Robust scheduling with budgeted uncertainty
Marin Bougeret, Artur Pessoa, Michael Poss

To cite this version:
Marin Bougeret, Artur Pessoa, Michael Poss. Robust scheduling with budgeted uncertainty. 2016. <hal-01345283>

HAL Id: hal-01345283
https://hal.archives-ouvertes.fr/hal-01345283
Submitted on 13 Jul 2016

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
Robust scheduling with budgeted uncertainty

Marin Bougeret · Artur Alves Pessoa · Michael Poss

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 polynomial algorithms and 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 $J$, a set of $m$ machines represented by $M$, and processing
times represented by the tuple \( p \), we look for a schedule \( \sigma \) of the jobs on the machines 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 \( \min_{\sigma \in S} f(\sigma, p) \).

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 different papers \([2, 6, 23]\) 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 does not admit a Polynomial Time Approximation Scheme (PTAS) either, unless \( \mathcal{P} = \mathcal{NP} \) (see \([17]\)). These negative results are not surprising since \([14]\) 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 \([18, 13]\), 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 \( \Gamma \), 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:

\[ U_\Gamma \equiv \left\{ p \in \mathbb{R}^n : p_j = \bar{p}_j + \delta_j \hat{p}_j, j \in \{1, \ldots, n\}, \delta_j \in \{0,1\}, j \in \{1, \ldots, n\}, \sum_{j=1}^n \delta_j \leq \Gamma \right\}. \]

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 \( U_\Gamma \) (denoted \( U_\Gamma \)-robust for short) belong to the complexity classes of their deterministic versions:

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

They provided a similar result for the approximation ratio of robust combinatorial optimization problems and [7] have extended these positive results to larger classes of \( U_\Gamma \)-robust combinatorial optimization problems, many of them relying on dynamic programming algorithms, see [1,12,21]. 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 \( U_\Gamma \)-robust counterpart; only an example of a weakly \( \mathcal{NP} \)-hard problem turning strongly \( \mathcal{NP} \)-hard in the \( U_\Gamma \)-robust case has been exhibited [19]. In addition to its theoretical tractability, \( U_\Gamma \)-robust problems can often be solved efficiently by applying the classical dualization approach to robust optimization [3]. This approach holds because set \( U_\Gamma \) can be described as the extreme points of a polytope described by a linear number of inequalities. The interest for set \( U_\Gamma \) is also motivated by its link with probabilistic constraints studied in [5,20]. In the context of combinatorial optimization problems with cost uncertainty, the results from [5,20] imply that \( \min_{x \in X} \max_{p \in U_\Gamma} p^T x \) provides a conservative approximation of minimizing the value-at-risk over \( X \), see also [21]. In spite of the tremendous success of set \( U_\Gamma \) in the robust combinatorial optimization literature, we are not aware of previous work studying the theoretical complexity of \( U_\Gamma \)-robust scheduling problems. This work starts filling this gap.

Recall the three-field notation \( \alpha|\beta|\gamma \) from [8] where \( \alpha \) describes the machine environment, \( \beta \) the job characteristics, and \( \gamma \) the objective function. In this first work on \( U_\Gamma \)-robust scheduling, we focus on the following classical scheduling problems. Let \( C_j(\sigma,p) \) denote the completion time of task \( j \) for schedule \( \sigma \) 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 \( f(\sigma, p) = \sum_{j \in J} w_j C_j(\sigma, p) \). We pay a particular attention to the case where \( w_j = 1 \) for each \( j \in J \), 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 $f(\sigma, p) = C_{\text{max}}(\sigma, p) = \max_{j \in J} C(\sigma, p)$. In the first case, processing job $j$ on machine $i$ is given by $p_{ij}$. 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 $P[|C_{\text{max}}|Q][|C_{\text{max}}|]$, and $R[|C_{\text{max}}|]$, respectively. We also consider the special case $Rm[|C_{\text{max}}|$ where the number of machines $m$ is not considered part of the instance.

We detail below our contributions more specifically and detail the structure of the paper. Let us extend Graham’s notation to $\alpha|\beta|U^T|\gamma$ to specify that the cost of any feasible schedule is obtained for the worst processing times in $U^T$.

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 $1|U_p|\sum C_j$ is polynomial by extending Theorem 1. Comparing with $\{2, 6, 23\}$, the result illustrates how $U^T$-robust scheduling can lead to more tractable problems than robust scheduling with arbitrary uncertainty sets. We show then that $1|U_p|\sum w_jC_j$ 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 $U^T$-robust counterpart. We also provide a special case solvable in polynomial time through an extension of Smith’s rule to the robust context. In Section 3 we show that $P[|U_p|C_{\text{max}}]$ is $3$-approximable and admits a PTAS if $\Gamma$ is constant. As it does not seem possible to directly use as a black box existing approximation algorithms (for example by looking only at $\sigma$ or $\bar{p}$ and applying a PTAS), we provide these results via an ad-hoc analysis. Section 4 is dedicated to $R[|U_p|C_{\text{max}}]$. We first show how the classical FPTAS of $[10]$ for $Rm[|C_{\text{max}}|$ can be adapted for $Rm[|U_p|C_{\text{max}}]$. 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.

\textbf{Example for $1|U_p|\sum C_j$} Let us consider the following instance of $1|U_p|\sum C_j$ with $n = 3$ jobs and $\Gamma = 1$. Let $\overline{p}_1 = 3, \overline{p}_2 = 1, \overline{p}_3 = 2, \bar{p}_1 = 1, \bar{p}_2 = 10, \bar{p}_3 = 5$. By definition we have $U^T = \{(3, 1, 2), (4, 1, 2), (3, 11, 2), (3, 1, 7)\}$. 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 $F(\sigma) = 41$.

\textbf{Example for $P[|U_p|C_{\text{max}}$} Let us now consider the following instance of $P[|U_p|C_{\text{max}}$ with $n = 4$ jobs, $m = 2$ machines and $\Gamma = 1$. Let $\overline{p}_1 = 2, \overline{p}_2 = 2, \overline{p}_3 = 3, \overline{p}_4 = 3, \bar{p}_1 = 1, \bar{p}_2 = 2, \bar{p}_3 = 12, \bar{p}_4 = 8$. By definition we have $U^T = \{(5, 3, 2, 2), (6, 3, 2, 2), (5, 5, 2, 2), (5, 3, 14, 2), (5, 3, 2, 10)\}$. Notice that a schedule is completely characterized by the set of jobs scheduled on each machine. For $\sigma$ 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 \( \sigma \) and \( \sigma_i \subseteq J \) denotes a schedule restricted to machine \( i \). An optimal schedule is denoted by \( \sigma^* \) and its value is denoted by \( \text{opt} \). 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 problem, yet it is \( \mathcal{NP} \)-hard in the weak sense for arbitrary uncertainty sets \( U \), even for two scenarios [23]. In contrast, we show below that the \( U^\Gamma \)-robust version of the problem can be solved in polynomial time.

Our approach applies an extension of Theorem 1 to problem \( 1||U^\Gamma_p|\sum C_j \).

Let \( x_{ij} \) be equal to 1 iff job \( j \) is scheduled in position \( i \). Problem \( 1||U^\Gamma_p|\sum C_j \) can be cast as

\[
\min_{x \in \mathcal{X}} \max_{p \in U^\Gamma} \sum_{(i,j) \in [n]^* \times \mathcal{J}} p_j(n + 1 - i)x_{ij} : \sum_{i \in [n]^*} x_{ij} = 1, j \in \mathcal{J}, \sum_{j \in \mathcal{J}} x_{ij} = 1, i \in [n]^*.
\]

(1)

Observation 1 Theorem 1 cannot be applied to problem (1) because its cost function is defined by a product of parameters \( p, q \) where only \( p \in U^\Gamma \).

We now provide an extension of Theorem 1, which encompasses problem (1). Theorem 2 Let \( \mathcal{X} \subseteq \{0, 1\}^{J \times J} : \sum_{i=1}^{I} x_{ij} = 1, j = 1, \ldots, J \} \) and let \( q \in \mathbb{R}^I \) and \( p \in U^\Gamma \) be cost vectors. The optimal solution to problem

\[
\min_{x \in \mathcal{X}} \max_{p \in U^\Gamma} \sum_{i,j} p_jq_ix_{ij}
\]

(2)

can be obtained by solving the problems \( \min_{x \in \mathcal{X}} \sum_{i,j} (\bar{p}_{ij} + \hat{p}_{ij})q_ix_{ij} \) and \( \min_{x \in \mathcal{X}} \sum_{i,j} (\bar{p}_{ij} + \tilde{p}_{kl}^{ij})q_ix_{ij} \), for each \( k \in I, l \in J \), where \( \tilde{p}_{kl}^{ij} = \max(0, \hat{p}_{ij} - \bar{p}_{kl}) \).

Proof The proof follows closely the lines of the proof of Theorem 1 from [4]. Let us detail the inner maximization of (2) as

\[
\sum_{i,j} \bar{p}_j q_i x_{ij} + \max \sum_{i,j} \delta_j \hat{p}_j q_i x_{ij}
\]

s.t. \( \sum_j \delta_j \leq \Gamma \), \( \delta_j \in \{0, 1\}, \ j = 1, \ldots, J \).
Removing the binary conditions on $\delta$ in the definition of $U^\Gamma$, one obtains a polytope whose extreme points are exactly the elements of $U^\Gamma$. Hence, we can consider the linear programming relaxation of the above problem, which is equal to the solution cost of its dual

$$\min \quad \Gamma \theta + \sum_j y_j$$

$$\text{s.t.} \quad \theta + y_j \geq \sum_i \hat{p}_j q_i x_{ij}, \quad j = 1, \ldots, J$$

$$\theta, y \geq 0.$$

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

$$\min_{x \in X, \theta \geq 0} \quad \Gamma \theta + \sum_{i,j} \overline{p}_j q_i x_{ij} + \sum_j \max(0, \sum_i \hat{p}_j q_i x_{ij} - \theta). \quad (3)$$

The crucial step of our proof (which differs from Theorem 1) is that, because the constraint $\sum_{i=1}^J x_{ij} = 1$ holds for each $j = 1, \ldots, J$, we can further reformulate (3) as

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

Introducing $r = (i,j)$ and renaming variable $x_{ij}$ as $z_r$, products $\overline{p}_j q_i$ and $\hat{p}_j q_i$ as $c_r$ and $d_r$, respectively, the rest of the proof is identical to the proof of Theorem 1 from [4].

Applying Theorem 2 to (1), we obtain that $1||U^\Gamma_p|\sum C_j$ 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 $\overline{p}_j + \hat{p}^{kl}$ break the structure of $1||U^\Gamma_p|\sum C_j$, i.e., the deterministic problems with cost vector $\overline{p} + \hat{p}^{kl}$ 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 [22] (i.e., scheduling jobs by non-decreasing $\frac{w_j}{p_j}$). However, it does not seem easy to extend that simple rule to the robust problem $1||U^\Gamma_p|\sum w_j C_j$. 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 $\Gamma$. For that, we need the following two lemmas.

**Lemma 1** Given $X \subset J$ such that $\frac{w_j}{p_j} \leq \frac{w_{\ell}}{p_{\ell} + \hat{p}_{\ell}}$, $\forall j \in X$, and $\ell \in J \setminus X$, in any optimal solution for $1||U^\Gamma_p|\sum w_j C_j$ the jobs in $X$ are the last $|X|$ in the schedule.
Theorem 3 There is a polynomial reduction from $1||\sum w_jC_j$ where two consecutive jobs $j$ and $\ell$ have
\[
\frac{w_j}{p_j} < \frac{w_\ell}{p_\ell + \bar{p}_\ell}.
\]

Let $c^*(p)$ denote the solution cost for the specific vector $p \in U^r$ and $c^*$ be the solution cost for worst deviations; that is, $c^* = \max_{p \in U^r} c^*(p)$. By swapping $j$ and $\ell$ in $\sigma^*$, we obtain an alternative schedule $\sigma'$ with cost denoted $c'$, which satisfies
\[
c' = \max_{p \in U^r} (c^*(p) + p_\ell w_j - p_j w_\ell) \\
\leq \max_{p \in U^r} c^*(p) + \max_{p \in U^r} (p_\ell w_j - p_j w_\ell) \\
\leq c^* + (\bar{p}_\ell + \bar{p}_j)w_j - \bar{p}_j w_\ell.
\]

From (4) and (5), we obtain that $c' < c^*$, which contradicts the optimality of $\sigma^*$.

Lemma 2 There exists an optimal solution for $1||\sum w_jC_j$ where, for any two jobs $j$ and $\ell$, with $p_j = p_\ell$, $w_j = w_\ell$, and $\bar{p}_j < \bar{p}_\ell$, $j$ is scheduled before $\ell$.

Proof Let $j$ and $\ell$ be two jobs satisfying the conditions of this proposition such that $\ell$ precedes $j$ in an optimal solution $\sigma^*$. We show that swapping $\ell$ and $j$ does not increase the robust cost $c^*$ of $\sigma^*$. Let $\sigma'$ be the resulting schedule. Let also $W_k$ be the sum of weights of all jobs that do not precede $k$ in $\sigma^*$, for all $k \in J$. Clearly $W_j < W_\ell$. Moreover, the total cost due to mean processing times is the same for $\sigma^*$ and $\sigma'$, and the total cost due to deviations is calculated by selecting the $\Gamma$ jobs with maximum $\bar{p}_k W_k$ among all $k \in J$, and summing up these values. After the swap, $\bar{p}_j W_j$ and $\bar{p}_\ell W_\ell$ are replaced by $\bar{p}_\ell W_j$ and $\bar{p}_j W_\ell$, and the remaining values are kept unchanged. Since $W_j < W_\ell$ and $\bar{p}_j < \bar{p}_\ell$, we have that $\bar{p}_j W_j + \bar{p}_\ell W_\ell < \bar{p}_\ell W_j + \bar{p}_j W_\ell$. 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 $1||\sum w_jC_j$, denoted by $(1||\sum w_jC_j, K)_{dec}$, asks for a schedule whose robust cost is not greater than a given integer $K$.

Theorem 3 There is a polynomial reduction from $k$-PARTITION to $(1||\sum w_jC_j, K)_{dec}$ with $\Gamma = N - 1$. 

Proof We prove the proposition by contradiction. Assume that there is an optimal solution $\sigma^*$ for $1||\sum w_jC_j$ where two consecutive jobs $j$ and $\ell$ have
\[
\frac{w_j}{p_j} < \frac{w_\ell}{p_\ell + \bar{p}_\ell}.
\]
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 $j = (k+1)N, \ldots, (k+2)N - 2 \leq n$, 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.

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

$$A^\sigma_j = \sum_{\ell = 1}^{kN} a_{\ell}$$

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 $\sigma$ for the created instance of $(1|U^\Gamma|\sum w_jC_j, K)_{dec}$ into six terms:

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

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

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

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

- 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
\[ c_4 = \sum_{j=1}^{kN} \sum_{\ell=(k+1)N}^{(k+2)N-2} p_j w_\ell \]
\[ = 2NA(N-1) - 2 \sum_{j=(k+1)N}^{(k+2)N-2} A_j^\sigma; \]

– the cost from separating jobs excluding deviations, given by
\[ c_5 = \sum_{j=(k+1)N}^{(k+2)N-2} \sum_{\ell=(k+1)N}^{(k+2)N-2} p_j w_\ell \]
\[ = N(N-1) + (N-1)^2 + \sum_{j=(k+1)N}^{(k+2)N-2} A_j^\sigma; \]

– the cost due to deviations from both the separating jobs and the tail jobs, given by
\[ c_6 = \sum_{j=kN+1}^{(k+2)N-2} \sum_{\ell=(k+1)N}^{(k+2)N-2} (p_j^\sigma - \bar{p}_j) w_\ell \]
\[ = \max_{j=(k+1)N} \left\{ 4NA, \sum_{\ell=(k+1)N}^{(k+2)N-2} \bar{p}_j w_\ell \right\} \]
\[ = \max_{j=(k+1)N} \left\{ 4NA - \frac{4NA}{W_j + \beta(j)A(W_j + A_j^\sigma)} \right\} , \]

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 \( \sigma \). All remaining terms are constant. Summing up the non-constant terms, we obtain
\[ \bar{c}(\sigma) = \sum_{j=(k+1)N}^{(k+2)N-2} \max \left\{ 4NA - A_j^\sigma, \frac{4NA}{W_j + \beta(j)A(W_j + A_j^\sigma)} - 1 \right\} . \]

Assuming that \( A > 3 \), we have that \( \frac{4NA}{W_j + \beta(j)A(W_j + A_j^\sigma)} > 2 \). Hence, the value of \( \bar{c}(\sigma) \) is minimized (and thus the total cost) when \( \beta(j)A = A_j^\sigma \), 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 \( A_j^\sigma \) in the two arguments of the maximum function, \( \bar{c}(\sigma) \) 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 \( \text{K-PARTITION} \) 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 $2(N - 1) \sum_{j \in J} w_j$. This yields a solution of cost $K - (N - 1) \sum_{j \in J} w_j$ in the case of a positive answer to $k$-PARTITION, and no solution of cost less than $K + (N - 1) \sum_{j \in J} w_j$ 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 $(N - 1) \sum_{j \in J} w_j$, 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** \((1||U \Gamma p|\sum w_j C_j, K)_{dec}\) is $\mathcal{NP}$-complete in the weak sense for $\Gamma = 1$ and strongly $\mathcal{NP}$-complete when $\Gamma$ 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 3.

We finish the section by proving that a special case of $1||U \Gamma p|\sum w_j C_j$ can be solved in polynomial time using a natural extension of Smith’s rule. Let $i, j$ be any pair of jobs in $J$ and consider the following property:

\[
\frac{p_i}{w_i} \leq \frac{p_j}{w_j} \iff \frac{\hat{p}_i}{w_i} \leq \frac{\hat{p}_j}{w_j}. \tag{6}
\]

When property (6) holds, Smith’s rule can be extended by considering $\hat{p}$ or $p$ indifferently. Yet the resulting algorithm is not optimal in general as we show on the simple instance with $\Gamma = 1$, $p_1 = (1, 11)$, $p_2 = (1, 10)$ (where $p_j = (\bar{p}_j, \hat{p}_j)$), $w_1 = 2$ and $w_2 = 1$. We see that Smith’s rule would schedule job 1 first and job 2 second, which costs 37, while scheduling job 2 first and job 1 second costs 35, which is optimal. Yet, we can prove that the rule is optimal when the processing times $\hat{p}_j$ are agreeable with the weights $w_j$, i.e.

\[
w_i < w_j \Rightarrow \hat{p}_i \geq \hat{p}_j, \tag{7}
\]

where the terminology agreeable has been borrowed from [15]. We see that the above example, for which Smith’s rule is suboptimal, does not satisfy (7).

We also mention that satisfying conditions (6) and (7) does not imply that the mean processing times $\bar{p}_j$ are agreeable with the weights $w_j$, consider for instance the case with $w_1 = 1$, $w_2 = 2$, $\bar{p}_1 = 3$, $\bar{p}_2 = 2$, $\hat{p}_1 = 2$, and $\hat{p}_2 = 3$. Let us first recall a well-known property of convex functions.

**Lemma 3** Let $f$ be a convex function and $\delta > 0$. It holds that $F(x) := f(x + \delta) - f(x)$ is non-decreasing.

**Theorem 4** If properties (6) and (7) are satisfied, then it is optimal for problem $1||U \Gamma p|\sum w_j C_j$ to schedule jobs according to the non-decreasing value of $\frac{p_j + \hat{p}_j}{w_j}$. 

Proof We formulate problem 1||U^Γ|∑w_jC_j as follows. Let x_{ij} be a binary variable that is equal to 1 iff job i is scheduled prior to job j. The formulation is

\[
\min \left( \max_{p \in \mathcal{U}^\Gamma} \sum_{i \in J} \sum_{j \in J} p_i w_j x_{ij} \right)
\]

s.t. \quad x_{ij} + x_{ji} = 1, \quad i \neq j \in J \quad (8)
\quad x_{ij} + x_{jk} \leq x_{ik} + 1, \quad i, j, k \in J \quad (9)
\quad x_{ii} = 1, \quad i \in J \quad (10)
\quad x_{ij} \in \{0, 1\}.

Let \mathcal{X} denote the set of binary vectors that satisfy (8)–(10). As in the proof of Theorem 2, we can dualize the inner maximization problem to obtain

\[
\min \Gamma \theta + \sum_{i \in J} y_i + \sum_{i \in J} \sum_{j \in J} \hat{p}_i w_j x_{ij}
\]

s.t. \quad \theta + y_i \geq \hat{p}_i \sum_{j \in J} w_j x_{ij}, \quad i \in J
\quad x \in \mathcal{X}
\quad \theta, y \geq 0,

which can be further rewritten

\[
\min \Gamma \theta + \sum_{i \in J} \hat{p}_i \max \left( 0, \sum_{j \in J} w_j x_{ij} - \frac{\theta}{\hat{p}_i} \right) + \sum_{i \in J} \sum_{j \in J} \hat{p}_i w_j x_{ij}
\]

s.t. \quad x \in \mathcal{X}
\quad \theta \geq 0,

where we assumed that all components of \( \hat{p} \) are positive (we show at the end of the proof how to handle components of \( \hat{p} \) equal to 0).

We have shown that solving 1||U^Γ|∑w_jC_j amounts to solve a scheduling problem with an objective function composed of three terms. The first term, \( \Gamma \theta \), does not depend on the schedule and can be forgotten for now. The third term, \( \sum_{i \in J} \sum_{j \in J} \hat{p}_i w_j x_{ij} \), consists in the weighted sum of completion times defined by the vector of weights \( w \) and the vector of processing times \( \hat{p} \). Thank’s to Smith’s rule, it is optimal to order the jobs by non-decreasing value of \( \frac{\hat{p}_j}{w_j} \). We show below using a swapping argument that it is optimal for the second term to order the jobs by non-decreasing value of \( \frac{\hat{p}_j}{w_j} \). Therefore, because property (6) holds, the second and the third terms of the objective function yield consistent orders for the jobs (if only one of the terms has a tie, the order is dictated by the other term). The result holds because the orders are independent of the value of \( \theta \).
Consider the second term

$$\sum_{i \in J} \hat{p}_i \max \left( 0, \sum_{j \in J} w_j x_{ij} - \frac{\theta}{\hat{p}_i} \right).$$

(11)

The objective function described by (11) can be interpreted as the weighted tardiness defined by the vector of weights \( \hat{p} \), the vector of processing times \( w \), the vector of deadlines \( \Theta \), and where the completion time \( W_j \) of each job \( j \) sums up the processing times \( w_i \) of all jobs scheduled after job \( j \), including job \( j \). Consider an optimal schedule for (11) where \( i \) is processed just after \( j \) and assume that

$$\frac{\hat{p}_i}{w_i} \leq \frac{\hat{p}_j}{w_j}$$

(12)

holds. We show below that swapping \( i \) and \( j \) does not deteriorate the objective function. First, we claim that \( w_i \geq w_j \). If not, (7) implies that \( \hat{p}_i \geq \hat{p}_j \), which is in contradiction with (12). Assume then that \( w_i > w_j \) (the case \( w_i = w_j \) can be treated similarly). Again, property (7) yields \( \hat{p}_i \leq \hat{p}_j \), and we also have the relation \( \frac{\theta}{\hat{p}_i} \geq \frac{\theta}{\hat{p}_j} \). Let \( C \) denote the cost due to jobs \( i \) and \( j \) in the original schedule, \( C' \) denote the cost after swapping \( i \) and \( j \), and let \( W \) be equal the sum of the weights of the jobs scheduled after \( i \) and \( j \). Function (11) implies that

$$C = \hat{p}_i \max \left( 0, W + w_i - \frac{\theta}{\hat{p}_i} \right) + \hat{p}_j \max \left( 0, W + w_j - \frac{\theta}{\hat{p}_j} \right),$$

and

$$C' = \hat{p}_i \max \left( 0, W + w_i + w_j - \frac{\theta}{\hat{p}_i} \right) + \hat{p}_j \max \left( 0, W + w_j - \frac{\theta}{\hat{p}_j} \right),$$

and we want to prove that \( C' \leq C \). Let us introduce \( \Theta_i = W - \frac{\theta}{\hat{p}_i} \) and \( \Theta_j = W - \frac{\theta}{\hat{p}_j} \); the relation \( \frac{\theta}{\hat{p}_i} \geq \frac{\theta}{\hat{p}_j} \) yields \( \Theta_i \leq \Theta_j \). Furthermore, let us denote \( \max(0, \cdot) \) as the function \( f(\cdot) \), and remark that \( f \) is convex and non-decreasing.

We have that

$$\frac{\hat{p}_j}{\hat{p}_i} (f(\Theta_j + w_i + w_j) - f(\Theta_j + w_j)) - f(\Theta_i + w_i + w_j) - f(\Theta_i + w_i)) \geq (f(\Theta_j + w_i + w_j) - f(\Theta_j + w_j)) - (f(\Theta_i + w_i + w_j) - f(\Theta_i + w_i)) \geq 0,$$

(13)

(14)

(15)

where (13) holds because \( \frac{\hat{p}_j}{\hat{p}_i} \geq 1 \), (14) holds because of Lemma 3 and \( \Theta_i \leq \Theta_j \), and (15) holds because \( f \) is non-decreasing and \( w_i > w_j \).

Notice finally that if \( \hat{p}_i = 0 \) for some job \( i \), the associated term in (11) is equal to 0 and the job can be scheduled arbitrarily for that objective function. \( \square \)
3 Minimizing makespan on identical machines

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

We say that an algorithm $A$ is a $\rho$-dual approximation if for any $\omega$ and instance $\mathcal{I}$, either $A(\mathcal{I}, \omega)$ builds a schedule $\sigma$ such that $C(\sigma) \leq \rho \omega$ or fails, which implies then that $\omega < \text{opt}$. Notice that any $\rho$-dual approximation can be converted to a $\rho$-approximation algorithm by performing a binary search on $\omega$ to find the smallest $\omega$ 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 $\hat{\sigma}_1$, apply a PTAS on a second instance where $p_j = \hat{p}_j$ to get a schedule $\hat{\sigma}_2$, and try to merge $\hat{\sigma}_1$ and $\hat{\sigma}_2$ 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 $P||U^\rho_p|C_{max}$, and prove the following theorem.

**Theorem 5** For any $\epsilon > 0$, if for any $j \in J$, $\overline{p}_j \leq \epsilon \omega$ and $\hat{p}_j \leq \epsilon \omega$, then $A$ is a min$(2 + 2\epsilon, 3)$ dual approximation algorithm for $P||U^\rho_p|C_{max}$. This implies that $P||U^\rho_p|C_{max}$ 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 $\omega$ 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 $\omega$ if there remain some jobs after filling $m$ machines. If (as in Theorem 5) for any $j \in J$, $\overline{p}_j \leq \epsilon \omega$ and $\hat{p}_j \leq \epsilon \omega$, this algorithm would not exceed $(1 + 2\epsilon)\omega$, thus improving over Theorem 5. However, this algorithm is not correct, as the existence of a $\sigma$ 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 would create $\sigma_1 = \{1, 2\}$, $\sigma_2 = \{3, 4, 5\}$ and rejects as
Algorithm 1 Algorithm A

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

**Observation 2** For any σ, C(σ) ≤ ω ⇒ ́p(J) + ́p(J) ≤ mω.

**Lemma 4** If for any j ∈ J, ́p_j ≤ ω and ́p_j ≤ ω, then for any i, C(σᵢ) ≤ \min((2 + 2)ω,3ω)

**Proof** In the worst case, before adding the last job j in the interior while loop we had ́p(X) = ω and ́p(J) = ω, and thus C(σᵢ) ≤ 2ω + ́p_j + ́p_j with ́p_j + ́p_j ≤ \min(2ω,ω) (if there is a job with ́p_j + ́p_j > ω, we can immediately reject ω).

**Lemma 5** If A fails, then opt > ω.

**Proof** Let us suppose that A fails and suppose by contradiction that opt ≤ ω. We say that machine i is of type 1 if ́p(J) > ω, and is of type 2 otherwise. Notice that a schedule on a machine of type 1 contains at most I jobs (as jobs are added by non-increasing ́p_j), and a schedule σᵢ on a machine of type 2 verifies ́p(J) > w. Let M₁ be the set of machines of type 1, and let J₁ be the set of jobs scheduled by A in machines M₁. Let M₂ and J₂ be defined in the same way. We have

- ́p(J₂) > |M₂|ω by definition of type 2
- ́p(J₁) > |M₁|ω by definition of type 1
- ́p(J) ≥ ́p(J₁)

Let us prove the last item. Notice first that for any schedule σᵢ of J₁ on m machines such that C(σᵢ) ≤ ω, ́p(σᵢ) = ́p(J₁). Indeed, let i be the last machine in M₁ and let x = |σᵢ|. Notice that as we select the jobs by non-increasing order of ́p_j in the interior while loop, σᵢ contains the x smallest jobs (in terms of ́p_j) of J₁. As i is type 1 we get ́p(J) > ω, and we deduce that in any schedule of J₁ that fits in ω, there is at most x ≤ I jobs on every machine. Thus, all jobs deviate in σᵢ, and ́p(σᵢ) = ́p(J₁).

C(σᵢ) > ω for any i and not all jobs are scheduled, whereas there exists a schedule σᵢ = {1, 5}, σᵢ = {2, 3, 4, 6} that fits in ω. This explains the design of Algorithm 1. The validy of the algorithm is shown in the rest of this section.
Then, notice that \( \hat{p}(\sigma^*) \geq \hat{p}(\sigma^*_{J_1}) \) where \( \sigma^*_{J_1} \) is the schedule we obtain by starting from \( \sigma^* \) and only keeping jobs of \( J_1 \) on each machine (and removing idle time). Thus, as \( \text{opt}_{\sigma_{J_1}} \) is a schedule of \( J_1 \) on \( m \) machines that fits in \( \omega \), we know that \( \hat{p}(\sigma^*_{J_1}) = \hat{p}(J_1) \), concluding the proof of the last item.

Thus, we get \( \bar{p}(J) + \hat{p}(\sigma^*) \geq \bar{p}(J_2) + \hat{p}(J_1) > m \omega \), and thus according to Observation 2 we deduce \( \text{opt} > \omega \), a contradiction. \( \square \)

Lemmas 4 and 5 directly imply Theorem 5.

3.2 Fixed \( \Gamma \): PTAS

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

The key observations leading to the algorithm can be summarized as follows. Small deviations (\( \hat{p}_j \leq \delta \omega \)) lead to an additive error of \( \Gamma \delta \omega \), which can be neglected when \( \Gamma \) is constant (see Observation 5 below). Large deviations (\( \hat{p}_j > \delta \omega \)) 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 \( J \) into \( J = \hat{B} \cup \bar{B} \cup \tilde{B} \cup S \), where \( \hat{B} = \{ j | \hat{p}_j > \delta \omega \text{ and } \bar{p}_j > \delta \omega \} \), \( \bar{B} = \{ j | \hat{p}_j \leq \delta \omega \text{ and } \bar{p}_j > \delta \omega \} \), \( \tilde{B} = \{ j | \hat{p}_j > \delta \omega \text{ and } \bar{p}_j \leq \delta \omega \} \), and \( S = \{ j | \hat{p}_j \leq \delta \omega \text{ and } \bar{p}_j \leq \delta \omega \} \). 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 \( \bar{p}_j \in [\omega \delta (1 + \delta)^{r-1}, \omega \delta (1 + \delta)^r] \) for some \( r \in [k] \) then \( \bar{p}_j' = \omega \delta (1 + \delta)^r \) (otherwise we define \( \bar{p}_j' = \bar{p}_j \)), and if \( \hat{p}_j \in [\omega \delta (1 + \delta)^{r-1}, \omega \delta (1 + \delta)^r] \) 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 \( \hat{B} \) has been rounded twice (i.e., \( \hat{p}_j \) and \( \bar{p}_j \) have been rounded), whereas a job \( j \) in \( \bar{B} \) or \( \tilde{B} \) 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 \( \bar{p}_j'' = 0 \), and otherwise we define \( \bar{p}_j'' = \bar{p}_j' \).

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

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

**Observation 5** From any schedule \( \sigma'' \) of \( \mathcal{I}'' \), we can deduce a schedule \( \sigma' \) 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 negligible amount by setting \( \delta \) sufficiently small (\( \approx \frac{\epsilon}{\Gamma} \)) as \( \Gamma \) 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 \( \hat{B}'', \bar{B}'', \tilde{B}'', \) and \( S'' \). Notice that:
– for any job \( j \in \tilde{B}'' \), \( p_j'' \) can take at most \( k \) different values;
– for any job \( j \in \tilde{B}'' \), \( p_j'' \) can take at most \( k \) different values, and \( p_j'' = 0 \);
– for any job \( j \in \tilde{B}'' \), \( p_j'' \) can take arbitrary values (but below \( \delta \omega \)), and \( p_j'' \) can take at most \( k \) different values;
– for any job \( j \in S'' \), \( p_j'' \) can take arbitrary values (but below \( \delta \omega \)), and \( p_j'' = 0 \).

**Theorem 6** There exists a polynomial \((1 + \Gamma \delta)\)-dual approximation algorithm for \( T'' \).

The key elements of the algorithm used to prove Theorem 6 follow.

**Definition 2** Given a schedule \( \sigma''_i \) of a machine \( i \) in \( T'' \), let us define \( \tilde{\sigma}''_i \), the partial profile associated to \( \sigma''_i \) as follows. For any \( r_1, r_2 \in [k] \), let \( \hat{\pi}_{i,r_1,r_2} = |\{ j \in \sigma''_i \cap \tilde{B}'' | p_j'' = \omega \delta (1 + \delta)^{r_1} \text{ and } p_j'' = \omega \delta (1 + \delta)^{r_2} \}| \), let \( \pi_{i,r_1} = |\{ j \in \sigma''_i \cap \tilde{B}'' | p_j'' = \omega \delta (1 + \delta)^{r_1} \}| \), and let \( \hat{n}_{i,r_1} = |\{ j \in \sigma''_i \cap \tilde{B}'' | p_j'' = \omega \delta (1 + \delta)^{r_1} \}| \). We define \( \tilde{\sigma}''_i = (t_{i,1}, t_{i,2}) \), where \( t_{i,1} = (\hat{\pi}_{i,1}, r_1 \in [k], r_2 \in [k]) \), \( t_{i,2} = (\pi_{i,1}, r_1 \in [k]) \), and \( t_{i,1} \) is defined in Observation 6, we iterate on how \( \tilde{\sigma}''_i \) for the jobs in \( \tilde{B}'' \) that deviate (which are the \( \Gamma \) jobs with largest value \( p_j'' \)). 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 \( \hat{\pi}_{i,1,r_2} \leq \frac{1}{3} \) and \( \pi_{i,r_1} \leq \frac{1}{3} \) (as all these jobs have \( p_j'' \geq \delta \omega \)), 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 \( k' = \frac{k^2}{\delta} \frac{k}{\delta} \Gamma k \).

Let us define a polynomial algorithm \( A \) that either schedules \( T'' \) in \((1 + \Gamma \delta)\omega \) or fails, implying that \( \omega < \text{opt}(T'') \). Let \( \sigma^* \) be an optimal solution of \( T'' \). 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 \( \sigma^* \) as an input. For any machine, let \( p_i^* \) be the size of the smallest deviating job on machine \( i \). More formally, using \( r^{\min}_{i} = \min \{ r \leq \text{opt}(T'' \backslash S'') \text{ such that } \hat{n}_{i,r} 
eq 0 \} \), we define \( p_i^* = \omega \delta (1 + \delta)^{r^{\min}_{i}} \). Let us assume w.l.o.g. that \( p_i^* \leq p_{i+1}^* \). In Algorithm 2, we explain in details how \( A \) creates a schedule \( \sigma'' \) of jobs of \( T'' \) given this partial profile. We recall that for any integer \( x \), \( [x]^* = [x] \setminus \{0\} = \{1, \ldots, x\} \).

**Observation 7** \( \tilde{p}^\Gamma(\sigma''_i) = \tilde{p}^\Gamma(\sigma^*_i) \). Indeed, if two schedules \( \sigma''^{(1)} \) and \( \sigma''^{(2)} \) of \( T'' \) have the same partial profile, then \( \tilde{p}^\Gamma(\sigma''^{(1)}) = \tilde{p}^\Gamma(\sigma''^{(2)}) \).

For any \( i \), let \( \sigma^*_{i,big} = \sigma^*_i \backslash S'' \). Let us also define \( \mathcal{M}_1 \cup \cdots \cup \mathcal{M}_m' \) as the partition of \([m]^* \) of minimal cardinality \( m' \) such that \( p_i^* = p_i^* \) for each pair \( i, j \in \mathcal{M}_k \) and each \( g \in [m]^* \) (for example with \( m = 4 \) if \( p_1^* = 10, p_2^* = 15, p_3^* = 15 \) and \( p_4^* = 20 \), 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 \( p_i^* \) for any \( i \in \mathcal{M}_g \).
Algorithm 2 \(A(\mathcal{I}'', \hat{\sigma}'', \omega)\)

**Phase 1** A schedules jobs in \(X_1 = \hat{B}' \cup \hat{B}''\) by scheduling all jobs of \(\hat{B}'\) and \(\hat{B}''\) according to \(\hat{\sigma}''\). Notice that we have \(|X_1| = \sum_{r \in [m], r_1 \in [k], r_2 \in [k]} \pi_i r_1 r_2 + \sum_{r \in [m], r_1 \in [k]} \pi_i \).

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

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

**Lemma 6** If \(C(\hat{\sigma}'') \leq \omega\), then during Phase 3, for any \(g \leq m'\), after scheduling the last machine of \(M_g\), we have \(|p(\bigcup_{i \in M_1, \ldots, \bigcup M_g} \sigma_{[i]gq}) | \leq |p(\bigcup_{i \in M_1, \ldots, \bigcup M_g} \sigma_i'')|\).

In informal terms, the total (non-deviating) processing time scheduled by \(A\) on machines from \(M_1 \cup \ldots \cup M_g\) 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 \(i \in M_g\) such that \(A\) schedules all jobs of \(\hat{A}_i\) on \(i\). This implies that at the end of Phase 3, \(A\) did not schedule any other job from \(X_3\) on machines \(\{i+1, \ldots, i_{\text{max}}\}\) where \(i_{\text{max}}\) is the last machine of \(M_g\).

Let us partition \(Z'' = \bigcup_{i \in M_1, \ldots, \bigcup M_g} \sigma_{[i]q}''\) into \(\hat{Z} = Z'' \cap \hat{B}''\), \(\hat{Z} = Z'' \setminus \hat{B}''\), and \(\hat{Z} = Z'' \setminus \hat{B}''\). Notice that \(\hat{Z} = Z'' \setminus \hat{B}''\), and define the same partition for \(Z = \bigcup_{i \in M_1, \ldots, \bigcup M_g} \sigma_i''\).

Now, we will prove that \(\hat{Z} = \hat{B}'' \setminus \hat{Z} \subseteq Z'' \setminus \hat{Z}\) (which together with the previous inequality implies the desired result \(p(\hat{Z}) \leq p(\hat{Z})\)). To prove this, it is sufficient to remark that any \(j \in \hat{Z} \cap J_{\leq \hat{p}''}\) is either in \(\hat{A}_i\) (implying immediately the claim) or in \(X_2\) (implying \(j\) was scheduled in Phase 2).

In the last case, observe that whenever Phase 2 schedules \(j\) on a machine \(i'\), then \(\hat{p}_j'' \geq \hat{p}_j'\), and thus \(j\) must have been scheduled by Phase 2 on a machine \(i' \leq i_{\text{max}}\), and thus \(j \in \hat{Z} \setminus J_{\leq \hat{p}''}\).

It remains to analyze the case where for any \(i \in M_g\), A stops filling \(i\) as \(C(\hat{\sigma}''') > \omega\). As \(\hat{p}''(\sigma_i''') = \hat{p}''(\sigma_i'')\), we have \(|p(\sigma_i''')| \geq |p(\sigma_i''')|\).

**Corollary 1** (of Lemma 6) If \(A\) fails after Phase 3, then \(\omega < C(\hat{\sigma}'')\).
Lemma 8

If \( C(\sigma^*) \leq \omega \). Applying Lemma 6 to the last group \( \mathcal{M}_m' \), we get \( \overline{p}(\mathcal{I}' \setminus \mathcal{S}'') = \overline{p}(\bigcup_{\sigma \in \mathcal{M}_m \cup \mathcal{M}_m'} \sigma_{\text{big}}) \leq \overline{p}(\bigcup_{\sigma \in \mathcal{M}_m \cup \mathcal{M}_m'} \sigma''_q) = \overline{p}(X_1 \cup X_2 \cup X_3) \). If \( A \) fails after Phase 3, it means that \( X_1 \cup X_2 \cup X_3 \subset \mathcal{I}' \setminus \mathcal{S}'' \), and we get \( \overline{p}(\mathcal{I}' \setminus \mathcal{S}'') > \overline{p}(X_1 \cup X_2 \cup X_3) \), a contradiction.

\[ \square \]

Lemma 7

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

\[ \begin{proof}
\text{According to Observation 7 we have } \hat{p}^*(\sigma'') = \hat{p}^*(\sigma^*). \text{ If } A \text{ fails after Phase 4, we have } C(\sigma''_q) > \omega \text{ for any } i, \text{ and thus } \overline{p}(\mathcal{I}'') > mw - \hat{p}^*(\sigma'') = mw - \hat{p}^*(\sigma^*), \implies \omega < C(\sigma^*). \end{proof} \]

Lemma 8

If \( A \) does not fail and produces \( \sigma''_q \), then \( C(\sigma''_q) \leq (1 + \Gamma \delta)\omega \).

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

Lemma 7 and 8 together prove Theorem 6.

Corollary 2

\( P||U^\Gamma_p|\mathcal{C}_{\text{max}} \) admits a PTAS.

\[ \begin{proof}
\text{Given } \epsilon > 0, \text{ and } \mathcal{I} \text{ an instance of } P||U^\Gamma_p|\mathcal{C}_{\text{max}} \text{ we provide a } (1 + \epsilon)\text{-dual approximation algorithm in the following way. Let } \omega \text{ be the current guess of } \text{opt}({\mathcal{I}}). \text{ We define } \mathcal{I}'' \text{ as previously, and run the (polynomial time) algorithm } A(\mathcal{I}'', \hat{\sigma}^*, \omega) \text{ from Theorem 6. If } A \text{ fails, then we also fail on } \mathcal{I} \text{ according to Observation 3. Otherwise, according to Theorem 6, Observation 4, and Observation 5 we get a schedule } \sigma \text{ of } \mathcal{I} \text{ with } C(\sigma) \leq (1 + \delta)(1 + \Gamma \delta)\omega + \Gamma \delta \omega, \text{ and thus we set } \delta \text{ such that } (1 + \delta)(1 + 2\Gamma \delta) = 1 + \epsilon. \end{proof} \]

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 \( Rm||U^\Gamma_p|\mathcal{C}_{\text{max}} \). Deducing an FPTAS will be straightforward by following the same approach as in [10]. Observe that the main difficulty here is to keep track of the deviating jobs, especially when \( \Gamma \) is not constant. For \( Qm||U^\Gamma_p|\mathcal{C}_{\text{max}} \), this difficulty can be handled by ordering the jobs in non-increasing order of \( \bar{p}_j \). 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 \( \frac{p_i}{s_i} \) if \( x_i = \Gamma \), and \( \frac{p_i + \hat{p}_i}{s_i} \) otherwise.

We show below how to extend the approach to \( Rm||U_{j'}^r|C_{\text{max}} \) 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 \( \hat{p}_{ij} = p_{ij} = 0 \)) so that we can suppose that there are at least \( \Gamma \) jobs on each machine. Thus, we add in the problem the extra constraint that there must be exactly \( \Gamma \) 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 \( \sigma^* \). This means that for any \( i \) and any job \( j \) scheduled on \( i \) in \( \sigma^* \), \( j \in \Gamma(\sigma^*_i) \) 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 \( \{j' \geq j\} \). Formally, let \( \ell \), \( x \), \( j \) be lower bounds on \( l \), \( x \), \( j \) respectively. Let \( \Gamma \) be the budget on deviation.

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 \( (l_i + p_{ij}, x_i) \),
- if \( \hat{p}_{ij} > \hat{p}_i^* \) and \( x_i = \Gamma \) then \( (l_i, x_i) \) becomes \( (\infty, x_i) \) (\( j \) cannot be scheduled on \( i \)),
- if \( \hat{p}_{ij} > \hat{p}_i^* \) and \( x_i < \Gamma \) then \( (l_i, x_i) \) becomes \( (l_i + p_{ij}, \hat{p}_{ij}, x_i + 1) \),
- if \( \hat{p}_{ij} = \hat{p}_i^* \) and \( x_i = \Gamma \) then \( (l_i, x_i) \) becomes \( (l_i + p_{ij}, x_i) \) (there are already \( \Gamma \) 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 \( (l_i + p_{ij}, x_i) \) or \( (l_i + p_{ij}, x_i + 1) \) (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 \( \Gamma \) then \( DP(l, x, j) \) returns \( +\infty \) (remember that we impose that there are exactly \( \Gamma \) deviating jobs on each machine), otherwise it returns \( \max(l_i) \). This concludes the description of \( DP \). Once the \( \{\hat{p}_i^*\} \) are fixed, \( DP \) runs in \( \mathcal{O}(u(\Gamma + 1)^{nm}) \), and thus the overall running time is in \( \mathcal{O}(nu(\Gamma + 1)^{nm}) \).

We can deduce an FPTAS from this pseudopolynomial DP by using the same arguments as in [10]. 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 \( \hat{p}_{ij} \)}, and get a new instance \( I' \) with \( \text{opt}(I') \leq \text{opt}(I) + 2nd \). We solve \( I' \) with the previous \( DP \) in \( \mathcal{O}(nu'(\Gamma + 1)^{nm}) \) where \( u' = \frac{u}{d} \). Thus, as we need \( 2nd \leq d \), we
take \( d = \frac{1}{2n} \) 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 \( \Gamma \) we can simply re-use any existing approximation algorithm.

**Lemma 9** From any polynomial-time \( \rho \)-approximation \( A \) for \( R|C_{\text{max}} \) we can deduce a polynomial \( (\rho + \Gamma) \)-dual approximation \( A^\Gamma \) for \( R|U^\Gamma|C_{\text{max}} \). In particular, [16] gives a \( 2 + \Gamma \) dual approximation.

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

We provide a more refined algorithm that yields an average \( O(\log m) \) approximation factor. Notice that the straightforward generalization of the formulation from [16] 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 \( i \in M \) and each \( \nu \subseteq J \), \( \lambda_{i\nu} = 1 \) if the set of jobs executed on machine \( i \) is precisely \( \nu \), and zero otherwise. Let \( \mu(j, \nu) = 1 \) if \( j \in \nu \), and zero otherwise, and \( \alpha(i, \nu) = \max \left\{ \sum_{j \in \nu} (\bar{p}_{ij} + \xi_j \hat{p}_{ij}) | \xi \in \{0, 1\}^n, \sum_{j \in J} \xi_j \leq \Gamma \right\} \).

The formulation follows:

\[
\begin{align*}
\text{Min} & \quad \omega \\
\text{S.t.} & \quad \sum_{i \in M} \sum_{\nu \subseteq \mathcal{J}} \mu(j, \nu) \lambda_{i\nu} = 1, \quad \forall j \in \mathcal{J} \\
& \quad \omega \geq \sum_{\nu \subseteq \mathcal{J}} \alpha(i, \nu) \lambda_{i\nu}, \quad \forall i \in M \\
& \quad \sum_{\nu \subseteq \mathcal{J}} \lambda_{i\nu} = 1, \quad \forall i \in M \\
& \quad \lambda_{i\nu} \in \{0, 1\}, \quad \forall (i, \nu) \in M \times 2^\mathcal{J}
\end{align*}
\]
As with the formulation from [16], the value of the lower bound improves if we drop the objective function and remove all variables \( \lambda_{iv} \) such that \( \alpha(i, \nu) \) is greater than a given target makespan value \( \omega \). Namely, we consider the lower bound for \( R||U\Gamma||C_{\text{max}} \) defined as follows

\[
LB = \{ \min \omega : \text{FP}(\omega) \text{ is feasible} \},
\]  
(16)

where \( \text{FP}(\omega) \) is defined by the following linear constraints:

\[
\sum_{j \in J} \sum_{\nu \subseteq J \atop \alpha(i, \nu) \leq \omega} \mu(j, \nu)\lambda_{iv} = 1, \quad \forall j \in J
\]  
(17)

\[
\sum_{\nu \subseteq J \atop \alpha(i, \nu) \leq \omega} \lambda_{iv} = 1, \quad \forall i \in M
\]  
(18)

\[
\lambda_{iv} \geq 0, \quad \forall (i, \nu) \in M \times 2^J
\]  
(19)

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

**Theorem 7** We can compute in polynomial time a 2-approximate solution for \( LB \).

**Proof** We solve problem (16) using a dual-approximation algorithm. Namely, for each value of \( \omega \), either we show that \( \text{LP}(2\omega) \) is feasible or that \( \text{FP}(\omega) \) is infeasible. Then, the minimum value of \( \omega \) for which \( \text{FP}(\omega) \) is feasible that leads to zero objective value can be found through a binary search.

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

\[
\text{Min} \quad \sum_{j \in J} s_j
\]  
(20)

\[
\text{S.t.} \quad \sum_{i \in M} \sum_{\nu \subseteq J \atop \alpha(i, \nu) \leq \omega} \mu(j, \nu)\lambda_{iv} + s_j = 1, \quad \forall j \in J
\]  
(21)

\[
\sum_{\nu \subseteq J \atop \alpha(i, \nu) \leq \omega} \lambda_{iv} = 1, \quad \forall i \in M
\]  
(22)

\[
\lambda_{iv} \geq 0, \quad \forall (i, \nu) \in M \times 2^J, \alpha(i, \nu) \leq \omega
\]  
(23)

\[
s_j \geq 0, \quad \forall j \in J
\]  
(24)
The continuous relaxation of the previous formulation can be solved in polynomial time, using for instance the Ellipsoid method [11], if the problem of pricing the \( \lambda \) variables is also polynomially solvable [9]. Such a pricing problem can be stated as follows. Let \( \pi_j \) and \( \theta_i \) be dual variables associated to constraints (21), and (22), 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 \( i \in M \), we want to find \( \nu \in J \) 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 \( \nu^* \) with reduced cost \( \bar{c}^* \leq 2\omega \) 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 \( \alpha(i, \nu) \leq 2\omega \), 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 \( \nu^* \). Remark that if \( \hat{p} = 0 \) (the problem is deterministic), such a solution \( \nu^* \) 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 [7].

In the remainder of the section, we let \( \omega \) be the solution returned by Theorem 7 and \( \lambda^* \) be the corresponding fractional vector. Our objective is to use randomized rounding to obtain an integer solution to \( R||U\Gamma||C_{max} \) with an average makespan of at most \( O(\log(m))\omega \). 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 9).

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 \( \nu \) associated to a variable \( \lambda_{i\nu} \) has a total processing time of at most \( \omega \), 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
Algorithm 3 Randomized rounding (input: a feasible solution $(\lambda^*)$ of $FP(\omega)$)

\[ y \leftarrow 0; \]
\[ \text{while there exists a job } j \in J \text{ not assigned to any machine do} \]
\[ \text{for } i \leftarrow 1, \ldots, m \text{ do} \]
\[ \text{Randomly select a schedule } \nu^* \text{ for machine } i \text{ with probability } \lambda^*_i \text{ of selecting each schedule } \nu; \]
\[ \text{for each } j \in \nu^* \text{ do} \]
\[ y_{ij} \leftarrow y_{ij} + 1; \]
\[ \text{if job } j \text{ is not assigned to any machine then} \]
\[ \text{Assign job } j \text{ to machine } i; \]
\[ \text{end if} \]
\[ \text{end for} \]
\[ \text{end for} \]
\[ \text{end while} \]

$K$ 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 $\mu$, the probability that $X < (1 - \delta)\mu$, for any $\delta > 0$, is smaller than $e^{-\delta^2/2}$. The next Theorem uses this bound to limit the value of $t$.

Theorem 8 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 $X^q_{j,i} = 1$ 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 $X^q_{j,i}$ with $t < q \leq t^*$ are set to 0). Clearly, the random variables $X^q_{j,i}$ are independent. Moreover, the constraints (21) ensure that $E(\sum_q X^q_{j,i}) = 1$ for any $j$ and $q$, and thus the expected value of $X^j = \sum_{q \in [t^*], i \in [m]} X^q_{j,i}$ is equal to $t^*$. Now, applying the Chernoff bound with $\delta = 1 - 1/t^*$, and assuming that $t^* \geq 4$, we obtain that

\[ \Pr[X^j < 1] < e^{-\left(\frac{t^*-1}{2t^*}\right)^2} < e^{-t^*/4} \leq \frac{1}{2n}. \] (25)

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

\[ \Pr[X = 0] = \Pr[\sum_{j=1}^n X^j < n] \leq \sum_{j=1}^n \Pr[X^j < 1] < 1/2, \] (26)

which completes our proof.

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

\[ \square \]
Proof An immediate consequence of Theorem 8 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 probability 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 10** $E(C_{\text{max}}) = O(\log(m)) \omega$.

**Proof** For each value $\lambda^*_j > 0$ 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.

$$\alpha_j(i, \nu) = \begin{cases} \bar{p}_j + \hat{p}_j, & \text{if } j \in \Gamma(\nu), \\ \bar{p}_j, & \text{if } j \in \nu \setminus \Gamma(\nu), \\ 0, & \text{if } j \not\in \nu. \end{cases}$$

(27)

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

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

(28)

where $\nu_{i, \ell}$ is the schedule selected for machine $i$ in the $\ell$th 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 (28), we consider a new upper bound on $E(C_{\text{max}})$ 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 $\ell$th iteration, which is defined as follows.

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

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

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

(29)

Note that $\beta_j(i, \ell)$ can be strictly positive only if $j$ is not scheduled before the $\ell$th 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 $q_{j, \ell} = (q_{j, 2})^{\ell-1}$. 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

$$q_{j2} = \prod_{i \in M} (1 - \Pr[job j is scheduled on machine i])$$

(30)

$$\leq (1 - 1/m)^m < 1/e$$

The first inequality is true because setting all probabilities equal to $1/m$ maximizes the right-hand side of (30) subject to the constraint that the sum of all probabilities is one. Hence, we have that $q_{j2} < 1/e^{\ell - 1}$. Then, we obtain that

$$\mathbb{E}(C_{\text{max}}) \leq \mathbb{E}\left(\max_{i \in M} \left\{ \sum_{\ell=1}^{t} \beta(i, \ell) \right\} \right)$$

(31)

$$\leq \mathbb{E}\left(\max_{i \in M} \left\{ \sum_{\ell=\lceil \ln m \rceil+1}^{t} \alpha(i, \nu, \ell) \right\} + \max_{i \in M} \left\{ \sum_{\ell=\lceil \ln m \rceil+1}^{t} \beta(i, \ell) \right\} \right)$$

(32)

$$\leq \sum_{\ell=1}^{\lceil \ln m \rceil} \omega + \sum_{i \in M} \mathbb{E}\left( \sum_{\ell=\lceil \ln m \rceil+1}^{t} \sum_{j \in J} \beta_j(i, \ell) \right)$$

(33)

$$< \omega \lceil \ln m \rceil + \sum_{i \in M} \sum_{\ell=\lceil \ln m \rceil+1}^{t} \frac{1}{e^{\ell-1}} \sum_{\nu \subseteq J} \lambda^*_i,\nu \alpha_j(i, \nu)$$

$$= \omega \lceil \ln m \rceil + \sum_{i \in M} \sum_{\ell=\lceil \ln m \rceil+1}^{t} \frac{1}{e^{\ell-1}} \sum_{j \in J} \lambda^*_i,\nu \alpha_j(i, \nu)$$

$$\leq \omega \lceil \ln m \rceil + \sum_{i \in M} \sum_{\ell=\lceil \ln m \rceil+1}^{t} \frac{1}{e^{\ell-1}}$$

$$< \omega \lceil \ln m \rceil + m \frac{e}{m(e-1)} \omega$$

$$= \left(\lceil \ln m \rceil + \frac{e}{e-1}\right) \omega.$$ 

Inequality (31) follows from $\beta(i, \ell) \leq \alpha(i, \nu, \ell)$, inequality (32) follows from $\mathbb{E}(\alpha(i, \nu, \ell)) \leq \omega$, which holds because of the definition of $FP(\omega)$, inequality (33) follows from the definition of $\beta_j(i, \ell)$ in (29), and the other inequalities are obtained similarly.

\[\square\]

We obtain easily the following result.
Theorem 9 There is an $O(\log(m))$-approximation in expectation for $R[|U\Gamma|C_{\text{max}}]$.

Proof (Proof of Theorem 9) Let us define a randomized $O(\log(m))$-dual approximation that given a threshold $\omega$ either creates a schedule with $E(C_{\text{max}}) \leq O(\log(m))\omega$, or fails, implying that $\omega < \text{opt}$ (where opt is the optimal solution cost of the $R[|U\Gamma|C_{\text{max}}]$ input). Given $\omega$, we apply Theorem 7 to either compute a fractional solution of cost $2\omega$ of $LB$, or fail (implying $\omega < \text{opt}(LB) \leq \text{opt}$). If the algorithm does not fail, we applying Lemma 10 to round this solution to an integer solution with expected makespan $E(C_{\text{max}}) \leq O(\log(m))2\omega$. □

References


