) *If A fails after Phase 3, then $\omega < C(\sigma^*)$.*

Proof Suppose $C(\sigma^*) \leq \omega$. Applying Lemma 5 to the last group $\mathcal{M}_{m'}$, we get $\overline{p}(\mathcal{I}'' \setminus S'') = \overline{p}(\bigcup_{i \in \mathcal{M}_1 \cup \ldots \cup \mathcal{M}_{m'}} \sigma^*_{i|big}) \leq \overline{p}(\bigcup_{i \in \mathcal{M}_1 \cup \ldots \cup \mathcal{M}_{m'}} \sigma''_i) = \overline{p}(X_1 \cup X_2 \cup X_3)$. If A fails after Phase 3, it means that $X_1 \cup X_2 \cup X_3 \subset \mathcal{I}'' \setminus S''$, and we get $\overline{p}(\mathcal{I}'' \setminus S'') > \overline{p}(X_1 \cup X_2 \cup X_3)$, a contradiction. □

Lemma 6 *If A fails after Phase 4, then $\omega < C(\sigma^*)$.*

Proof According to Observation 7 we have $\hat{p}^{\Gamma}(\sigma'') = \hat{p}^{\Gamma}(\sigma^*)$. If A fails after Phase 4, we have $C(\sigma''_i) > \omega$ for any i, and thus $\overline{p}(\mathcal{I}'') > mw - \hat{p}^{\Gamma}(\sigma'') = mw - \hat{p}^{\Gamma}(\sigma^*)$, implying that $\omega < C(\sigma^*)$. □

Lemma 7 *If A does not fail and produces σ'', then $C(\sigma'') \leq (1 + \Gamma\delta)\omega$.*

Proof In Phase 1, A does not exceed ω by construction. At the end of Phase 2 we have $C(\sigma'') \leq (1 + \Gamma\delta)\omega$ as in the worse case t^3_i refers to Γ jobs, and each of these jobs has $\overline{p}''_j = \delta\omega$. As jobs in Phase 3 cannot deviate, scheduling a job $j \in \widehat{B}'' \setminus X_2$ on a machine i having $C(\sigma''_i) < \omega$ can increase C to most $(1+\delta)\omega$. As the bound is also $(1 + \delta)\omega$ for Phase 4, we get the desired result. □

Lemma 6 and 7 together prove Theorem 4.

Corollary 2 *$P||U^{\Gamma}_p|C_{max}$ admits a PTAS.*

Proof Given $\epsilon > 0$, and $\mathcal{I}$ an instance of $P||U^{\Gamma}_p|C_{max}$ we provide a $(1+\epsilon)$-dual approximation algorithm in the following way. Let ω be the current guess of $\text{opt}(\mathcal{I})$. We define $\mathcal{I}''$ as previously, and run the (polynomial time) algorithm $A(\mathcal{I}'', \tilde{\sigma}^*, \omega)$ from Theorem 4. If A fails, then we also fail on $\mathcal{I}$ according to Observation 3. Otherwise, according to Theorem 4, Observation 4, and Observation 5 we get a schedule σ of $\mathcal{I}$ with $C(\sigma) \leq (1+\delta)((1+\Gamma\delta)\omega + \Gamma\delta\omega)$, and thus we set δ such that $(1 + \delta)(1 + 2\Gamma\delta) = 1 + \epsilon$. □

4 Minimizing makespan on unrelated machines

In this section, we denote by $\bar{p}_{ij}$ and $\hat{p}_{ij}$ respectively the mean and deviating processing times for job $j \in \mathcal{J} = \{1, \ldots, n\}$ on machine $i \in \mathcal{M} = \{1, \ldots, m\}$.

4.1 Constant number of machines

We provide below a pseudopolynomial dynamic programing algorithm for $Rm||U^{\Gamma}_p|C_{max}$. Deducing an FPTAS will be straightforward by following the same approach as in [12]. Observe that the main difficulty here is to keep track of the deviating jobs, especially when Γ is not constant. For $Qm||U^{\Gamma}_p|C_{max}$, this difficulty can be handled by ordering the jobs in non-increasing order of $\hat{p}_j$. Namely, given two m-dimensional vectors l and x, and an integer j, we write a dynamic programming algorithm $DP(l, x, j)$ that keeps track for every machine of its current total load l_i (including deviations) and of the number of

deviating jobs $x_i \in [\Gamma]$, and computes the optimal makespan when scheduling jobs $\{j' \geq j\}$. Let s_i be the speed of machine i. Due to the ordering of the jobs, the potential contribution of job j to machine i is $\frac{\overline{p}_j}{s_i}$ if $x_i = \Gamma$, and $\frac{\overline{p}_j + \hat{p}_j}{s_i}$ otherwise.

We show below how to extend the approach to $Rm||U_p^{\Gamma}|C_{max}$ for which we cannot define such an ordering of the jobs. Let u be an upper bound on opt. Without loss of generality, we add enough dummy jobs (with $\hat{p}_{ij} = \overline{p}_{ij} = 0$) so that we can suppose that there are at least Γ jobs on each machine. Thus, we add in the problem the extra constraint that there must be exactly Γ deviating jobs on each machine.

We first guess for each machine machine what is the size $\hat{p}_i^*$ of the smallest deviating job in an optimal solution σ^*. This means that for any i and any job j scheduled on i in σ^*, $j \in \Gamma(\sigma_i^*)$ implies $\hat{p}_{ij} \geq \hat{p}_i^*$. Then, we consider the algorithm $DP(l, x, j)$ that remembers for any machine i the total load $l_i \in [u]$ (including deviations), the number of already deviating jobs $x_i \in [\Gamma]$, and computes the optimal makespan when scheduling jobs $\{j' \geq j\}$. More formally, given (l, x, j), we define the two following notions. The subinstance $\mathcal{I}_j$ is the subinstance containing the m machines and jobs $\{j' \geq j\}$. For any schedule $\sigma = \{\sigma_i\}$ of $\mathcal{I}_j$ we define $C^i(\sigma_i) = \overline{p}(\sigma_i) + \hat{p}^{\Gamma - x_i}(\sigma_i) + l_i$ (we denote by $\hat{p}^{\Gamma - x_i}(\sigma_i)$ the sum of the $\hat{p}$ of the $\Gamma - x_i$ largest jobs of σ_i, and thus C^i represents the total makespan including the Γ deviating jobs), and $C'(\sigma) = \max_i C^i(\sigma_i)$. Our objective is now to define $DP(l, x, j)$ that computes a σ minimizing C'.

To that end, $DP(l, x, j)$ branches m times to decide where j is scheduled, and calls $DP(l', x', j+1)$. If j is scheduled on i, then

- if $\hat{p}_{ij} < \hat{p}_i^*$ then (l_i, x_i) becomes $(l_i + \overline{p}_{ij}, x_i)$,
- if $\hat{p}_{ij} > \hat{p}_i^*$ and $x_i = \Gamma$ then (l_i, x_i) becomes (∞, x_i) (j cannot be scheduled on i),
- if $\hat{p}_{ij} > \hat{p}_i^*$ and $x_i < \Gamma$ then (l_i, x_i) becomes $(l_i + \overline{p}_{ij} + \hat{p}_{ij}, x_i + 1)$,
- if $\hat{p}_{ij} = \hat{p}_i^*$ and $x_i = \Gamma$ then (l_i, x_i) becomes $(l_i + \overline{p}_{ij}, x_i)$ (there are already Γ deviating jobs and job j must not deviate on machine i),
- if $\hat{p}_{ij} = \hat{p}_i^*$ and $x_i < \Gamma$ then (l_i, x_i) becomes either $(l_i + \overline{p}_{ij}, x_i)$ or $(l_i + \overline{p}_{ij} + \hat{p}_{ij}, x_i + 1)$ (the algorithm branches to choose if j deviates or not).

Notice that the two last items are necessary when for example $\sigma_i^* = \{j_1, j_2, j_3, j_4\}$ with $\hat{p}_{ij_1} > \hat{p}_{ij_2} = \hat{p}_{ij_3} > \hat{p}_{ij_4}$ and $\Gamma = 2$: only one of the two jobs of processing time $\hat{p}_i^* = \hat{p}_{ij_2}$ deviates. Finally, when $j = n + 1$ (i.e., all the jobs are scheduled), if one of the x_i is strictly lower than Γ then $DP(l, x, j)$ returns $+\infty$ (remember that we impose that there are exactly Γ deviating jobs on each machine), otherwise it returns $\max_i(l_i)$. This concludes the description of DP. Once the $\{\hat{p}_i^*\}$ are fixed, DP runs in $\mathcal{O}((u(\Gamma+1))^m nm)$, and thus the overall running time is in $\mathcal{O}((nu(\Gamma+1))^m nm)$.

We can deduce an FPTAS from this pseudopolynomial DP by using the same arguments as in [12]. Let l be a lower bound, and d an integer. We round each $\hat{p}_{ij} \in]qd, (q+1)d]$ to $\hat{p}'_{ij} = (q+1)d$ (and similarly for the $\{\overline{p}_{ij}\}$), and get a new instance $\mathcal{I}'$ with $\text{opt}(\mathcal{I}') \leq \text{opt}(\mathcal{I}) + 2nd$. We solve $\mathcal{I}'$ with the previous DP in $\mathcal{O}((nu'(\Gamma+1))^m nm)$ where $u' = \frac{u}{d}$. Thus, as we need $2nd \leq \epsilon l$, we

take $d = \frac{\epsilon l}{2n}$ and get an FPTAS as we can find u and l with $\frac{u}{l} = n$ with for example any trivial n-approximation algorithm.

4.2 The number of machines belongs to the input

Let us start with a simple result showing that for small values of Γ we can simply re-use any existing approximation algorithm.

Lemma 8 *From any polynomial-time ρ-approximation A for $R||C_{max}$ we can deduce a polynomial $(\rho + \Gamma)$-dual approximation A^Γ for $R||U_p^\Gamma|C_{max}$. In particular, [17] gives a $2 + \Gamma$ dual approximation.*

Proof Let $\mathcal{I}$ be an instance of $R||U_p^\Gamma|C_{max}$. Let ω be the current value of the guess. Without loss of generality, for any i and j we can suppose that either $((\hat{p}_{ij} \leq \omega)$ and $(\overline{p}_{ij} \leq \omega))$, or $((\hat{p}_{ij} > \omega)$ and $(\overline{p}_{ij} > \omega))$. Indeed, if $(\overline{p}_{ij} > \omega)$ and $(\hat{p}_{ij} \leq \omega)$, then j cannot be processed on i, and thus we can set $\hat{p}_{ij} = \omega+1$. If $(\overline{p}_{ij} \leq \omega)$ and $(\hat{p}_{ij} > \omega)$ then again j cannot be processed on i (as either j or a job j' with $\hat{p}_{ij'} \geq \hat{p}_{ij}$ will deviate), and we set $\overline{p}_{ij} = \omega + 1$. Then, we define $\mathcal{I}'$ as the corresponding instance of $R||C_{max}$ without deviation ($\mathcal{I}'$ has m machines, and $p'_{ij} = \overline{p}_{ij}$), and we compute $A(\mathcal{I}')$. If $A(\mathcal{I}') > \rho\omega$, then we reject as it implies that $\text{opt}(\mathcal{I}') > \omega$ (and $\text{opt}(\mathcal{I}') \leq \text{opt}(\mathcal{I})$). Otherwise, we keep the schedule σ as computed by $A(\mathcal{I}')$, and as for any job j scheduled on a machine i we have $\hat{p}_{ij} \leq \omega$, we have $C(\sigma) \leq \rho\omega + \Gamma\omega$. □

We provide a more refined algorithm that yields an average $O(\log m)$ approximation factor. Notice that the straightforward generalization of the formulation from [17] is not useful in the robust context because its fractional solution may contain up to nm fractional variables. Hence, we must use a different approach, based on the extended formulation described next.

Define, for each $i \in \mathcal{M}$ and each $\nu \subseteq \mathcal{J}$, $\lambda_{i\nu} = 1$ if the set of jobs executed on machine i is precisely ν, and zero otherwise. Let $\mu(j, \nu) = 1$ if $j \in \nu$, and zero otherwise, and $\alpha(i, \nu)$ be the robust completion time of machine i when it executes the jobs in ν, formally $\alpha(i, \nu) = \max\left\{\sum_{j\in\nu}(\bar{p}_{ij} + \xi_j\hat{p}_{ij}) | \xi \in \{0, 1\}^n, \sum_{j\in\mathcal{J}} \xi_j \leq \Gamma\right\}$. The formulation follows:

$$
\begin{array}{lll}
\text{Min} & \omega & \\
\text{S.t.} & \displaystyle\sum_{i\in\mathcal{M}}\sum_{\nu\subseteq\mathcal{J}} \mu(j,\nu)\lambda_{i\nu} = 1, & \forall j \in \mathcal{J} \\
& \omega \geq \displaystyle\sum_{\nu\subseteq\mathcal{J}} \alpha(i,\nu)\lambda_{i\nu}, & \forall i \in \mathcal{M} \\
& \displaystyle\sum_{\nu\subseteq\mathcal{J}} \lambda_{i\nu} = 1, & \forall i \in \mathcal{M} \\
& \lambda_{i\nu} \in \{0, 1\}, & \forall (i,\nu) \in \mathcal{M} \times 2^{\mathcal{J}}
\end{array}
$$

As with the formulation from [17], the value of the lower bound improves if we drop the objective function and remove all variables $\lambda_{i\nu}$ such that $\alpha(i,\nu)$ is greater than a given target makespan value ω. Namely, we consider the lower bound for $R||U^T|C_{max}$ defined as follows

$$LB = \{\min \omega : FP(\omega) \text{ is feasible}\}, \tag{5}$$

where $FP(\omega)$ is defined by the following linear constraints:

$$\sum_{i\in\mathcal{M}} \sum_{\substack{\nu\subseteq\mathcal{J}\\ \alpha(i,\nu)\leq\omega}} \mu(j,\nu)\lambda_{i\nu} = 1, \quad \forall j \in \mathcal{J} \tag{6}$$

$$\sum_{\substack{\nu\subseteq\mathcal{J}\\ \alpha(i,\nu)\leq\omega}} \lambda_{i\nu} = 1, \quad \forall i \in \mathcal{M} \tag{7}$$

$$\lambda_{i\nu} \geq 0, \quad \forall (i,\nu) \in \mathcal{M} \times 2^{\mathcal{J}} \tag{8}$$

We show below how we can assert in polynomial time whether $FP(\omega)$ is infeasible or prove its feasibility for 2ω. This algorithm can be further combined with a binary search on the minimum value ω for which $FP(\omega)$ is feasible, yielding the following result.

Theorem 5 *We can compute in polynomial time a 2-approximate solution for LB.*

Proof We solve problem (5) using a dual-approximation algorithm. Namely, for each value of ω, either we show that $FP(2w)$ is feasible or that $FP(\omega)$ is infeasible. Then, the minimum value of ω for which $FP(\omega)$ is feasible that leads to zero objective value can be found through a binary search.

Let ω be the current value of the guess. We can check the feasibility of $FP(\omega)$ by adding artificial variables s_j that allow penalized infeasibilities, leading to the following linear program.

$$\text{Min} \quad \sum_{j\in\mathcal{J}} s_j \tag{9}$$

$$\text{S.t.} \quad \sum_{i\in\mathcal{M}} \sum_{\substack{\nu\subseteq\mathcal{J}\\ \alpha(i,\nu)\leq\omega}} \mu(j,\nu)\lambda_{i\nu} + s_j = 1, \qquad \forall j \in \mathcal{J} \tag{10}$$

$$\sum_{\substack{\nu\subseteq\mathcal{J}\\ \alpha(i,\nu)\leq\omega}} \lambda_{i\nu} = 1, \qquad \forall i \in \mathcal{M} \tag{11}$$

$$\lambda_{i\nu} \geq 0, \qquad \forall (i,\nu) \in \mathcal{M} \times 2^{\mathcal{J}}, \alpha(i,\nu) \leq \omega \tag{12}$$

$$s_j \geq 0, \qquad \forall j \in \mathcal{J} \tag{13}$$

The continuous relaxation of the previous formulation can be solved in polynomial time, using for instance the Ellipsoid method [13], if the problem of pricing the λ variables is also polynomially solvable [11]. Such a pricing problem can be stated as follows. Let π_j, and θ_i be dual variables associated to constraints (10), and (11), respectively. The reduced cost of the variable $\lambda_{i\nu}$, denoted by $\bar{c}(\lambda_{i\nu})$, is equal to $-\sum_{j\in\nu}\pi_j - \theta_i$.

Then, for each $i \in \mathcal{M}$, we want to find $\nu \in \mathcal{J}$ that maximizes $\sum_{j\in\nu}\pi_j$ subject to $\alpha(i,\nu) \leq \omega$. This problem is the robust binary knapsack problem, which is an $\mathcal{NP}$-hard problem. Hence, suppose that we can compute in polynomial time a solution ν^* with reduced cost $\bar{c}^*$ such that $\alpha(i,\nu) \leq 2\omega$ and such that no solution with a smaller reduced cost exists where $\alpha(i,\nu) \leq \omega$. We obtain a relaxed primal solution that may use variables $\lambda_{i\nu}$ with $\alpha(i,\nu)$ up to 2ω, and whose objective value is not greater than the optimal value of a linear program where all variables $\lambda_{i\nu}$ have $\alpha(i,\nu) \leq \omega$. As a result, a positive value on the objective function ensures that $FP(\omega)$ is infeasible while a null value provides a fractional feasible solution for $FP(2\omega)$.

It remains to show how to find the solution ν^*. Remark that if $\hat{p} = 0$ (the problem is deterministic), such a solution ν^* can be found by using the greedy algorithm for the knapsack problem and rounding up the unique fractional variable. Then, one readily verifies that the deterministic approach can be extended to the robust context by solving $n+1$ deterministic problems in the spirit of Theorem 1 and its extension to robust constraints, as studied in [9]. □

In the remainder of the section, we let ω be the solution returned by Theorem 5 and λ^* be the corresponding fractional vector. Our objective is to use randomized rounding to obtain an integer solution to $R||U^\Gamma|C_{\max}$. We first present a straightforward analysis showing that the approximation ratio of the solution is at most $O(\log(n))\omega$. Then, we present a more elaborate analysis by considering the probability that a job is already scheduled when scheduling it again. The latter abalysis leads to a ratio of $O(\log(m))\omega$. Since $\omega/2$ is a lower bound for opt, this will lead to an average $O(\log(m))$-approximation ratio for $R||U^\Gamma|C_{\max}$ (see Theorem 7).

The proposed rounding procedure iteratively adds schedules to all machines until every job is assigned to one of the machines. At each iteration, one additional schedule is selected for each machine and added to the current solution, allowing that the same schedule is added more than once to a given machine. The procedure maintains a variable y_{ij} for each machine i and each job j representing the number of times that job j belongs to a scheduled that is added to machine i. These variables are used only to prove the approximation bound on the obtained makespan. The integer solution consists of simply assigning each job j to the machine that receives the first schedule that contains j.

The pseudocode for this rounding procedure is given in Algorithm 3. Let C_{max} be the random variable corresponding to the makespan of the schedule computed by Algorithm 3. Let t be the number of iterations performed by the while loop of this algorithm. Since every schedule ν associated to a variable

Algorithm 3 Randomized rounding (input: a feasible solution (λ^*) of $FP(\omega)$)

```
y ← 0;
while there exists a job j ∈ J not assigned to any machine do
  for i ← 1, ..., m do
    Randomly select a schedule ν* for machine i with probability λ*_iν of selecting each
    schedule ν;
    for each j ∈ ν* do
      y_ij ← y_ij + 1;
      if job j is not assigned to any machine then
        Assign job j to machine i;
      end if
    end for
  end for
end while
```

$\lambda_{i\nu}$ has a total processing time of at most ω, it is clear that $C_{max} \leq \omega t$. Thus, it remains to give an upper bound on the expected value of t. For that, we use the well-known Chernoff bound that can be described as follows. Given K independent random variables $X_1, \ldots, X_K$, each one taking the value 1 with certain probability and zero otherwise, such that the expected value of $X = \sum_{k=1}^{K} X_k$ is equal to μ, the probability that $X < (1-\delta)\mu$, for any $\delta > 0$, is smaller than $e^{-\mu\delta^2/2}$. The next Theorem uses this bound to limit the value of t.

Theorem 6 *The probability that $t > \lceil 4\ln(2n) \rceil$ is less than $1/2$.*

Proof Let $t^* = \lceil 4\ln(2n) \rceil$. For a given job j, machine i and iteration $q \leq t^*$ of the while loop, let $X^j_{q,i} = 1$ if the value of y_{ij} is increased during this iteration, and zero otherwise (if the algorithm stopped after $t < t^*$ iterations of the while loop then all the $X^j_{q,i}$ with $t < q \leq t^*$ are set to 0). Clearly, the random variables $X^j_{q,i}$ are independent. Moreover, the constraints (10) ensure that $\mathbb{E}(\sum_i X^j_{q,i}) = 1$ for any j and q, and thus the expected value of $X^j = \sum_{q \in [t^*]^*, i \in [m]^*} X^j_{q,i}$ is equal to t^*. Now, applying the Chernoff bound with $\delta = 1 - 1/t^*$, and assuming that $t^* \geq 4$, we obtain that

$$\Pr[X^j < 1] < e^{-\frac{(t^*-1)^2}{2t^*}} < e^{-t^*/4} \leq \frac{1}{2n}. \tag{14}$$

Note that $\Pr[X^j < 1]$ is the probability that the job j is not scheduled after t^* iterations, and that the random variables $X^1, \ldots, X^n$ are not necessarily independent. Let $X = 1$ if every job is scheduled after t^* iterations, and zero if at least one job is not scheduled. Note that $X = 0$ is equivalent to state that the Algorithm 3 does not finish after t^* iterations, i.e., $t > t^*$. Moreover, we have that

$$\Pr[X = 0] = \Pr[\sum_{j=1}^{n} X^j < n] \leq \sum_{j=1}^{n} \Pr[X^j < 1] < 1/2, \tag{15}$$

which completes our proof. □

Corollary 2 $\mathbb{E}(C_{max}) = O(\log(n))\omega$.

Proof An immediate consequence of Theorem 6 is that for any integer $c \geq 1$, the probability that $t > c \times \lceil 4\ln(2n) \rceil$ is less than $1/2^c$ (indeed, $1/2^c$ upper bounds the probabilty that none of c parallel execution of Algorithm 3 schedules all the jobs, where each run only performs $\lceil 4\ln(2n) \rceil$ iterations of the while loop). As a result, the expected value of t is smaller than $2 \times \lceil 4\ln(2n) \rceil$. □

We present below a tighter analysis of the approximation ratio of Algorithm 3.

Lemma 9 $\mathbb{E}(C_{max}) = O(\log(m))\omega$.

Proof For each value $\lambda^*_{i\nu} > 0$ considered by Algorithm 3, let $\alpha_j(i,\nu)$ be the contribution of the job j to the value of $\alpha(i,\nu)$, defined as follows.

$$\alpha_j(i,\nu) = \begin{cases} \bar{p}_j + \hat{p}_j, \text{ if } j \in \Gamma(\nu), \\ \bar{p}_j, \text{ if } j \in \nu \setminus \Gamma(\nu), \\ 0, \text{ if } j \notin \nu. \end{cases} \tag{16}$$

Clearly, $\alpha(i,\nu) = \sum_{j\in\mathcal{J}} \alpha_j(i,\nu)$. Note that the makespan of the solution computed by Algorithm 3 can be bounded by

$$C_{\max} \leq \max_{i\in\mathcal{M}} \left\{ \sum_{\ell=1}^{t} \sum_{j\in\mathcal{J}} \alpha_j(i,\nu_{i,\ell}) \right\}, \tag{17}$$

where $\nu_{i,\ell}$ is the schedule selected for machine i in the ℓth iteration. Note that $\nu_{i,\ell}$ is a random variable, and so is $\alpha(i,\nu_{i,\ell})$.

If order to improve the upper bound given by (17), we consider a new upper bound on $\mathbb{E}(C_{\max})$ where the probability that j is already scheduled when adding each term $\alpha_j(i,\nu_{i,\ell})$ is taken into account. Let $\beta(i,\ell)$ be the increase on makespan of the current schedule for the machine i at the ℓth iteration, which is defined as follows.

$$\beta(i,\ell) = \begin{cases} \alpha(i,\nu_{i,\ell}), \text{ if } \ell = 1, \\ \alpha(i,\nu'_{i,\ell}) - \alpha(i,\nu'_{i,\ell-1}), \text{ if } \ell > 1, \end{cases}$$

where $\nu'_{i,\ell} = \bigcup_{k=1}^{\ell} \nu_{i,k}$. We also define $\beta_j(i,\ell)$ as the contribution of job j to $\beta(i,\ell)$, given by

$$\beta_j(i,\ell) = \begin{cases} \alpha_j(i,\nu_{i,\ell}), \text{ if } \ell = 1, \\ \alpha_j(i,\nu'_{i,\ell}) - \alpha_j(i,\nu'_{i,\ell-1}), \text{ if } \ell > 1. \end{cases} \tag{18}$$

Note that $\beta_j(i,\ell)$ can be strictly positive only if j is not scheduled before the ℓth iteration.

Let $q_{j\ell}$ be the probability that the job j is not scheduled on any machine during the $\ell-1$ first iterations. Since each iteration corresponds to an independent and identical random try, we have that $q_{j\ell} = (q_{j2})^{\ell-1}$. Moreover, since

the schedule selections on different machines are independent and the sum of the probabilities of scheduling a given job j for all machines is equal to 1, we have that

$$\begin{aligned} q_{j2} &= \prod_{i\in\mathcal{M}} (1 - \Pr[\text{job } j \text{ is scheduled on machine } i]) \\ &\leq (1-1/m)^m < 1/e \end{aligned} \tag{19}$$

The first inequality is true because setting all probabilities equal to $1/m$ maximizes the right-hand side of (19) subject to the constraint that the sum of all probabilities is one. Hence, we have that $q_{j\ell} < 1/e^{\ell-1}$. Then, we obtain that

$$\begin{aligned} \mathbb{E}(C_{\max}) &= \mathbb{E}\left(\max_{i\in\mathcal{M}}\left\{\sum_{\ell=1}^{t}\beta(i,\ell)\right\}\right) \\ &\leq \mathbb{E}\left(\max_{i\in\mathcal{M}}\left\{\sum_{\ell=1}^{\lceil\ln m\rceil}\alpha(i,\nu_{i,\ell})\right\} + \max_{i\in\mathcal{M}}\left\{\sum_{\ell=\lceil\ln m\rceil+1}^{t}\beta(i,\ell)\right\}\right) && (20) \\ &\leq \sum_{\ell=1}^{\lceil\ln m\rceil}\omega + \sum_{i\in\mathcal{M}}\mathbb{E}\left(\sum_{\ell=\lceil\ln m\rceil+1}^{t}\sum_{j\in\mathcal{J}}\beta_j(i,\ell)\right) && (21) \\ &\leq \sum_{\ell=1}^{\lceil\ln m\rceil}\omega + \sum_{i\in\mathcal{M}}\sum_{\ell=\lceil\ln m\rceil+1}^{t}\sum_{j\in\mathcal{J}} q_{j\ell}\sum_{\nu\subseteq\mathcal{J}}\lambda^*_{i,\nu}\alpha_j(i,\nu) && (22) \\ &< \omega\lceil\ln m\rceil + \sum_{i\in\mathcal{M}}\sum_{\ell=\lceil\ln m\rceil+1}^{t}\sum_{j\in\mathcal{J}}\frac{1}{e^{\ell-1}}\sum_{\nu\subseteq\mathcal{J}}\lambda^*_{i,\nu}\alpha_j(i,\nu) \\ &= \omega\lceil\ln m\rceil + \sum_{i\in\mathcal{M}}\sum_{\ell=\lceil\ln m\rceil+1}^{t}\frac{1}{e^{\ell-1}}\sum_{\nu\subseteq\mathcal{J}}\lambda^*_{i,\nu}\sum_{j\in\mathcal{J}}\alpha_j(i,\nu) \\ &\leq \omega\lceil\ln m\rceil + \sum_{i\in\mathcal{M}}\sum_{\ell=\lceil\ln m\rceil+1}^{t}\frac{1}{e^{\ell-1}}\omega \\ &< \omega\lceil\ln m\rceil + m\frac{e}{m(e-1)}\omega \\ &= \left(\lceil\ln m\rceil + \frac{e}{e-1}\right)\omega. \end{aligned}$$

Inequality (20) follows from $\beta(i,\ell,) \leq \alpha(i,\nu_{i,\ell})$, inequality (21) follows from $\mathbb{E}(\alpha(i,\nu_{i,\ell})) \leq \omega$, which holds because of the definition of $FP(\omega)$, inequality (22) follows from the definition of $\beta_j(i,\ell)$ in (18), and the other inequalities are obtained similarly.

□

We obtain easily the following result.

Theorem 7 *There is a $O(\log(m))$-approximation in expectation for $R||U^{\Gamma}|C_{\max}$.*

Proof Let us define a randomized $O(\log m)$-dual approximation that given a threshold ω either creates a schedule with $\mathbb{E}(C_{\max}) \leq O(\log m)\omega$, or fails, implying that $\omega < \text{opt}$ (where opt is the optimal solution cost of the $R||U^{\Gamma}|C_{\max}$ input). Given ω, we apply Theorem 5 to either compute a fractional solution of cost 2ω of LB, or fail (implying $\omega < \text{opt}(LB) \leq \text{opt}$). If the algorithm does not fail, we applying Lemma 9 to round this solution to an integer solution with expected makespan $\mathbb{E}(C_{\max}) \leq O(\log m)2\omega$. □

Acknowledgement

This research benefited from the support of the Jacques Hadamard Mathematical Foundation(FMJH) Gaspard Monge Program for optimization and operations research, and the EDF as well as the ANR project ROBUST [ANR-16-CE40-0018].

References

1. A. Agra, M. Christiansen, R. Figueiredo, L. M. Hvattum, M. Poss, and C. Requejo. The robust vehicle routing problem with time windows. *Computers & OR*, 40(3):856–866, 2013.
2. M. A. Aloulou and F. D. Croce. Complexity of single machine scheduling problems under scenario-based uncertainty. *Operations Research Letters*, 36(3):338 – 342, 2008.
3. A. Ben-Tal and A. Nemirovski. Robust solutions of uncertain linear programs. *Oper. Res. Lett.*, 25(1):1–13, 1999.
4. D. Bertsimas and M. Sim. Robust discrete optimization and network flows. *Math. Program.*, 98(1-3):49–71, 2003.
5. D. Bertsimas and M. Sim. The price of robustness. *Operations Research*, 52(1):35–53, 2004.
6. P. Brucker. *Scheduling algorithms*, volume 3. Springer, 2007.
7. R. L. Daniels and P. Kouvelis. Robust scheduling to hedge against processing time uncertainty in single-stage production. *Management Science*, 41(2):pp. 363–376, 1995.
8. B. T. Denton, A. J. Miller, H. J. Balasubramanian, and T. R. Huschka. Optimal allocation of surgery blocks to operating rooms under uncertainty. *Operations research*, 58(4-part-1):802–816, 2010.
9. K. Goetzmann, S. Stiller, and C. Telha. Optimization over integers with robustness in cost and few constraints. In *Approximation and Online Algorithms - 9th International Workshop, WAOA 2011, Saarbrücken, Germany, September 8-9, 2011, Revised Selected Papers*, pages 89–101, 2011.
10. R. L. Graham, E. L. Lawler, J. K. Lenstra, and A. R. Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of discrete mathematics*, 5:287–326, 1979.
11. M. Grötschel, L. Lovász, and A. Schrijver. *Geometric Algorithms and Combinatorial Optimization*. Springer, Berlin, 1988.
12. E. Horowitz and S. Sahni. Exact and approximate algorithms for scheduling nonidentical processors. *J. ACM*, 23(2):317–327, 1976.
13. L. G. Khachiyan. Polynomial algorithms in linear programming. *USSR Computational Mathematics and Mathematical Physics*, 20(1):53–72, 1980.

14. O. Klopfenstein and D. Nace. A robust approach to the chance-constrained knapsack problem. *Oper. Res. Lett.*, 36(5):628–632, 2008.
15. O. Koné, C. Artigues, P. Lopez, and M. Mongeau. Event-based MILP models for resource-constrained project scheduling problems. *Computers & OR*, 38(1):3–13, 2011.
16. P. Kouvelis and G. Yu. *Robust discrete optimization and its applications*, volume 14. Springer Science & Business Media, 1997.
17. J. Lenstra, D. Shmoys, and E. Tardos. Approximation algorithms for scheduling unrelated parallel machines. *Mathematical Programming*, 46(1-3), 1990.
18. J. K. Lenstra, A. R. Kan, and P. Brucker. Complexity of machine scheduling problems. *Annals of discrete mathematics*, 1:343–362, 1977.
19. M. Mastrolilli, N. Mutsanas, and O. Svensson. Approximating single machine scheduling with scenarios. In *Approximation, Randomization and Combinatorial Optimization. Algorithms and Techniques*, pages 153–164. Springer Berlin Heidelberg, 2008.
20. W. Naji, M.-L. Espinouse, and V.-D. Cung. Towards a robust scheduling on unrelated parallel machines: A scenarios based approach. In H. A. Le Thi, T. Pham Dinh, and N. T. Nguyen, editors, *Modelling, Computation and Optimization in Information Systems and Management Sciences*, volume 360 of *Advances in Intelligent Systems and Computing*, pages 343–355. Springer International Publishing, 2015.
21. A. A. Pessoa, L. D. P. Pugliese, F. Guerriero, and M. Poss. Robust constrained shortest path problems under budgeted uncertainty. *Networks*, 66(2):98–111, 2015.
22. M. L. Pinedo. *Scheduling: Theory, Algorithms, and Systems*. Springer Publishing Company, Incorporated, 3rd edition, 2008.
23. M. Poss. Robust combinatorial optimization with variable budgeted uncertainty. *4OR*, 11(1):75–92, 2013.
24. M. Poss. Robust combinatorial optimization with variable cost uncertainty. *European Journal of Operational Research*, 237(3):836–845, 2014.
25. W. E. Smith. Various optimizers for single-stage production. *Naval Research Logistics Quarterly*, 3(1-2):59–66, 1956.
26. B. Tadayon and J. C. Smith. Algorithms and complexity analysis for robust single-machine scheduling problems. *J. Scheduling*, 18(6):575–592, 2015.
27. J. Yang and G. Yu. On the robust single machine scheduling problem. *Journal of Combinatorial Optimization*, 6(1):17–33, 2002.

A Proof of Proposition 1

Let us detail the inner maximization of (2) as

$$\begin{array}{lll} \sum_{i,j} \overline{p}_j q_i x_{ij} & + \quad \max & \sum_{i,j} \delta_j \hat{p}_j q_i x_{ij} \\ & \text{s.t.} & \sum_j \delta_j \leq \Gamma, \\ & & \delta_j \in \{0,1\}, \quad j = 1, \ldots, J. \end{array}$$

Removing the binary conditions on δ in the definition of U^Γ, one obtains a polytope whose extreme points are exactly the elements of U^Γ. Hence, we can consider the linear programming relaxation of the above problem, which is equal to the solution cost of its dual

$$\begin{array}{ll} \min & \Gamma\theta + \sum_j y_j \\ \text{s.t.} & \theta + y_j \geq \sum_i \hat{p}_j q_i x_{ij}, \quad j = 1, \ldots, J \\ & \theta, y \geq 0. \end{array}$$

Substituting y_j by $\max(0, \sum_i \hat{p}_j q_i x_{ij} - \theta)$, we can further reformulate (2) as

$$\min_{x \in \mathcal{X}, \theta \geq 0} \Gamma\theta + \sum_{i,j} \overline{p}_j q_i x_{ij} + \sum_j \max(0, \sum_i \hat{p}_j q_i x_{ij} - \theta). \tag{23}$$

The only step of our proof that differs from Theorem 1 is that, because the constraint $\sum_{i=1}^{I} x_{ij} = 1$ holds for each $j = 1, \ldots, J$, we can further reformulate (23) as

$$\min_{x \in \mathcal{X}, \theta \geq 0} \Gamma\theta + \sum_{i,j} \overline{p}_j q_i x_{ij} + \sum_j \sum_i x_{ij} \max(0, \hat{p}_j q_i - \theta). \tag{24}$$

Let us denote by $f_x(\theta)$ the objective function of (24). Then, observe that for any $x \in \mathcal{X}$, $f_x(\theta)$ is a piece-wise linear convex function that reaches its minimum at one of its kink points. The result then follows by observing that, for any $x \in \mathcal{X}$, the set of kink points of $f_x(\theta)$ is included into the set $\{\hat{p}_l q_k : k \in I, l \in J\}$.