Scheduling Jobs with Stochastic Processing Time on Parallel Identical Machines

Richard Stec[1] , **Antonin Novak**[1,2*] , **Premysl Sucha**[1] and **Zdenek Hanzalek**[1]
[1]Czech Institute of Informatics, Robotics, and Cybernetics, Czech Technical University in Prague, CZ
[2]Faculty of Electrical Engineering, Czech Technical University in Prague, CZ
antonin.novak@cvut.cz

Abstract

Many real-world scheduling problems are characterized by uncertain parameters. In this paper, we study a classical parallel machine scheduling problem where the processing time of jobs is given by a normal distribution. The objective is to maximize the probability that jobs are completed before a given common due date. This study focuses on the computational aspect of this problem, and it proposes a Branch-and-Price approach for solving it. The advantage of our method is that it scales very well with the increasing number of machines and is easy to implement. Furthermore, we propose an efficient lower bound heuristics. The experimental results show that our method outperforms the existing approaches.

1 Introduction

This paper addresses the scheduling of jobs with stochastic processing times on parallel identical machines. The aim of this problem is to assign jobs to machines such that the assignment maximizes the probability that all the jobs are completed before a given common due date. Due to the structure of the objective, the problem can be classified as a β-robust scheduling problem [Daniels and Carrillo, 1997], i.e., approaches that maximize the probability that the objective is below some threshold.

The scheduling problem is given by a set of n independent jobs J and a set of m identical machines M. In order to exclude trivial instances, we assume that $n > m$. Each job $j \in J$ is characterized by a stochastic processing time π_j. The processing time is defined by a normal distribution function $\mathcal{N}(\mu_j, \sigma_j^2)$ with mean $\mu_j \in \mathbb{N}$ and variance $\sigma_j^2 \in \mathbb{N}$. The last parameter of the problem is a common due date $\delta \in \mathbb{N}_0$. In the classical $\alpha|\beta|\gamma$ notation of [Graham *et al.*, 1979], the problem can be denoted by $P|\pi_j \sim \mathcal{N}(\mu_j, \sigma_j^2)|\Pr[C_{\max} \leq \delta]$. Therefore, the solution of the problem is related to the partitioning of jobs between the machines such that machines are similarly loaded.

To solve the problem, one can utilize the property that the sum of independent normally distributed variables has a normal distribution. Then by introducing a binary decision variable $z_{i,j}$, which is 1 if and only if job $j \in J$ is assigned to machine $i \in M$, the problem can now be formulated [Ranjbar *et al.*, 2012] as the following non-linear model

$$\max \prod_{i \in M} \Phi\left(\frac{\delta - \mu_{M_i}}{\sqrt{\sigma_{M_i}^2}}\right) \tag{1}$$

subject to

$$\sum_{i \in M} z_{i,j} = 1 \quad \forall j \in J \tag{2}$$

$$\mu_{M_i} = \sum_{j \in J} \mu_j \cdot z_{i,j} \quad \forall i \in M \tag{3}$$

$$\sigma_{M_i}^2 = \sum_{j \in J} \sigma_j^2 \cdot z_{i,j} \quad \forall i \in M, \tag{4}$$

where Φ is normalized cumulative normal distribution function with μ_{M_i} and $\sigma_{M_i}^2$ being the mean and the variance of the total processing time on machine i respectively.

1.1 Related Work

The problem stated above is a fundamental problem for many real-world applications where deterministic processing time cannot be assumed. A good example is the duration of surgeries in the operating room scheduling problem [Sagnol *et al.*, 2018]. Many problems with stochastic processing time are modeled as β-robust. For example, $P|\pi_j \sim \mathcal{N}(\mu_j, \sigma_j^2)|\Pr[C_{\max} \leq \delta]$ introduced by Ranjbar *et al.* [Ranjbar *et al.*, 2012], is based on β-robustness. In the same paper, the authors devised two Branch-and-Bound algorithms that differ in their branching schemes. The main limitation of this approach are symmetries induced by the branching scheme. The authors mitigate this issue by a suitable solution encoding and by a dominance rule; however, the results indicate that the CPU time sharply grows with a increasing number of machines. The same problem with a different objective is addressed in [Pishevar and Tavakkoi-Moghaddam, 2014]. The authors maximize the probability that the sum of job completion times will be lower than a given bound. The problem is slightly simpler since the objective function does not require the product operator. The paper proposes a mixed-integer nonlinear programming model which is approximated

*Corresponding Author

by a mixed-integer programming model. The same β-robust problem where the sum of job completion times in the objective is substituted by the sum of job completion flow times is studied by Alimoradi *et al.* [Alimoradi *et al.*, 2016]. The authors propose a Branch-and-Bound algorithm that outperforms an earlier work [Wu *et al.*, 2009] addressing the single-machine problem.

A robust scheduling problem minimizing the number of identical machines required while completing all the given jobs before a deadline is investigated in [Song *et al.*, 2018]. Unlike the above-mentioned works, the processing time is not defined via a distribution function but using intervals, similarly as in [Hamaz *et al.*, 2018]. The "price" of robustness, i.e., the number of jobs that can deviate from their nominal processing times, is defined as a parameter. The problem is solved by a Branch-and-Price procedure. Another class of robust scheduling problems, where job processing times are stochastic without any assumed form of the distribution, is known as distributionally robust. Chang *et al.* [Chang *et al.*, 2019] studies parallel machine scheduling problem which minimizes the worst-case expected total flow time out of all given probability distributions. This min-max problem is solved by a reduction to an integer second-order cone program. Unrelated machine scheduling problem with stochastic processing times was addressed by [Skutella *et al.*, 2016].

1.2 Contribution and Paper Outline

In this paper, we study an exact method for solving the problem $P|\pi_j \sim \mathcal{N}(\mu_j, \sigma_j^2)|\Pr[C_{\max} \leq \delta]$. The devised algorithm is based on the Branch-and-Price approach. In comparison with [Ranjbar *et al.*, 2012] our approach is simpler to implement and has better scalability since there are no symmetries in the exploration of the solution space. The experimental results show better results compared to Branch-and-Bound algorithms from [Ranjbar *et al.*, 2012]. Furthermore, we prove that the Pricing Problem is at least weakly $\mathcal{NP}$-hard and we show how it can be solved by mixed-integer linear programming. Last but not least, we substantially improved the initial heuristic described in [Ranjbar *et al.*, 2012]. The proposed heuristic is used in the Branch-and-Price algorithm to initialize the Master Problem.

The paper is organized as follows. Description of the Branch-and-Price algorithm is divided into three sections: decomposition, Pricing Problem and branching scheme. Section 5 compares our results with the algorithm from [Ranjbar *et al.*, 2012]. The last section concludes the work.

2 Branch-and-Price Decomposition

Branch-and-Price [Barnhart *et al.*, 1998] is one of the prominent exact methods for solving combinatorial problems. It is a branch-and-bound algorithm where each node of the search tree consists of a Linear Program (LP), which is called *Master Problem*. The Master Problem typically contains an exponential number of variables that serves as indicators, to denote whether the particular *configuration* is a part of the solution. A *configuration* represents a fragment of the complete solution. All variables are not explicitly enumerated in the LP, but rather lazily generated during the solution as needed. Generation of new configurations is an iterative process, where at each step, the dual solution of the LP with the current set of configuration is considered. With this, we ask for a cut in dual LP which is violated by this dual solution. The question which configuration (i.e., a cut in dual) might improve the current primary objective value is solved by so-called *Pricing Problem*. This method is usually known as *column generation* [Desaulniers *et al.*, 2006]. Most importantly, it can be shown [Lübbecke and Desrosiers, 2005], that it is not always necessary to generate all the variables (configurations) in order to prove the optimality of the full model. When no new configuration improving the solution can be found, a branching procedure is used to ensure an integer solution.

In the next section, we show how to reformulate the formulate problem (1)–(4) to fit the Branch-and-Price method.

2.1 Master Problem

The Master Problem utilizes the fact that maximizing the sum of logprobs maximizes the product of probabilities, hence, linearizing the objective. The configurations P in the Master Problem correspond to all possible machine configurations, i.e., a subset of jobs that are scheduled on a machine. Each configuration $k \in P$ consists of a vector of column coefficients $\boldsymbol{a}_k$ for the constraint matrix and a single objective coefficient $\log p_k$. The $\boldsymbol{a}_k$ is a characteristic vector, i.e., $a_{j,k} = 1$ if and only if job $j \in J$ is scheduled on some machine with configuration $k \in P$. The objective coefficient $\log p_k$ is given as

$$\log p_k = \log \Phi \left(\frac{\delta - \boldsymbol{\mu}^T \boldsymbol{a}_k}{\sqrt{\boldsymbol{\sigma}^T \boldsymbol{a}_k}} \right)$$

where $\boldsymbol{\mu} = (\mu_1, \ldots, \mu_n)^T$ and $\boldsymbol{\sigma} = \left(\sigma_1^2, \ldots, \sigma_n^2\right)^T$. Hence, p_k is the probability that the makespan on a machine with configuration $k \in P$ is less or equal to δ. Then, the problem (1)–(4) can be reformulated as the Master Problem as follows

$$\max \sum_{k \in P} y_k \cdot \log p_k \tag{5}$$

subject to

$$\sum_{k \in P} a_{j,k} \cdot y_k \geq 1 \quad \forall j \in J, \tag{6}$$

$$\sum_{k \in P} y_k \leq |M|, \tag{7}$$

$$y_k \geq 0 \quad \forall k \in P, \tag{8}$$

where y_k is an indicator, whether configuration $k \in P$ is selected for some machine. The constraint (6) states that each job is scheduled on some machine while (7) ensures that we use at most $|M|$ machines.

In practice, the set of all configurations P is replaced with a smaller set $P' \subseteq P$ that initially starts with $|M|$ configurations, providing an initial feasible solution. The set P' is iteratively enlarged by new configurations generated by the Pricing Problem.

2.2 Initial Heuristics

The Branch-and-Price starts with the set of configurations P' that corresponds to a feasible solution found heuristically. Our initial heuristics, called *Large Job Allocated First*

Algorithm 1 LJAF heuristics

let $\mu_1 \cdot \sigma_1^2 \geq \mu_2 \cdot \sigma_2^2 \geq \ldots \geq \mu_n \cdot \sigma_n^2$
$p_i \leftarrow 1$, $\boldsymbol{a}_i \leftarrow \mathbf{0} \quad \forall i \in M$
for $j \leftarrow 1$ **to** n **do**
  $i^\star \leftarrow \arg\max_{i \in M} p_i$
  $a_{i^\star,j} \leftarrow 1$
  $p_i \leftarrow \Phi\left(\frac{\delta - \boldsymbol{\mu}^T \boldsymbol{a}_{i^\star}}{\sqrt{\boldsymbol{\sigma}^T \boldsymbol{a}_{i^\star}}}\right)$
end for
return $p_i, \boldsymbol{a}_i \quad \forall i \in M$

(LJAF), is shown in Algorithm 1. It first sorts jobs in a non-increasing order of products of means and variances $\mu_j \cdot \sigma_j^2$. The ties are broken by larger μ_j. In each step, a job is taken and assigned to the machine $i \in M$ with the currently largest probability p_i.

We note that according to our experiments in Section 5, Algorithm 1 produces better initial solutions than the lower bound heuristics proposed by [Ranjbar *et al.*, 2012].

3 Pricing Problem

3.1 Problem Statement and Complexity

From the dual problem of (5)–(8) it can be derived, that the Pricing Problem takes the form of

$$\boldsymbol{x}^\star = \arg \max_{\boldsymbol{x} \in \{0,1\}^n} \log \Phi\left(\frac{\delta - \boldsymbol{\mu}^T \boldsymbol{x}}{\sqrt{\boldsymbol{\sigma}^T \boldsymbol{x}}}\right) + \boldsymbol{\psi}^T \boldsymbol{x}. \qquad (9)$$

The formulation is based on binary decision variable x_j determining whether job j will be selected into the new configuration or not. Furthermore, $\boldsymbol{\psi} \geq \mathbf{0}$ are *negative* dual prices associated with constraints (6). Since $\psi_j \geq 0$ and $\mu_j \geq 0$, it can be seen, that the Pricing Problem is balancing the gain ψ_j from taking job $j \in J$ into the new configuration and the loss, proportional to μ_j. We note that without loss of generality we may assume that $\boldsymbol{\sigma}^T \boldsymbol{x} > 0$ as in any optimal solution at least one job is allocated on each machine.

In the rest of the subsection we show, that the Pricing Problem is at least weakly $\mathcal{NP}$-hard by the reduction from KNAPSACK PROBLEM [Kellerer *et al.*, 2004]:

Definition (KNAPSACK PROBLEM)**.** *The instance of the problem and the solution is given as follows:*

INPUT: $S = \{1, 2, \ldots, n\}$, $\boldsymbol{v} = (v_1, v_2, \ldots, v_n) \in \mathbb{N}^n$, $\boldsymbol{w} = (w_1, w_2, \ldots, w_n) \in \mathbb{N}^n$ *and* $C, k \in \mathbb{N}$.

OUTPUT: *Is there* $S' \subseteq S$ *such that* $\sum_{i \in S'} v_i \geq k$ *and* $\sum_{i \in S'} w_i \leq C$*?*

Proposition 1. *The Pricing Problem is at least weakly $\mathcal{NP}$-hard.*

Proof. The general idea for the reduction is the following. We set $\boldsymbol{\psi} = \boldsymbol{v}$, $\boldsymbol{\mu} = \boldsymbol{w}$ and $\delta = C + \Delta$ where $\Delta > 0$ is some small non-negative number. Let $\Phi_\sigma(\cdot)$ denote a cumulative normal distribution with zero mean and variance σ^2. Then it can be seen, as $\sigma^2 \to 0$, the $\log \Phi_\sigma(\delta - \boldsymbol{\mu}^T \boldsymbol{x})$ approaches to zero, when $\delta > \boldsymbol{\mu}^T \boldsymbol{x}$; see Figure 1. Hence, if we would set $\boldsymbol{\sigma} \to \mathbf{0}$, $\log \Phi(\cdot)$ serves as a penalty function reflecting whether $\boldsymbol{\mu}^T \boldsymbol{x} \leq \delta$ is violated.

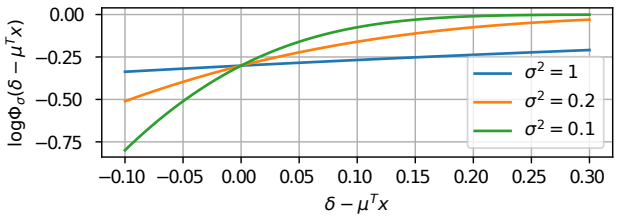

Figure 1: Logarithm of the cumulative normal distribution function converging towards 0 for $\delta - \boldsymbol{\mu}^T \boldsymbol{x} > 0$ as a function of σ^2.

However, since $\boldsymbol{\sigma} > \mathbf{0}$ has to hold, we will set it close to zero such that $\log \Phi(\cdot)$ is *almost* zero within the controllable error ϵ for $\boldsymbol{\mu}^T \boldsymbol{x} \leq C + \Delta$ for some small $\Delta > 0$ and a large negative number for $C + \Delta < \boldsymbol{\mu}^T \boldsymbol{x}$ such that every optimal solution of the Pricing Problem represents a feasible solution for KNAPSACK PROBLEM.

Assuming σ_i being a constant equal to o, we look for $1 > o > 0$, such that

$$-\log \Phi\left(\frac{\delta - \boldsymbol{w}^T \boldsymbol{x}}{\sqrt{o \cdot \mathbf{1}^T \boldsymbol{x}}}\right) < \epsilon \qquad (10)$$

where $\delta = C + \Delta$ with $\Delta = 10^{-\mathbf{1}^T \boldsymbol{w}}$ and $\epsilon = 10^{-\mathbf{1}^T \boldsymbol{v}}$ for any $\boldsymbol{x}$ such that $\boldsymbol{w}^T \boldsymbol{x} \leq C$. Hence, solving for o we get

$$o < \frac{10^{-2 \cdot \mathbf{1}^T \boldsymbol{w}}}{n \cdot (\Phi^{-1}(10^{-\epsilon}))^2}$$

where Φ^{-1} is quantile function of $\mathcal{N}(0, 1)$. Therefore, we set $\boldsymbol{\sigma} = o \cdot \mathbf{1}$ for the reduction. Note that the assumption of setting $o < 1$ is not crucial for the reduction; instead of it, we could multiply δ and $\boldsymbol{\mu}$ by a factor $\lceil o^{-1} \rceil$ and let $\boldsymbol{\sigma} = \mathbf{1}$ to achieve the same controllable error ϵ.

Next, we ensure, that $\boldsymbol{\sigma}$ is so small, such that every optimal solution of the Pricing Problem represents a feasible solution for KNAPSACK PROBLEM. Hence, we need to ensure, that the smallest violation of $\boldsymbol{w}^T \boldsymbol{x} \leq C$ dominates the largest possible gain of $\boldsymbol{v}^T \boldsymbol{x}$. That can be formulated as

$$\log \Phi\left(\frac{\Delta - 1}{\sqrt{o \cdot n}}\right) < -\boldsymbol{v}^T \mathbf{1} \qquad (11)$$

from which we get

$$o < \frac{(\Delta - 1)^2}{n \cdot \Phi^{-1}(\epsilon)^2}.$$

Therefore, we can see that the choice

$$o = \frac{10^{-2 \cdot \mathbf{1}^T \boldsymbol{w}}}{2n \cdot (\Phi^{-1}(10^{-\epsilon}))^2}$$

suffices.

It is easy to see that the decision version of the Pricing Problem in $\mathcal{NP}$. Hence, we are left to show that the Pricing Problem has a solution greater than some $k' > 0$ if and only if KNAPSACK PROBLEM has solution greater or equal than $k > 0$, such that $k' = k - \epsilon$.

– KNAPSACK PROBLEM → PRICING PROBLEM
Assume knapsack solution S' given as characteristic vector $\boldsymbol{s}'$. We have that $\boldsymbol{v}^T\boldsymbol{s}' = \boldsymbol{\psi}^T\boldsymbol{s}' \geq k$. Furthermore, since $\boldsymbol{\mu}^T\boldsymbol{s}' = \boldsymbol{w}^T\boldsymbol{s}' \leq C$, we have that

$$\log\Phi\left(\frac{\delta - \boldsymbol{\mu}^T\boldsymbol{s}'}{\sqrt{\boldsymbol{\sigma}^T\boldsymbol{s}'}}\right) \geq \log\Phi\left(\frac{\Delta}{\sqrt{o \cdot \mathbf{1}^T\boldsymbol{s}'}}\right) \geq$$
$$\geq \log\Phi\left(\frac{\Delta}{\sqrt{o \cdot n}}\right) > -\epsilon$$

where the first inequality follows from that S' is a feasible solution, the second from the fact that $o < 1$. The last inequality follows from the choice of $\boldsymbol{\sigma}$. Therefore, the the Pricing Problem has solution with objective at least $k' = k - \epsilon$.

– PRICING PROBLEM → KNAPSACK PROBLEM
Let us assume a solution $\boldsymbol{x}^\star$ of the Pricing Problem with objective greater than some $k' > 0$. Let us define a solution of KNAPSACK PROBLEM as $S' = \{i \,|\, \forall x_i^\star = 1\}$ with the corresponding characteristic vector $\boldsymbol{s}' = \boldsymbol{x}^\star$. We need to show that (i) $\boldsymbol{w}^T\boldsymbol{s}' \leq C$ and (ii) $\boldsymbol{v}^T\boldsymbol{s}' \geq k$.

(i) By contradiction. Assume that $\boldsymbol{w}^T\boldsymbol{s}' \geq C + 1$. Then by (11) we have that

$$\log\Phi\left(\frac{\delta - \boldsymbol{\mu}^T\boldsymbol{s}'}{\sqrt{\boldsymbol{\sigma}^T\boldsymbol{s}'}}\right) + \boldsymbol{\psi}^T\boldsymbol{s}' \leq$$
$$\leq \log\Phi\left(\frac{\Delta - 1}{\sqrt{o \cdot n}}\right) + \boldsymbol{v}^T\boldsymbol{s}' < -\boldsymbol{v}^T\mathbf{1} + \boldsymbol{v}^T\boldsymbol{s}' \leq 0,$$

which contradicts the assumption $k' > 0$.

(ii) By (10) we have that the log term is bounded by

$$0 > \log\Phi\left(\frac{\delta - \boldsymbol{\mu}^T\boldsymbol{x}^\star}{\sqrt{\boldsymbol{\sigma}^T\boldsymbol{x}^\star}}\right) > -\epsilon.$$

Therefore

$$\boldsymbol{\psi}^T\boldsymbol{x}^\star > k - \epsilon.$$

However, since $\boldsymbol{\psi}^T\boldsymbol{x}^\star = \boldsymbol{v}^T\boldsymbol{s}' \in \mathbb{N}$, $k \in \mathbb{N}$ and $0 < \epsilon < 1$, then it follows that $\boldsymbol{v}^T\boldsymbol{s}' \geq k$.

□

3.2 Exact Algorithm

To find an exact solution of the Pricing Problem, we adapt a trick used for solving stochastic knapsack problems [Morton and Wood, 1998]. We observe that solving (9) is equivalent to the following

$$\max_{v \in V} \max_{\boldsymbol{x} \in \{0,1\}^n} \log\Phi\left(\frac{\delta - \boldsymbol{\mu}^T\boldsymbol{x}}{\sqrt{v}}\right) + \boldsymbol{\psi}^T\boldsymbol{x} \qquad (12)$$

subject to

$$v = \boldsymbol{\sigma}^T\boldsymbol{x} \qquad (13)$$

where $V = \{\underline{v}, \underline{v}+1, \ldots, \overline{v}-1, \overline{v}\}$ such that $\underline{v}$ and $\overline{v}$ is a lower and upper bound on $\boldsymbol{\sigma}^T\boldsymbol{x}$ respectively. The advantage is that for any fixed v, the expressions $\delta/\sqrt{v}$ and $\boldsymbol{\mu}/\sqrt{v}$ become constants. Hence, for every $v \in V$, we optimize

$$\max_{\boldsymbol{x} \in \{0,1\}^n} \log\Phi\left(\frac{\delta}{\sqrt{v}} - \frac{1}{\sqrt{v}}\boldsymbol{\mu}^T\boldsymbol{x}\right) + \boldsymbol{\psi}^T\boldsymbol{x}$$

which can be done by formulating it as a Mixed-Integer Programming (MIP) where $\log\Phi(\cdot)$ is approximated by a piecewise linear function. The formulation can be solved by a special simplex method for piecewise linear functions implemented in Gurobi solver [Gurobi, 2019]. Furthermore, we impose the constraint on the minimum and the maximum number of jobs that can be scheduled on a machine derived by [Ranjbar *et al.*, 2012] to speed up the solution.

The objective value (9) of an optimal solution $\boldsymbol{x}^\star$ is compared with γ, which is the dual price for constraint (7). When the objective (9) is greater than γ, the configuration is added to the Master Problem and resolved. When it is less or equal to γ, the Master Problem is solved optimally.

However, we note that this approach for solving the Pricing Problem is efficient only when bounds on $\boldsymbol{\sigma}^T\boldsymbol{x}$ are tight, i.e., $|V| \approx 20$. For other cases, we fall back to a linearized model of (1)–(4) for a single machine, except two differences:

– the term $\boldsymbol{\psi}^T\boldsymbol{z}$ is included in the objective
– the constraint (2) relaxed.

In the next section, we describe a branching procedure and the rest of the Branch-and-Price algorithm.

4 Branching Scheme

When the optimal solution of the Master Problem is not integer, a branching procedure is used. Unlike the original problem formulation (1)–(4) the branching scheme in this Branch-and-Price is chosen such that it does not consider the assignment to machines. The reason is not to introduce symmetries caused by the fact that the machines are identical. Therefore the branching directly uses decision variables of the Pricing Problem. The scheme selects a couple of jobs (a, b) for which one branch enforces that both jobs must always be scheduled on the same machine (i.e., $x_a = x_b$), while the other branch defines the opposite (i.e., $x_a + x_b \leq 1$).

The couple of jobs for branching is chosen with respect to $\boldsymbol{y}^\star$, i.e., the last result of the Master Problem. Indices of jobs j are sorted in ascending order with respect to $\max y_k : a_{j,k} = 1$. Then the algorithm first tries the first job as a with the second one as b. If this couple is not selected, then it tries the first one with the third one, etc. When the second index gets to the end the second one is taken as a and the third one as b, etc. Each couple (a, b) is tested whether it was not selected before in the active branch and whether some of its branches does not cause a conflict with the previous branching decisions stored in set B. An element of the set is defined as tuple (a, b, e), where e is $true$ if jobs a and b must be scheduled on the same machine and $false$ otherwise. The branching scheme takes the first non-conflicting couple (a, b).

Finally, it remains to explain the way the algorithm tests whether a set of branching decisions is non-conflicting. If B is a set of non-conflicting branching decisions, it means that there must be at most $|M|$ configurations that covers all jobs in J. If there are no such configurations the Master Problem cannot be feasible. Then the test can be formulated with the following constraints

$$\sum_{i \in M} x_{j,i} \geq 1 \quad \forall j \in J, \qquad (14)$$

machines	jobs	Branch-and-Price runtime [s]	Branch-and-Price nodes [–]	Branch-and-Price timeouts [%]	Branch-and-Bound [Ranjbar *et al.*, 2012] runtime [s]	Branch-and-Bound [Ranjbar *et al.*, 2012] nodes [–]	Branch-and-Bound [Ranjbar *et al.*, 2012] timeouts [%]
$\|M\| = 2$	$n = 14$	273.9 ($\pm$164.3)	1.1 ($\pm$0.2)	0	**0.2** ($\pm$0.0)	3.6K ($\pm$2.3K)	0
	$n = 16$	753.5 ($\pm$502.2)	1.2 ($\pm$0.7)	0	**0.7** ($\pm$0.1)	15.5K ($\pm$7.6K)	0
	$n = 18$	1.53K ($\pm$551.4)	1.3 ($\pm$0.7)	5	**3.0** ($\pm$0.3)	31.5K ($\pm$32.4K)	0
	$n = 20$	2.32K ($\pm$485.2)	1.1 ($\pm$0.2)	20	**12.3** ($\pm$0.9)	159.0K ($\pm$133.7 K)	0
$\|M\| = 4$	$n = 14$	26.8 ($\pm$18.4)	5.2 ($\pm$10.5)	0	**22.3** ($\pm$8.6)	253.2K ($\pm$192.6K)	0
	$n = 16$	**85.7** ($\pm$115.6)	9.4 ($\pm$14.3)	0	383.9 ($\pm$109.5)	5.7M ($\pm$3.2M)	0
	$n = 18$	**496.5** ($\pm$574.4)	20.0 ($\pm$20.1)	0	–	–	100
	$n = 20$	**1.17K** ($\pm$893.9)	47.4 ($\pm$68.9)	5	–	–	100
$\|M\| = 6$	$n = 14$	10.9 ($\pm$4.6)	1.7 ($\pm$2.3)	0	**9.6** ($\pm$5.2)	100.1K ($\pm$101.6K)	0
	$n = 16$	**19.3** ($\pm$9.2)	3.3 ($\pm$4.9)	0	378.7 ($\pm$224.7)	4.7M ($\pm$4.8M)	0
	$n = 18$	**49.8** ($\pm$30.4)	9.6 ($\pm$13.8)	0	2.2K ($\pm$587.0)	19.2M ($\pm$26.8M)	90
	$n = 20$	**217.1** ($\pm$295.5)	54.0 ($\pm$88.0)	15	–	–	100

Table 1: Comparison with an existing exact approach.

$$x_{a,i} = x_{b,i} \quad \forall (a, b, true) \in B, \forall i \in M, \tag{15}$$

$$x_{a,i} + x_{b,i} \leq 1 \quad \forall (a, b, false) \in B, \forall i \in M, \tag{16}$$

where $x_{j,i}$ is a binary decision variable equal to one if job j is in configuration i and zero otherwise. Constraints (14) guarantee that all jobs are selected at least once and the rest of the constraints represent the branching decisions form B. In fact, the structure of the above formulation shows that it is a decision version GRAPH COLORING problem. The vertices of the graph are individual jobs. Two vertices a and b are merged into a single one if there is a branching decision $(a, b, true)$ in B. Branching decisions $(a, b, false)$ correspond to edges of the graph. The problem tests whether the graph can be colored with $|M|$ colors. Even though the decision problem is $\mathcal{NP}$-complete for $|M| \geq 3$, the experiments show that its computational difficulty is negligible compared to the difficulty of the Pricing Problem.

The test whether jobs (a, b) are admissible for branching is carried out as follows. The algorithm creates two sets $B^+ = B \cup \{(a, b, true)\}$ and $B^- = B \cup \{(a, b, false)\}$. Both are tested using formulation (14)–(16). Then, (a, b) is admissible for braching if both B^+ and B^- are feasible. Moreover, the configurations found by formulation (14)–(16) are added to the Master Problem in order to preserve its feasibility after removing configurations conflicting with an appropriate branching decision.

After branching, constraints defined in the new B need to be propagated to the Pricing Problem such that generated configurations are consistent with respect to the given branching decisions. Since we solve the Pricing Problem as a MIP, these constraints are naturally incorporated by putting equivalent constraints into the model (12)–(13) without affecting the structure of the problem.

5 Experimental Results

5.1 Instances and Testing Environment

All the instances were generated according to the methodology proposed by [Ranjbar *et al.*, 2012]. We chose 2, 4 and 6 parallel machines with 14,16,18 and 20 jobs to schedule. Parameter c, which influences the range in which the variances were generated, had two possible values, specifically 0.25 and 0.75. For each combination of the parameters, 10 instances were created, resulting in a total of 240 instances.

Both the Branch-and-Price and Branch-and-Bound [Ranjbar *et al.*, 2012] were implemented in C++. The Master and Pricing problems were solved using Gurobi 8.0 solver. The program was run on a server with two Intel Xeon E5 2.60 GHz processors, 252 GB RAM memory and a 64-bit operating system. The Branch-and-Price method was allowed to use up to 16 CPU cores. Since the Branch-and-Bound method proposed in [Ranjbar *et al.*, 2012] is not designed to use more than single core, it was run only on a single CPU. Time limit for each instance was set to 3600 seconds.

5.2 Branch-and-Price and LJAF Heuristics

The experimental evaluation is focused on the comparison of our algorithm with the best of the Branch-and-Bound algorithms introduced by [Ranjbar *et al.*, 2012]. The results for different benchmark sets are summarized in Table 1. Each benchmark set is characterized by the number of jobs n and machines $|M|$. Each set has 10 instances generated with $c = 0.25$ and 10 instances with $c = 0.75$, i.e., 20 all together.

For the Branch-and-Price algorithm, we provide the mean running time in seconds, mean number of expanded nodes in the branching scheme and percentage of instances that did not finish before the specified time limit. Note that if the value of expanded nodes for some instance is 1, it means that no branching was necessary and the instance was solved in the initial node. For both the runtimes and expanded nodes we also provide standard deviation. The Branch-and-Bound algorithm is also described by the mean runtime in seconds, then by the the number of expanded nodes in total, and again by percentage of instances that were not solved before the given time limit. Again, runtimes and the number of expanded nodes are given with their corresponding standard deviations.

Comparing the results for instances with two parallel machines (i.e., $|M| = 2$) we observe that although our algorithm (i.e., Branch-and-Price) almost did not branch, it performed significantly worse than the reference Branch-and-

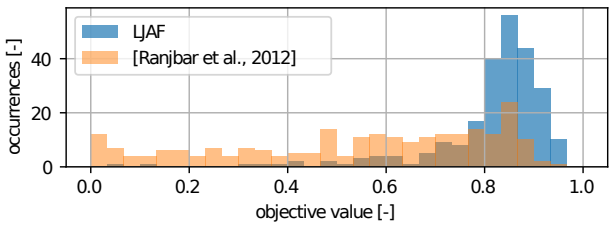

Figure 2: Comparison of objective distributions for different initial heuristics.

Bound. The reason is discussed later in this section. However, we still manage to solve most of the instances below the given time limit.

For 4 and 6 machines, our algorithm outperforms the Branch-and-Bound in most cases. We observe that the Branch-and-Bound is faster for instances with 14 jobs; however, the difference is already minimal. For instances with a higher number of jobs, our approach becomes dominant. Compared to the Branch-and-Bound, we were able to drastically reduce the running times even in cases with 18 and 20 jobs, where the other algorithm had 90% timeout rate or higher. Our algorithm suffered from several timeouts on these instances as well, but only for cases with 20 jobs and even in those we did not have more than 3 unsolved instances.

Regarding the instances with two parallel machines, it was revealed that the Pricing Problem generates new configurations for the Master Problem that assign approximately $n/|M|$ jobs to each configuration and rarely adds a few more. This implies that instances with a lower number of available machines and a higher number of jobs (i.e., where the number of jobs to schedule to a single machine is also higher) have more possible combinations of configurations. Further inspection had indeed shown that for the instances with two machines, the Branch-and-Price generated in general more configurations per instance than for instances with more parallel machines with the same number of jobs. Moreover, the mean time spent for generation of one such configuration in the Pricing Problem was also higher. Therefore, the combination of both these factors is responsible for longer runtime. However, a possible improvement would be not to solve the Pricing Problem to optimality in every iteration of the column generation and thus save some computational time. Hence, a heuristic method for Pricing Problem can be employed to improve the performance.

Finally, the comparison of the initial heuristic solutions by [Ranjbar *et al.*, 2012] and our LJAF approach is shown in Figure 2. The histogram depicts the distribution of initial lower bounds produced by these two heuristics, the higher, the better. It can be seen that our approach is more likely to provide better initial bound than the former one. This is most likely due to the similarity with the LPT (*largest processing time first*) rule used for deterministic problem $P||C_{\max}$, where it acts as a $4/3$-approximation algorithm.

5.3 Non-linear Mixed-Integer Model

For further comparison, we have also developed a non-linear model, i.e., the equivalent of model (1)–(4). However, since cumulative normal distribution function Φ does not have an analytical form, for $x \geq 0$ we use the approximation

$$\Phi(x) \approx 0.5 + 0.5(1 - \frac{1}{30}(7e^{-0.5x^2} + 16e^{-x^2(2-\sqrt{2})} + \\ + (7 + 0.25\pi x^2)e^{-x^2}))^{1/2}.$$

To further improve the performance of the model, we have added a symmetry breaking constraint in form of

$$\mu_{M_1} \geq \mu_{M_2} \geq \ldots \geq \mu_{M_m}$$

to break the symmetry between identical machines. The results show that for instances with $|M| = 4$ machines and $n = 10$ jobs, the solution with SCIP 6.0.0 [Gleixner *et al.*, 2018] non-linear mixed-integer solver takes about 3000 s. Therefore, we have excluded this method from further experiments. During the experiments, we observed that results were similar for different approximations of Φ.

6 Conclusions

This research work reflects a growing interest in stochastic scheduling problems. In this paper, we concentrated on a stochastic variant of $P||C_{\max}$ where the processing time of each job is given by a normal distribution function. This variant is formulated as a β-robust scheduling problem where the objective is to maximize the probability the schedule is completed before a given common due date.

The main contribution of our work is a new algorithm significantly outperforming a Branch-and-Bound algorithm proposed in [Ranjbar *et al.*, 2012]. The main reason for the better performance is a better elimination of symmetrical solution in the solution search space. Nevertheless, the Branch-and-Bound algorithm of Ranjbar *et al.* performs better on the data set with two machines. This case would probably need a different approach to the pricing problem since the experimental results indicate that this is the bottleneck of the algorithm. Other data sets show that our algorithm scales significantly better and can solve larger instances in a shorter time.

For future work, we suggest generalizing our approach to other families of probability distributions. For distributions that have known distributions of their sum, such as the ones that are closed under convolution (e.g., Poisson, Cauchy, Gamma) the current form of the Pricing Problem can be modified to handle them. For others, such as lognormal distributions, approximations of their sum exist [Mehta *et al.*, 2007]. To handle general cases, sampling methods from stochastic knapsack problems [Morton and Wood, 1998] might be applicable to modify the structure and solution of the Pricing Problem.

Acknowledgements

This work was supported by the EU and the Ministry of Industry and Trade of the Czech Republic under the Project OP PIK CZ.01.1.02/0.0/0.0/15_019/0004688 and by the Grant Agency of the Czech Technical University in Prague, grant No. SGS19/175/OHK3/3T/13.

References

[Alimoradi *et al.*, 2016] Soroush Alimoradi, Milad Hematian, and Ghasem Moslehi. Robust scheduling of parallel machines considering total flow time. *Computers & Industrial Engineering*, 93:152 – 161, 2016.

[Barnhart *et al.*, 1998] Cynthia Barnhart, Ellis L. Johnson, George L. Nemhauser, Martin W.P. Savelsbergh, and Pamela H. Vance. Branch-and-price: Column generation for solving huge integer programs. *Operations research*, 46(3):316–329, 1998.

[Chang *et al.*, 2019] Zhiqi Chang, Jian-Ya Ding, and Shiji Song. Distributionally robust scheduling on parallel machines under moment uncertainty. *European Journal of Operational Research*, 272(3):832 – 846, 2019.

[Daniels and Carrillo, 1997] Richard L. Daniels and Janice E. Carrillo. β-robust scheduling for single-machine systems with uncertain processing times. *IIE Transactions*, 29(11):977–985, Nov 1997.

[Desaulniers *et al.*, 2006] Guy Desaulniers, Jacques Desrosiers, and Marius M. Solomon. *Column generation*, volume 5. Springer Science & Business Media, 2006.

[Gleixner *et al.*, 2018] Ambros Gleixner, Michael Bastubbe, Leon Eifler, Tristan Gally, Gerald Gamrath, Robert Lion Gottwald, Gregor Hendel, Christopher Hojny, Thorsten Koch, Marco E. Lübbecke, Stephen J. Maher, Matthias Miltenberger, Benjamin Müller, Marc E. Pfetsch, Christian Puchert, Daniel Rehfeldt, Franziska Schlösser, Christoph Schubert, Felipe Serrano, Yuji Shinano, Jan Merlin Viernickel, Matthias Walter, Fabian Wegscheider, Jonas T. Witt, and Jakob Witzig. The SCIP Optimization Suite 6.0. ZIB-Report 18-26, Zuse Institute Berlin, July 2018.

[Graham *et al.*, 1979] Ronald L. Graham, Eugene L. Lawler, Jan Karel Lenstra, and Alexander H.G.Rinnooy Kan. Optimization and approximation in deterministic sequencing and scheduling: a survey. In P.L. Hammer, E.L. Johnson, and B.H. Korte, editors, *Discrete Optimization II*, volume 5 of *Annals of Discrete Mathematics*, pages 287 – 326. Elsevier, 1979.

[Gurobi, 2019] Gurobi. Constraints. http://www.gurobi.com/documentation/8.1/refman/constraints.html, 2019. Accessed Februrary 12, 2019.

[Hamaz *et al.*, 2018] Idir Hamaz, Laurent Houssin, and Sonia Cafieri. A robust basic cyclic scheduling problem. *EURO Journal on Computational Optimization*, 6(3):291–313, Sep 2018.

[Kellerer *et al.*, 2004] Hans Kellerer, Ulrich Pferschy, and David Pisinger. Introduction to np-completeness of knapsack problems. In *Knapsack problems*, pages 483–493. Springer, 2004.

[Lübbecke and Desrosiers, 2005] Marco E. Lübbecke and Jacques Desrosiers. Selected topics in column generation. *Operations research*, 53(6):1007–1023, 2005.

[Mehta *et al.*, 2007] Neelesh B. Mehta, Jingxian Wu, Andreas F. Molisch, and Jin Zhang. Approximating a sum of random variables with a lognormal. *IEEE Transactions on Wireless Communications*, 6(7):2690–2699, July 2007.

[Morton and Wood, 1998] David P. Morton and R. Kevin Wood. On a stochastic knapsack problem and generalizations. In *Advances in computational and stochastic optimization, logic programming, and heuristic search*, pages 149–168. Springer, 1998.

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

[Ranjbar *et al.*, 2012] Mohammad Ranjbar, Morteza Davari, and Roel Leus. Two branch-and-bound algorithms for the robust parallel machine scheduling problem. *Computers & Operations Research*, 39(7):1652 – 1660, 2012.

[Sagnol *et al.*, 2018] Guillaume Sagnol, Christoph Barner, Ralf Borndörfer, Mickaël Grima, Matthes Seeling, Claudia Spies, and Klaus Wernecke. Robust allocation of operating rooms: A cutting plane approach to handle lognormal case durations. *European Journal of Operational Research*, 271(2):420 – 435, 2018.

[Skutella *et al.*, 2016] Martin Skutella, Maxim Sviridenko, and Marc Uetz. Unrelated machine scheduling with stochastic processing times. *Mathematics of operations research*, 41(3):851–864, 2016.

[Song *et al.*, 2018] Guopeng Song, Daniel Kowalczyk, and Roel Leus. The robust machine availability problem – bin packing under uncertainty. *IISE Transactions*, 50(11):997–1012, 2018.

[Wu *et al.*, 2009] Christine Wei Wu, Kenneth N. Brown, and J. Christopher Beck. Scheduling with uncertain durations: Modeling β-robust scheduling with constraints. *Computers & Operations Research*, 36(8):2348 – 2356, 2009.