Proceedings of the Thirty-First International Conference on Automated Planning and Scheduling (ICAPS 2021)

Scheduling Stochastic Jobs - Complexity and Approximation Algorithms

Liangde Tao[1], **Lin Chen**[2*], **Guochuan Zhang**[1]
[1] College of Computer Science, Zhejiang University
[2] Department of Computer Science, Texas Tech University
vast.tld@gmail.com, chenlin198662@gmail.com, zgc@zju.edu.cn

Abstract

Uncertainty is an omnipresent issue in real-world optimization problems. This paper studies a fundamental problem concerning uncertainty, known as the β-robust scheduling problem. Given a set of identical machines and a set of jobs whose processing times follow a normal distribution, the goal is to assign jobs to machines such that the probability that all the jobs are completed by a given common due date is maximized. We give the first systematic study on the complexity and algorithms for this problem. A strong negative result is shown by ruling out the existence of any polynomial-time algorithm with a constant approximation ratio for the general problem unless P=NP. On the positive side, we provide the first FPT-AS (fixed parameter tractable approximation scheme) parameterized by the number of different kinds of jobs, which is a common parameter in scheduling problems. It returns a solution arbitrarily close to the optimal solution, provided that the job processing times follow a few different types of distributions. We further complement the theoretical results by implementing our algorithm. The experiments demonstrate that by choosing an appropriate approximation ratio, the algorithm can efficiently compute a near-optimal solution.

Introduction

Scheduling is a fundamental problem in computer science and has received extensive study in the literature. The classical scheduling problems, where jobs have deterministic processing times, are well understood. However, in many real-world scheduling problems, the uncertainty of the input data is inevitable. This is particularly the case in the scheduling and planning of activities of multiple agents under an uncertain environment (see, e.g., (Hiatt et al. 2009)). Unfortunately, the stochastic version of the scheduling problem, where jobs have processing times that follow a certain distribution, is much more challenging and far from well understood so far.

Very recently, Stec et al. (2019) revisited the central problem proposed by Daniels and Carrillo (1997), known as the β-robust scheduling problem. Specifically, Stec et al. (2019) considered the problem, where each job has a processing time that follows a normal distribution. The goal is to assign jobs to machines in order to maximize the probability that all the jobs are completed by a given common due date.

Many real-world applications, such as network routing, can be viewed as an example of the β-robust scheduling problem. Consider the traffic demands between a pair of nodes s and t in the network. Each traffic demand can be viewed as a (stochastic) job and needs to be routed through one of the links (or paths) between s and t. Each link can be viewed as a machine, which can accept traffic demands (jobs) up to the bandwidth limitation. The goal is to distribute traffic demands (jobs) among links (machines) which maximizes the probability that the total load of every link is no more than the bandwidth limitation. One very common network structure is the ring topology, where there are exactly two links between s and t, and hence the number of machines is 2. It is known that that typical traffic demands like web surfing, downloading, streaming, etc., usually belong to a few categories (e.g., routers manufactured by Huawei limit the maximum number of different types of traffic demands to be 16). Approximating the traffic demand in each category via the normal distribution is also a common approach (Pras et al. 2009).

Stec et al. (2019) proposed an algorithm for the β-robust scheduling problem based on Branch-and-Price and demonstrated its efficiency through experiments. Motivated by this, it is natural to ask whether the problem admits an efficient algorithm with a theoretical approximation ratio guarantee. Indeed, given that the deterministic scheduling problem has several simple heuristics with a small approximation ratio (e.g., List-Scheduling (Graham 1969)), and that normal distribution is a class of common distribution that is usually considered as "mild" in stochastic scheduling (contrasting to those distributions with a heavy tail), one may well expect a good approximation algorithm. Unfortunately, we prove that such an algorithm does not exist under a standard complexity assumption. In the meantime, we also establish efficient algorithms by exploiting parameters that are typically small and complement our theoretical results through experiments. The specific contribution of this paper is as follows.

Our Contribution

This paper gives the first systematic study on the approximability of β-robust scheduling on parallel machines. As a

*Corresponding author.

strong negative result, we prove that no polynomial time algorithm can distinguish between an instance of the β-robust scheduling with an objective value arbitrarily close to 1 and an instance with an objective value arbitrarily close to 0 unless P=NP. This immediately rules out any polynomial time algorithm with a multiplicative approximation ratio of $O(1)$ as well as that with an $O(1)$ additive error.

We then investigate the problem where a parameter can be utilized to break the strong complexity barrier. We first establish an algorithm which runs in $O((\frac{1}{\epsilon}n^2 m\mu_{max}\log(n\sigma_{max}))^m)$time and returns a near-optimal solution. Here, m, n are the number of machines and jobs, respectively, and μ_{max} and σ_{max} are the maximal mean and standard deviation among all normal distributions. It is not hard to see that the running time of the algorithm relies heavier on μ_{max} than σ_{max}. This coincides with the complexity proof, where the reduction consists of normal distributions with arbitrary means but unit standard deviation. Consequently, our results illustrate that, interestingly, the mean is much more critical than the standard deviation. It remains an important open problem whether this observation holds for more general distributions.

Next, we establish an algorithm that returns a near-optimal solution and runs in $O((\frac{1}{\epsilon})^{2m}(mk)^{O(mk)}L^{2m+1})$ time, where L is the length of the input and k is the number of different kinds of jobs, i.e., job processing times only belong to k distinct normal distributions. Following the recent advances in the FPT (fixed-parameter tractable) algorithms for deterministic scheduling problems (see, e.g., (Mnich and Wiese 2015; Knop and Koutecký 2018)), this is the first FPT algorithm for robust scheduling models. It indicates that the common observation in deterministic scheduling problems, where reducing the "diversity" of jobs greatly improves tractability, is also true in robust scheduling.

Finally, we complement our theoretical results by implementing our FPT algorithm. The experimental results show that with a reasonably small error, our algorithm also runs efficiently in practice.

Formal Statement of the Problem

The β-robust scheduling problem (β-RSP) is characterized by a set of n independent jobs $J = \{1, \ldots, n\}$ and a set of m parallel identical machines $M = \{1, \ldots, m\}$. Each job $j \in J$ has a stochastic processing time p_j obeying independent normal distribution with mean $\mu_j \in \mathbb{N}$ and variance $\sigma_j^2 \in \mathbb{N}^+$. The common due date is denoted by $\delta \in \mathbb{N}$. Here β denotes the formulation of the objective i.e., the probability of the maximum load not exceeding the common due date. Using the three-field notation (Graham et al. 1979), the problem can be represented by $P|p_j \sim \mathcal{N}(\mu_j, \sigma_j^2)|Pr[C_{max} \leq \delta]$. To exclude the trivial cases, we assume $n > m$ so that in optimal solutions each machine would handle at least one job. Let Φ be the cumulative function of the standard normal distribution $\mathcal{N}(0, 1)$. The β-robust scheduling problem can be formulated as the following non-linear integer program (NIP).

$$\max \quad \prod_{i \in M} \Phi\left(\frac{\delta - \mu_{M_i}}{\sqrt{\sigma_{M_i}^2}}\right)$$

$$\text{s.t.} \quad \sum_{j \in J} \mu_j \cdot x_{ij} = \mu_{M_i} \qquad \forall i \in M \qquad (1a)$$

$$\sum_{j \in J} \sigma_j^2 \cdot x_{ij} = \sigma_{M_i}^2 \qquad \forall i \in M \qquad (1b)$$

$$\sum_{i \in M} x_{ij} = 1 \qquad \forall j \in J \qquad (1c)$$

$$x_{ij} \in \{0, 1\} \qquad \forall i \in M, j \in J$$

The binary decision variable $x_{ij} = 1$ denotes that job j is scheduled on machine i, and $x_{ij} = 0$ otherwise.

We explain the above formulation. Note that normal distribution has the following property: the summation of finitely many independent normally distributed variables is also a normally distributed random variable, with its mean equaling the summation of means, and variance equaling the summation of variances. First, notice that Eq (1c) enforces that every job has to be scheduled on exactly one machine. In Eq (1a) and Eq (1b), we consider all jobs scheduled on machine i, the summation of their processing time is a normally distributed random variable with mean $\mu_{M_i} = \sum_{j \in J} \mu_j \cdot x_{ij}$ and variance $\sigma_{M_i}^2 = \sum_{j \in J} \sigma_j^2 \cdot x_{ij}$.

The objective of NIP is straightforward. Given the mean and variance of the random variable that represents the load of each machine, the probability that the load of machine i does not exceed the common due date δ can be expressed by $\Phi(\frac{\delta - \mu_{M_i}}{\sqrt{\sigma_{M_i}^2}})$. Consequently, the probability that the load for each machine does not exceed the common due date δ is the product of the probability for each machine.

Related Work

Unlike the well-understood classic scheduling problem, how to handle stochastic processing times is a big challenge. To this end, Daniels and Carrillo (1997) firstly introduced the β-robust measure for single machine scheduling minimizing the total flow time and gave the NP-hardness proof. In their model, the processing time of each job is independent and obeys normal distribution. They proposed a branch-and-bound algorithm to solve this problem which could only handle 20 jobs. More efficient and heuristic algorithms were also addressed by Wu et al. (2009). Latter, Alimoradi et al. (2016) generalized the problem to parallel identical machines setting and obtained an exact algorithm that has better performance on the data set. Recently, Zhang et al. (2018) addressed another variant, where the independent normal distribution assumption is not required. In their model, the job's processing times are not independent. Only mean and covariance are known, the aim is to maximize the probability of total flow time no greater than the given threshold in the worst case.

Under identical machine setting, Ranjbar et al. (2012) and Stec et al. (2019) considered the β-robust scheduling minimizing the makespan and propose exact algorithms. Their

algorithms are based on the same nonlinear programming model and able to solve instances with 6 machines and 20 jobs within 1000 seconds. Under unrelated machine setting, Pishevar et al. (2014) took the total completion time as the objective. The authors approximately modeled the problem with mixed-integer programming and devise a heuristic algorithm. Different from the β-robust measure, stochastic scheduling aims at minimizing/maximizing the expected value of the objectives, see, e.g. (Bruno, Downey, and Frederickson 1981; Glazebrook 1979; Jäger and Skutella 2018; Im, Moseley, and Pruhs 2015; Eberle et al. 2019; Möhring, Schulz, and Uetz 1999; Megow, Uetz, and Vredeveld 2006; Skutella, Sviridenko, and Uetz 2016).

In recent years, there are some breakthroughs on FPT algorithms for scheduling (Mnich and Wiese 2015) and followed by several works including (van Bevern, Niedermeier, and Suchỳ 2017; Chen et al. 2017; Knop and Koutecký 2018; Jansen, Maack, and Solis-Oba 2020). We refer the reader to a nice survey (Mnich and van Bevern 2018). Most of the existing results are on deterministic scheduling problems, while we are not aware of FPT algorithms for the stochastic/robust version.

Preliminaries

In general, approximation algorithms are defined with respect to the multiplicative ratio.

Definition (α-approximation algorithm)**.** *For a maximization problem, ALG is an α-approximation algorithm if for any instance I of the problem it holds that $ALG(I) \geq \alpha \cdot OPT(I)$.*

In this paper, we are primarily interested in algorithms with a small additive error, as defined below.

Definition (approximation algorithm with additive error ϵ)**.** *For a maximization problem, ALG is an approximation algorithm with additive error ϵ if for any instance I of the problem it holds that $ALG(I) \geq OPT(I) - \epsilon$.*

Definition (Parameterized algorithm)**.** *For an instance of a parameterized problem $\Pi \subseteq \Sigma^* \times \mathbb{N}$, a pair (x, k) consists of the input x and the parameter k. A parameterized problem is fixed-parameter tractable (FPT) if there is an algorithm solving any instance of Π with parameter k and size n in $f(k) \cdot poly(n)$ time for some computable function f.*

Inapproximability

The goal of this section is to show that the β-robust scheduling problem remains hard to approximate even the number of machines is restricted to 2. More specifically, we prove the following theorem.

Theorem 1. *For any $c \in (0, 1)$, there is no polynomial time algorithm that can return a feasible solution with objective value at least $OPT - c$ for any instance of the problem $2|p_j \sim \mathcal{N}(\mu_j, \sigma_j^2)|Pr[C_{max} \leq \delta]$, assuming $P \neq NP$.*

Towards the proof, we reduce from $2\text{-}PARTITION$. The instance of the problem and the solution is given as follows:

2-PARTITION
Input: $S = \{1, \ldots, n\}$, $w = (w_1, \ldots, w_n) \in \mathbb{N}^n$ guaranteeing $\sum_{i \in S} w_i = 2M$.
Output: Is there $S' \subseteq S$ such that $\sum_{i \in S'} w_i = \sum_{i \in S \setminus S'} w_i$?

Proof of Theorem 1. As we know, $2\text{-}PARTITION$ is a weakly NP-hard problem. Assume $P \neq NP$, there is no polynomial time algorithm for $2\text{-}PARTITION$ (Garey and Johnson 2002). Given an instance (n, w, M) of $2\text{-}PARTITION$, we construct an instance (δ, μ, σ) of the β-robust scheduling problem with 2 machines, n jobs, the common due date $\delta = (M + \frac{1}{2})n$ and $\mu_j = nw_j, \sigma_j^2 = 1$ for each job. All jobs constructed have a uniform standard deviation.

Recall the formulation of the non-linear integer program (NIP). Suppose the answer of $2\text{-}PARTITION$ instance (n, w, M) is "yes". Then it is possible to distribute jobs on two machines such that for jobs on each machine, the summation of their means is exactly Mn. Consequently, the objective value of NIP becomes $\Phi(\frac{\sqrt{n}}{2}) \cdot \Phi(\frac{\sqrt{n}}{2})$, since Φ is the cumulative function of standard normal distribution.

Suppose the answer of $2\text{-}PARTITION$ instance (n, w, M) is "no". Then it is impossible to evenly distribute all jobs regarding their means. In particular, in any feasible solution the expected load of one machine is $(M + \tau)n$ and of the other machine is $(M - \tau)n$, for some integer $\tau \geq 1$. Given that $\delta = (M + \frac{1}{2})n$, it is easy to see that for the machine whose expected load is $(M + \tau)n$, the probability that the random variable is bounded by δ is $\Phi(-(\tau - \frac{1}{2})\sqrt{n}) \leq \Phi(-\frac{\sqrt{n}}{2})$. That is, for any feasible solution, the objective value of NIP is no more than $\Phi(-\frac{\sqrt{n}}{2})$.

Note that as $n \to \infty$, $\Phi(\frac{\sqrt{n}}{2}) \to 1$ and $\Phi(-\frac{\sqrt{n}}{2}) \to 0$. Since c is a constant, for sufficiently large n, we have $\Phi(\frac{\sqrt{n}}{2}) \cdot \Phi(\frac{\sqrt{n}}{2}) > \frac{c+1}{2}$ and $\Phi(-\frac{\sqrt{n}}{2}) < \frac{1-c}{2}$. Suppose there exists a polynomial time algorithm that returns a solution with objective value at least $OPT - c$ for any constant $c \in (0, 1)$. Then if the answer for $2\text{-}PARTITION$ instance (n, w, M) is "yes", the algorithm should return a solution for the constructed scheduling instance with objective value strictly larger than $\frac{c+1}{2} - c = \frac{1-c}{2}$. Otherwise, the algorithm returns a solution with objective value strictly smaller than $\frac{1-c}{2}$. That is, such an algorithm for scheduling problem can be used to solve $2\text{-}PARTITION$ in polynomial time, which is a contradiction. □

Using the same reduction, the corollary below is also true.

Corollary 1. *For any $c \in (0, 1)$, there is no polynomial time algorithm could give a c-approximation answer for any instance of the problem $P|p_j \sim \mathcal{N}(\mu_j, \sigma_j^2)|Pr[C_{max} \leq \delta]$, assuming $P \neq NP$.*

Algorithms

Given Theorem 1, we know that if we aim for a good approximation algorithm, then the running time is inevitably non-polynomial (pseudo-polynomial or exponential) in some of

Algorithm 1 Dynamic Programming for β-RSP

Input: $n, m, \{\mu_j\}, \{\sigma_j^2\}, \delta$
Output: an optimal assignment

$\mathcal{F}_0 = \{(0, \dots, 0)\}$
for $h = 1$ to n **do**
  $\mathcal{F}_h = \emptyset$
  for all $(\mu_{M_1}, \sigma^2_{M_1}, \dots, \mu_{M_m}, \sigma^2_{M_m}) \in \mathcal{F}_{h-1}$ **do**
    for $i = 1$ to m **do**
      $\mathcal{F}_h \leftarrow \mathcal{F}_h \cup (\dots, \mu_{M_i} + \mu_h, \sigma^2_{M_i} + \sigma^2_h, \dots)$
    end for
  end for
end for
return $\max_{\mathbf{x} \in \mathcal{F}_n} obj(\mathbf{x})$

the parameters. The research on FPT algorithms for the classical (deterministic) scheduling problem has revealed two important parameters: the largest job processing time and the number of distinct jobs (Chen et al. 2017; Knop and Koutecký 2018; Mnich and Wiese 2015). The two parameters yield tractability in theory. The question is, what are their corresponding parameters in robust scheduling?

Parameters corresponding to the largest job processing time. It is easy to see if the job processing time follows a normal distribution, then two parameters contribute to the realization of its processing time, mean and standard deviation. A job may have a huge processing time if either of the two parameters is large. It is thus natural to expect a good algorithm if both the mean and standard deviation are small. Indeed, we present an optimal algorithm (Algorithm 1) that runs polynomially in μ_{max} and σ_{max} if the number of machines is a constant, where μ_{max} and σ_{max} are the largest mean and standard deviation. Somewhat surprisingly, we show a near-optimal algorithm (Algorithm 2) that runs polynomially in μ_{max} and $\log \sigma_{max}$. This, combined with the fact that our reduction in Theorem 1 only constructs jobs with unit standard deviation, reveals the following interesting fact: Although both mean and standard deviation contributes to the random processing time, mean is much more critical for robust scheduling problems in the sense there is no complementary near-optimal algorithm that runs polynomially in $\log \mu_{max}$ and σ_{max}.

Parameters corresponding to the number of distinct jobs. In deterministic scheduling problems, jobs of the same kind refer to jobs of the same processing time. In robust scheduling, jobs with the same kind of processing times follow the same distribution. Note that it does not necessarily mean they have the same processing time. Nevertheless, we are still able to establish a near-optimal FPT algorithm (Algorithm 3) parameterized by the number of distinct distributions. In many real-world applications, the number of distinct jobs (i.e., the parameter k) is reasonably small. Taking k as the parameter is acceptable. For example, in the network routing problem (discussed in the introduction) the parameter k is at most 16 which is a constant.

An Optimal Algorithm Pseudo-Polynomially in Both Mean and Variance

The subsection is to establish a dynamic programming (Algorithm 1) which is pseudo-polynomial in both the largest mean and the largest variance.

We consider the following sub-problem: Is it possible to assign the first h jobs, i.e., job $1, \dots, h$, such that the summation of the means of all jobs on machine i is μ_{M_i} and the summation of their variances is $\sigma^2_{M_i}$?

Denote the sub-problem by a $(2m+1)$-dimensional vector $(h, \mu_{M_1}, \sigma^2_{M_1}, \dots, \mu_{M_m}, \sigma^2_{M_m})$. If the answer is yes, then the vector is feasible, called a stage-h state. We maintain feasible states throughout the following algorithm where all stage-h states are stored in $\mathcal{F}_h$.

Initially, $(0, \dots, 0) \in \mathcal{F}_0$ is the only stage-0 state.

Iterative procedure: Suppose all stage-(h-1) states are computed and stored in $\mathcal{F}_{h-1}$ so far. For each stage-(h-1) state $(\mu_{M_1}, \sigma^2_{M_1}, \dots, \mu_{M_m}, \sigma^2_{M_m}) \in \mathcal{F}_{h-1}$, we compute m stage-h states, which are $(\mu_{M_1} + \mu_h, \sigma^2_{M_1} + \sigma^2_h, \dots, \mu_{M_m}, \sigma^2_{M_m})$, $\dots$, $(\mu_{M_1}, \sigma^2_{M_1}, \dots, \mu_{M_m} + \mu_h, \sigma^2_{M_m} + \sigma^2_h)$, each corresponding to one extension of the partial solution by assigning job h to one of the machines.

Finally, after all the stage-n states are calculated and stored in $\mathcal{F}_n$, the algorithm simply selects one state from $\mathcal{F}_n$ with the maximum objective value.

Theorem 2. *Algorithm 1 gives an optimal solution for the problem $P|p_j \sim \mathcal{N}(\mu_j, \sigma_j^2)|Pr[C_{max} \le \delta]$ within $O(mn^{2m+1}(\mu_{max}\sigma_{max})^m)$ time, where $\mu_{max} = \max_j \mu_j$ and $\sigma_{max} = \max_j \sigma_j^2$.*

Proof. Suppose in an optimal assignment, machine i handles all jobs in J_i^*. For convenience, let $J_i^*(h) = J_i^* \cap \{1, \dots, h\}$. We know for $1 \le h \le n$, it holds that $(\dots, \sum_{j \in J_i^*(h)} \mu_j, \sum_{j \in J_i^*(h)} \sigma_j^2, \dots) \in \mathcal{F}_h$. Hence, the optimal solution is contained in $\mathcal{F}_n$.

It is not hard to see that the total number of operations done in Algorithm 1 is $O(m \sum_{h=1}^{n} |\mathcal{F}_h|)$. As all numbers involved are integers, the size of $\mathcal{F}_h$ is bounded by $(n^2 \mu_{max} \sigma_{max})^m$. □

A Near-Optimal Algorithm Pseudo-polynomially in Mean

In this part, we show that if $O(\epsilon)$-loss in the objective value is allowed, then there exists an algorithm with running time only pseudo-polynomial in the largest mean. Specifically, it can be achieved by modifying Algorithm 1 a little bit. Reviewing Algorithm 1, the most time-consuming part is the enumeration of all the possible states stored in $\mathcal{F}_1, \dots, \mathcal{F}_n$. We seek to store fewer states without losing much. For the normal distribution, the cumulative distribution function has the property that when variables $x, \hat{x}$ are close, the difference between $\Phi(x)$ and $\Phi(\hat{x})$ is also small. The property can be expressed as Lemma 1, which explains why we can round the variance but not the mean. For an arbitrary normal distribution $X \sim \mathcal{N}(\mu, \sigma^2)$, if we perturb the variance slightly to generate a new random variable $\hat{X} \sim \mathcal{N}(\mu, \hat{\sigma}^2)$, then the probability $\Pr(\hat{X} \le x)$ is close to $\Pr(X \le x)$, as is ensured

Algorithm 2 Approximation Scheme for β-RSP

Input: $\epsilon, n, m, \{\mu_j\}, \{\sigma_j^2\}, \delta$
Output: an $(OPT-\epsilon)$ assignment

$\xi = \epsilon/2nm$
$\hat{\mathcal{F}}_0 = \{(0, \dots, 0)\}$
for $h = 1$ to n **do**
  $U_h = \emptyset$
  for all $(\mu_{M_1}, \hat{\sigma}^2_{M_1}, \dots, \mu_{M_m}, \hat{\sigma}^2_{M_m}) \in \hat{\mathcal{F}}_{h-1}$ **do**
    for $i = 1$ to m **do**
      $U_h \leftarrow U_h \cup (\dots, \mu_{M_i} + \mu_h, \hat{\sigma}^2_{M_i} + \sigma^2_h, \dots)$
    end for
  end for
  $\hat{\mathcal{F}}_h$=Subroutine($U_h, \Gamma_\mu, \Gamma_\sigma(\xi)$)
end for
return $\max_{\mathbf{x} \in \hat{\mathcal{F}}_n} obj(\mathbf{x})$

by Lemma 1. However, if we perturb the mean slightly, the probability may change significantly.

Lemma 1. *(Chen et al. 2019) For any $x \in (-\infty, \infty)$ and $\delta > 0$, it holds that $|\Phi((1+\delta)x) - \Phi(x)| \le \delta$.*

In Algorithm 2, we no longer store all states. In each step, a few representative states will be constructed and stored. In the following step, Algorithm 2 will use the representative states to generate new (representative) states. The key is to bound the overall error introduced.

Let $\xi = \epsilon/2mn$. Partition geometrically the set of integers into $\Gamma_\sigma(\xi) = \{[0], (0,1], (1, 1+\xi], \dots, ((1+\xi)^{\gamma-1}, (1+\xi)^\gamma]\}$, where $(1+\xi)^{\gamma-1} < \sum_j \sigma_j^2 \le (1+\xi)^\gamma$. To be consistent, we define $\Gamma_\mu = \{[0], [1], \dots, [\sum_j \mu_j]\}$, which is essentially a set of integers. Algorithm 2 proceeds in a very similar way as Algorithm 1. We call $(\mu_{M_1}, \hat{\sigma}^2_{M_1}, \dots, \mu_{M_m}, \hat{\sigma}^2_{M_m}) \in \hat{\mathcal{F}}_h$ a trimmed stage-h state. We again start from the initial trimmed stage-0 state $(0, 0, \cdots, 0) \in \hat{\mathcal{F}}_0$. Each stage-($h$-1) trimmed state gives rise to m vectors, where each computed vector falls into one region partition by Γ_μ and $\Gamma_\sigma(\xi)$. In each region, the computed vectors are rounded up to a trimmed stage-h state. Subroutine 1 summarizes the details with the rounding procedure. The following lemma illustrates the relationship between $\mathcal{F}_h$ and $\hat{\mathcal{F}}_h$ and estimates the error accumulated.

Lemma 2. *For any $(\mu_{M_1}, \sigma^2_{M_1}, \dots, \mu_{M_m}, \sigma^2_{M_m}) \in \mathcal{F}_h$, there exists $(\mu_{M_1}, \hat{\sigma}^2_{M_1}, \dots, \mu_{M_m}, \hat{\sigma}^2_{M_m}) \in \hat{\mathcal{F}}_h$ such that*

$$\sigma^2_{M_i} \le \hat{\sigma}^2_{M_i} \le (1+\xi)^h \sigma^2_{M_i}.$$

Finally, we can show the near-optimality of Algorithm 2 as Theorem 3, with the help of the following lemmas.

Lemma 3. *If $a_i, b_i \in (0,1)$, then it holds that $|\prod_{i=1}^m a_i - \prod_{i=1}^m b_i| \le \sum_{i=1}^m |a_i - b_i|$.*

Lemma 4. *For any $x \in [0,1]$ and $m \ge 1$, it holds that $(1 + x/m)^m \le 1 + 2x$.*

Theorem 3. *For any $\epsilon > 0$, Algorithm 2 gives an $(OPT-\epsilon)$ solution for the problem $P|p_j \sim \mathcal{N}(\mu_j, \sigma_j^2)|Pr[C_{max} \le \delta]$ within $O((\frac{1}{\epsilon} n^2 m \mu_{max} \log(n\sigma_{max}))^m)$ time, where $\mu_{max} = \max_j \mu_j$ and $\sigma_{max} = \max_j \sigma_j^2$.*

Subroutine 1 Subroutine-for-Algorithm 2

Input: $\mathcal{F}, \Gamma_\mu, \Gamma_\sigma$
Output: $\hat{\mathcal{F}}$

for all $S \in (\Gamma_\mu \times \Gamma_\sigma)^m$ **do**
  for $i = 1$ to m **do**
    $\mu_{M_i} = \max\{\mu_{M_i} : (\dots, \mu_{M_i}, \dots) \in \mathcal{F} \cap S\}$
    $\hat{\sigma}^2_{M_i} = \max\{\sigma^2_{M_i} : (\dots, \sigma^2_{M_i}, \dots) \in \mathcal{F} \cap S\}$
  end for
  $\hat{\mathcal{F}} \leftarrow \hat{\mathcal{F}} \cup (\mu_{M_1}, \hat{\sigma}^2_{M_1}, \dots, \mu_{M_m}, \hat{\sigma}^2_{M_m})$
end for

Proof. It is no hard to see that the total operations done by Algorithm 2 is bounded by $O(m \sum_{h=1}^n |\hat{\mathcal{F}}_h|)$. From the trim operation, the size of $\hat{\mathcal{F}}_h$ is bounded by $O((n\mu_{max} \log_{1+\xi}(n\sigma_{max}))^m)$, where $\xi = \epsilon/2nm$. Suppose $x^* = (\mu^*_{M_1}, \sigma^{2*}_{M_1}, \dots, \mu^*_{M_m}, \sigma^{2*}_{M_m})$ is an optimal solution. According to Lemma 2, there exists a corresponding $\hat{x} = (\mu^*_{M_1}, \hat{\sigma}^2_{M_1}, \dots, \mu^*_{M_m}, \hat{\sigma}^2_{M_m}) \in \hat{\mathcal{F}}_n$. Using Lemma 1 and Lemma 4, the total error could be bounded.

$$\begin{aligned} obj(x^*) - obj(x) &\le \sum_{i=1}^m |\Phi(\frac{\delta - \mu^*_{M_i}}{\sqrt{\sigma^{2*}_{M_i}}}) - \Phi(\frac{\delta - \mu^*_{M_i}}{\sqrt{\hat{\sigma}^2_{M_i}}})| \\ &\le \sum_{i=1}^m 2n\xi \le \epsilon \end{aligned}$$

□

A Near-optimal Algorithm Parameterized by #Distinct Distributions

The subsection aims at an FPT approximation scheme parameterized by k, which is the number of different distributions. It will be very useful in practice, where most job processing times are i.i.d. (independent and identically distributed).

The basic idea is natural. We first guess two values for each machine, the summation of means and the summation of variances of jobs on this machine. For each guess (consisting of $2m$ values), we use an integer linear program (ILP) to test if it is possible to schedule jobs satisfying the guess. Note that, however, the possible values for the summation of means range within $[0, n\mu_{max}]$, and the summation of variances range within $[0, n\sigma_{max}]$, which are too many to guess. Following our observation from Algorithm 2, with $O(\epsilon)$-loss in the objective value it suffices to round the summation of variances, i.e., we guess which interval of the form $[(1+\xi)^{h-1}, (1+\xi)^h]$ does the summation of variances lie in. It is easy to see there are polynomially different kinds of choices for the summation of variances of jobs on each machine. Now turn to the summation of means of jobs on each machine. Similarly, by Lemma 1 it suffices to geometrically round its distance to δ. More specifically, given that the summation of variances of jobs on this machine belongs to $((1+\xi)^{h-1}, (1+\xi)^h]$ for some $h \in \mathbb{N}^+$, we guess which interval of the form $[\delta + (1+\xi)^{t-h/2}, \delta + (1+\xi)^{t+h/2}]$ or

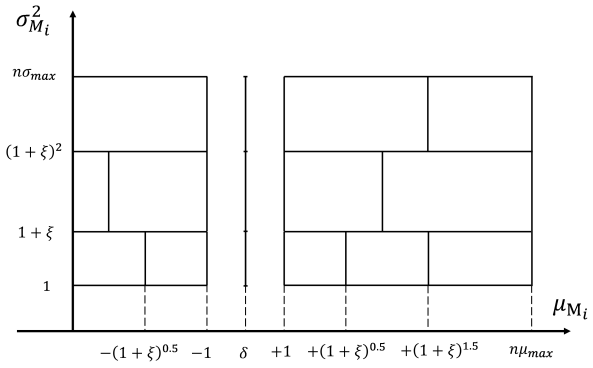

Figure 1: Illustration of Algorithm 3 for one machine case

$[\delta-(1+\xi)^{t+h/2}, \delta-(1+\xi)^{t-h/2}]$ does the summation of means belong to. We remark that within an error of $O(\epsilon)$ it is fine for a value to lie in multiple different intervals.

Suppose we guess out that for jobs on every machine i, the summation of their means belongs to interval S_{μ_i}, and the summation of their variances belongs to interval $S_{\sigma_i^2}$. For every such guess, we establish an integer linear programming – feasibility test as follows. And we denote it as ILP($\{S_\mu\}, \{S_{\sigma^2}\}$).

$$\sum_{j\in J} \mu_j \cdot x_{ij} \in S_{\mu_i} \qquad \forall i \in M \qquad \text{(2a)}$$

$$\sum_{j\in J} \sigma_j^2 \cdot x_{ij} \in S_{\sigma_i^2} \qquad \forall i \in M \qquad \text{(2b)}$$

$$\sum_{i\in M} x_{i\ell} = n_\ell \qquad \forall \ell \in K \qquad \text{(2c)}$$

$$x_{i\ell} \in \mathbb{N} \qquad \forall i \in M, \ell \in K$$

Here $K = \{1, 2, \cdots, k\}$ denotes the k different distributions, n_ℓ is the number of jobs that follow the ℓ-th normal distribution, where $\ell \in K$. The integer decision variable $x_{i,\ell}$ denotes the number of jobs with type ℓ distribution be scheduled on machine i.

We explain Constraints. First notice that Eq (2c) enforces that all jobs are scheduled. Eq (2a) and Eq (2b) ensure the summation of means/variance for jobs be scheduled on machine i belongs to interval S_{μ_i} are $S_{\sigma_i^2}$ respectively.

If the above ILP($\{S_\mu\}, \{S_{\sigma^2}\}$) admits a feasible solution, then there is a schedule corresponding to the guess and we say the guess is valid. Otherwise, the guess is discarded. Observe that there are in total mk different integer variables, it turns out ILP($\{S_\mu\}, \{S_{\sigma^2}\}$) can be solved efficiently using the following Lemma 5. Hence, we can find all valid guesses and select the best one. The above discussions are presented in Algorithm 3.

Lemma 5 ((Kannan 1987))**.** *An integer linear programming feasibility problem with N variables can be solved using $O(N^{O(N)}L)$ arithmetic operations, where L is the number of bits in the input.*

As a toy example, Figure 1 illustrates how Algorithm 3 works for single machine cases. The horizontal segments represent the possible range of S_μ. The vertical segments represent the possible range of S_{σ^2}.

Algorithm 3 FPT-Approximation Scheme for β-RSP

Input: $\epsilon, n, m, \{\mu_j\}, \{\sigma_j^2\}, \delta$
Output: an $(OPT-\epsilon)$ assignment

$\xi = \epsilon/3m$
$\Gamma_\sigma = \bigcup_h \{((1+\xi)^{h-1}, (1+\xi)^h]\}$
for all $(S_{\sigma_1^2}, \ldots, S_{\sigma_m^2}) \in \Gamma_\sigma^m, i \in [1, m]$ **do**
  $S_{\sigma_i^2} = [(1+\xi)^{h_i-1}, (1+\xi)^{h_i}]$
  $\Gamma^+_{\mu_i} = \bigcup_t \{[\delta + (1+\xi)^{t-\frac{h_i}{2}}, \delta + (1+\xi)^{t+\frac{h_i}{2}}]\}$
  $\Gamma^-_{\mu_i} = \bigcup_t \{[\delta - (1+\xi)^{t+\frac{h_i}{2}}, \delta - (1+\xi)^{t-\frac{h_i}{2}}]\}$
  $\Gamma_{\mu_i} = \Gamma^-_{\mu_i} \cup [\delta - (1+\xi)^{\frac{h_i}{2}}, \delta - 1] \cup [\delta] \cup [\delta+1, \delta + (1+\xi)^{\frac{h_i}{2}}] \cup \Gamma^+_{\mu_i}$
  for all $(S_{\mu_1}, \ldots, S_{\mu_m}) \in \Gamma_{\mu_1} \times \cdots \times \Gamma_{\mu_m}$ **do**
    solve ILP($\{S_\mu\}, \{S_{\sigma^2}\}$)
    if ILP($\{S_\mu\}, \{S_{\sigma^2}\}$) has solution x **then**
      $\mathcal{F} = \mathcal{F} \cup \{x\}$
    end if
  end for
end for
return $\max_{\mathbf{x}\in\mathcal{F}} obj(\mathbf{x})$

Lemma 6. *For any $S_{\sigma_i^2}, S_{\mu_i} \neq [\delta]$ and any (μ, σ^2), $(\bar{\mu}, \bar{\sigma}^2) \in S_{\mu_i} \times S_{\sigma_i^2}$, it holds that*

$$\max\{\frac{\delta-\mu}{\sqrt{\sigma^2}}/\frac{\delta-\bar{\mu}}{\sqrt{\bar{\sigma}^2}}, \frac{\delta-\bar{\mu}}{\sqrt{\bar{\sigma}^2}}/\frac{\delta-\mu}{\sqrt{\sigma^2}}\} \leq (1+\xi)^{3/2}.$$

Theorem 4. *For any $\epsilon > 0$, Algorithm 3 gives an $(OPT-\epsilon)$ solution for the problem $P|p_j \sim \mathcal{N}(\mu_j, \sigma_j^2)|Pr[C_{max} \leq \delta]$ within $O((\frac{1}{\epsilon})^{2m}(mk)^{O(mk)}L^{2m+1})$ time, where k is number of different distributions and L is the number of bits in the input.*

Proof. In Algorithm 3, we solve ILP($\{S_\mu\}, \{S_{\sigma^2}\}$) for $O(2^m(1/\xi)^{2m}\log^m(n\sigma_{max})\log^m(\delta + n\mu_{max}))$ times, where $\xi = \epsilon/3m$. Due to Lemma 5, we know that solving ILP($\{S_\mu\}, \{S_{\sigma^2}\}$) once need $O((mk)^{O(mk)}L)$ time. Combine all, the time complexity of Algorithm 3 is $O((\frac{1}{\epsilon})^{2m}(mk)^{O(mk)}L^{2m+1})$.

For machine i, if we view $\Phi(\frac{\delta-\mu_{M_i}}{\sqrt{\sigma^2_{M_i}}})$as a 2-dimensional function, where the variable $\mu_{M_i} \in S_{\mu_i}$ and $\sigma^2_{M_i} \in S_{\sigma_i^2}$, then for every S_{μ_i} and $S_{\sigma_i^2}$, the function are both monotonic on μ_{M_i} and $\sigma^2_{M_i}$. Hence, the min/max points are both located on boundary points. Suppose $(\mu^*_{M_1}, \sigma^{2*}_{M_1}, \ldots, \mu^*_{M_m}, \sigma^{2*}_{M_m})$ is an optimal solution. There exist $S^*_{\mu_i}$ and $S^*_{\sigma_i^2}$ in our partition such that $\mu^*_{M_i} \in S^*_{\mu_i}$ and $\sigma^{2*}_{M_i} \in S^*_{\sigma_i^2}$. According to Lemmas 1 and 6, the error for machine i is bounded by $(1+\xi)^{3/2} - 1$. Combining all into one, the total error is at most ϵ.

$$OPT - ALG \leq \sum_{i=1}^{m}((1+\xi)^{3/2} - 1) \leq 3m\xi = \epsilon.$$

machines	jobs	kinds of jobs	bench-mark	fixed parameter tractable approximation scheme $\xi = \epsilon/3m$											
				$\xi = 10$			$\xi = 25$			$\xi = 40$			$\xi = 50$		
m	n	k	time[s]	opt-alg		time[s]	opt-alg		time[s]	opt-alg		time[s]	opt-alg		time[s]
			avg	avg	std	avg	avg	std	avg	avg	std	avg	avg	std	avg
2	20	3	66	0.1%	0.2%	<1	0.1%	0.2%	<1	0.1%	0.2%	<1	0.1%	0.3%	<1
		5	100	0.3%	0.3%	<1	0.2%	0.3%	<1	0.3%	0.4%	<1	0.3%	0.4%	<1
		10	111	0.5%	0.4%	<1	0.4%	0.4%	<1	0.5%	0.4%	<1	0.5%	0.4%	<1
3	20	3	39	3.4%	5.4%	<1	2.4%	4.1%	<1	2.9%	4.8%	<1	3.5%	5.9%	<1
		5	41	3.8%	4.9%	<1	2.7%	3.1%	<1	3.0%	3.8%	<1	3.6%	4.5%	<1
		10	44	3.3%	4.2%	<1	3.2%	3.1%	<1	2.9%	3.1%	<1	2.9%	3.7%	<1
4	20	3	69	0.6%	1.6%	<1	0.7%	1.7%	<1	0.9%	2.2%	<1	1.0%	2.3%	<1
		5	66	0.2%	0.4%	<1	0.2%	0.4%	<1	0.2%	0.4%	<1	0.2%	0.4%	<1
		10	72	0.8%	1.0%	<1	0.9%	1.2%	<1	0.9%	1.3%	<1	0.9%	1.4%	<1
5	20	3	70	0.7%	1.9%	2	0.6%	1.7%	<1	0.6%	1.8%	<1	0.9%	2.6%	<1
		5	71	1.2%	1.6%	2	1.5%	2.2%	<1	1.6%	2.3%	<1	1.8%	2.5%	<1
		10	77	1.1%	1.2%	3	1.3%	1.5%	<1	1.4%	1.8%	<1	1.5%	2.1%	<1
6	20	3	71	1.6%	4.6%	80	1.1%	3.4%	28	1.3%	3.9%	26	1.6%	4.1%	28
		5	73	3.1%	4.8%	48	2.4%	4.0%	25	2.9%	4.9%	22	3.6%	5.8%	22
		10	74	4.1%	4.0%	62	3.2%	4.0%	43	3.5%	4.6%	41	4.5%	5.9%	41

Table 1: Simulation result of Algorithm 3

□

Getting rid of the exponential dependency on m, the number of machines, is unlikely even if we only look for an approximate solution. Chen, et al. (2014) prove that for the classical scheduling problem where each job processing time is deterministic instead of being a random variable, bounding the makespan, which is δ in our problem, up to a precision of ϵ requires a running time at least $(1/\epsilon)^{\Omega(m)}$. This classical scheduling problem is, of course, different from our stochastic problem, but it suggests that an algorithm that significantly outperforms our algorithm (in the sense that it runs in subexponential or even polynomial in m) is very unlikely to exist, as a stochastic setting is typically considered as harder than a deterministic setting.

Experiments

In the previous part, see e.g., Theorem 4, we prove Algorithm 3 is an efficient algorithm for the β-robust scheduling problem and theoretically analyze the running time and the quality of solutions. One may wonder that whether Algorithm 3 could work in practice? To investigate its actual performance, we conduct the experiment. We implement Algorithm 3 and compare it with the benchmark on the dataset.

All instances in the dataset contain 20 jobs. We set three different values on the number of job kinds, i.e., $k = 3, 5, 10$, and five different values on the number of machines, i.e., $m = 2, 3, 4, 5, 6$. For each combination of (m, k), 500 instances are generated. In each instance, the distribution of each kind of job is randomly generated, that is, the value of mean is selected uniformly at random from $[15, 25]$, and the value of variance is selected uniformly at random from $[1, 3]$. For the common due δ, we adopt the same generating method in previous work (Ranjbar, Davari, and Leus 2012; Stec et al. 2019) but with some distortion d uniformly selected form $[-5, 5]$. To be exact, the common due date δ is selected to be $\delta = \bar{\mu} + \sqrt{m\bar{\sigma}^2} + d$ where $\bar{\mu}$ is the average mean and $\bar{\sigma}^2$ is the average variance.

The benchmark is obtained through directly solving the master problem (Stec et al. 2019) which gives the optimal solution of the problem. We run Algorithm 3 under different parameter settings ($\xi = 10, 25, 40, 50$) and compare it with the benchmark from the following two aspects: the quality of solutions (i.e., the gap between the optimal solution); the running time. In our experiment, the time limit of Algorithm 3 is set to be 300 seconds. Once exceed the time limit, we stop the procedure and output the current best solution.

Procedures[1] are coded in C++ using Gurobi 9.0.3. All experiments are performed on a computer with an Intel Xeon-w2295 processor and 256GB memory.

Comparison results are summarized in Table 1. As we can see, Algorithm 3 performs pretty well in the quality of solutions. For each parameter setting ($\xi = 10, 25, 40, 50$) and combination of (m, k), the average gap between the optimal solution is at most 4.5% and the standard deviation of the gap is at most 5.9%. Meanwhile, if the parameter ξ is appropriately chosen Algorithm 3 performs much faster than the benchmark. Especially, if we set the parameter $\xi = 25$, the average running time of Algorithm 3 is less than one second for instances with $m = 2, 3, 4, 5$.

Conclusions

This paper gives the first systematic study on the approximability of β-robust scheduling on parallel machines. We pro-

[1]Codes, dataset and test results can be accessed in a publicly accessible repository at https://github.com/polyapp/beta-robust-scheduling.

vide a strong inapproximability result. Meanwhile, we extend the common FPT algorithms for deterministic scheduling problems to the robust setting by presenting the first near-optimal algorithm (with ϵ-additive error) when the maximal mean of distributions, or the number of distinct distributions is small. The theoretical results are supported by extensive experiments. While the landscape of approximability is now clear for β-robust scheduling with normally distributed jobs, the problem remains unsolved for other common distributions (e.g., Poisson, Cauchy, Gamma). It is interesting to see whether the observation that mean is more critical than variance is also true for other distributions.

Acknowledgements

Research of Liangde Tao and Guochuan Zhang was partly supported by "New Generation of AI 2030" Major Project (2018AAA0100902). Research of Lin Chen was partly supported by NSF Grant 1756014.

References

Alimoradi, S.; Hematian, M.; and Moslehi, G. 2016. Robust scheduling of parallel machines considering total flow time. *Computers & Industrial Engineering* 93: 152–161.

Bruno, J.; Downey, P.; and Frederickson, G. N. 1981. Sequencing tasks with exponential service times to minimize the expected flow time or makespan. *Journal of the ACM* 28(1): 100–113.

Chen, L.; Jansen, K.; and Zhang, G. 2014. On the optimality of approximation schemes for the classical scheduling problem. In *Proceedings of the 25th annual ACM-SIAM symposium on Discrete algorithms*, 657–668.

Chen, L.; Marx, D.; Ye, D.; and Zhang, G. 2017. Parameterized and Approximation Results for Scheduling with a Low Rank Processing Time Matrix. In *Proceedings of the 34th Symposium on Theoretical Aspects of Computer Science*, volume 66, 22:1–22:14.

Chen, L.; Xu, L.; Xu, S.; Gao, Z.; and Shi, W. 2019. Election with bribed voter uncertainty: Hardness and approximation algorithm. In *Proceedings of the 33rd AAAI Conference on Artificial Intelligence*, volume 33, 2572–2579.

Daniels, R. L.; and Carrillo, J. E. 1997. β-Robust scheduling for single-machine systems with uncertain processing times. *IIE Transactions* 29(11): 977–985.

Eberle, F.; Fischer, F.; Matuschke, J.; and Megow, N. 2019. On index policies for stochastic minsum scheduling. *Operations Research Letters* 47(3): 213–218.

Garey, M. R.; and Johnson, D. S. 2002. *Computers and intractability*, volume 29. wh freeman New York.

Glazebrook, K. D. 1979. Scheduling tasks with exponential service times on parallel processors. *Journal of Applied Probability* 16(3): 685–689.

Graham, R. L. 1969. Bounds on multiprocessing timing anomalies. *SIAM journal on Applied Mathematics* 17(2): 416–429.

Graham, R. L.; Lawler, E. L.; Lenstra, J. K.; and Kan, A. R. 1979. Optimization and approximation in deterministic sequencing and scheduling: a survey. *Annals of Discrete Mathematics* 5: 287–326.

Hiatt, L. M.; Zimmerman, T. L.; Smith, S. F.; and Simmons, R. 2009. Strengthening schedules through uncertainty analysis. In *Proceedings of the 21st International Joint Conference on Artificial Intelligence*, 175–180.

Im, S.; Moseley, B.; and Pruhs, K. 2015. Stochastic scheduling of heavy-tailed jobs. In *Proceedings of the 32nd International Symposium on Theoretical Aspects of Computer Science*, volume 30, 474–486.

Jäger, S.; and Skutella, M. 2018. Generalizing the Kawaguchi-Kyan Bound to Stochastic Parallel Machine Scheduling. In *Proceedings of the 35th Symposium on Theoretical Aspects of Computer Science*, volume 96, 43:1–43:14.

Jansen, K.; Maack, M.; and Solis-Oba, R. 2020. Structural parameters for scheduling with assignment restrictions. *Theoretical Computer Science* 844: 154–170.

Kannan, R. 1987. Minkowski's convex body theorem and integer programming. *Mathematics of Operations Research* 12(3): 415–440.

Knop, D.; and Kouteckỳ, M. 2018. Scheduling meets n-fold integer programming. *Journal of Scheduling* 21(5): 493–503.

Megow, N.; Uetz, M.; and Vredeveld, T. 2006. Models and algorithms for stochastic online scheduling. *Mathematics of Operations Research* 31(3): 513–525.

Mnich, M.; and van Bevern, R. 2018. Parameterized complexity of machine scheduling: 15 open problems. *Computers & Operations Research* 100: 254–261.

Mnich, M.; and Wiese, A. 2015. Scheduling and fixed-parameter tractability. *Mathematical Programming* 154(1-2): 533–562.

Möhring, R. H.; Schulz, A. S.; and Uetz, M. 1999. Approximation in stochastic scheduling: the power of LP-based priority policies. *Journal of the ACM* 46(6): 924–942.

Pishevar, A.; and Tavakkoi-Moghaddam, R. 2014. β-robust parallel machine scheduling with uncertain durations. *Universal Journal of Industrial and Business Management* 2(3): 69–74.

Pras, A.; Nieuwenhuis, L.; van de Meent, R.; and Mandjes, M. 2009. Dimensioning network links: a new look at equivalent bandwidth. *IEEE network* 23(2): 5–10.

Ranjbar, M.; Davari, M.; and Leus, R. 2012. Two branch-and-bound algorithms for the robust parallel machine scheduling problem. *Computers & Operations Research* 39(7): 1652–1660.

Skutella, M.; Sviridenko, M.; and Uetz, M. 2016. Unrelated machine scheduling with stochastic processing times. *Mathematics of Operations Research* 41(3): 851–864.

Stec, R.; Novak, A.; Sucha, P.; and Hanzalek, Z. 2019. Scheduling Jobs with Stochastic Processing Time on Parallel Identical Machines. In *Proceedings of the 28th International Joint Conference on Artificial Intelligence*, 5628–5634.

van Bevern, R.; Niedermeier, R.; and Suchỳ, O. 2017. A parameterized complexity view on non-preemptively scheduling interval-constrained jobs: few machines, small looseness, and small slack. *Journal of Scheduling* 20(3): 255–265.

Wu, C. W.; Brown, K. N.; and Beck, J. C. 2009. Scheduling with uncertain durations: Modeling β-robust scheduling with constraints. *Computers & Operations Research* 36(8): 2348–2356.

Zhang, Y.; Shen, Z.-J. M.; and Song, S. 2018. Exact Algorithms for Distributionally β-Robust Machine Scheduling with Uncertain Processing Times. *INFORMS Journal on Computing* 30(4): 662–676.