Distributionally robust scheduling algorithms for total flow time minimization on parallel machines using norm regularizations

Antonin Novak[a,c,*], Andrzej Gnatowski[b], Premysl Sucha[a]

[a] *Czech Institute of Informatics, Robotics and Cybernetics, Czech Technical University in Prague, CZ*
[b] *Department of Control Systems and Mechatronics, Wrocław University of Science and Technology, PL*
[c] *Faculty of Electrical Engineering, Czech Technical University in Prague, CZ*

Abstract

In this paper, we study a distributionally robust parallel machines scheduling problem, minimizing the total flow time criterion. The distribution of uncertain processing times is subject to ambiguity belonging to a set of distributions with constrained mean and covariance. We show that the problem can be cast as a deterministic optimization problem, with the objective function composed of an expectation and a regularization term given as an ℓ_p norm. The main question we ask and answer is whether the particular choice of the used ℓ_p norm affects the computational complexity of the problem and the robustness of its solution. We prove that if durations of the jobs are independent, the solution in terms of any ℓ_p norm can be solved in a pseudopolynomial time, by the reduction to a non-linear bipartite matching problem. We also show an efficient, polynomial-time algorithm for ℓ_1 case. Furthermore, for instances with dependent durations of the jobs, we propose computationally efficient formulation and an algorithm that uses ℓ_1 norm. Moreover, we identify a class of covariance matrices admitting a faster, polynomial-time algorithm. The computational experiments show that the proposed algorithms provide solutions with a similar quality to the existing algorithms while having significantly better computational complexities.

Keywords: scheduling, distributionally robust optimization, uncertain processing time, total flow time, computational complexity

1. Introduction

Real-life processes often involve uncertainty, i.e., values of some parameters of the system are not known beforehand. Provided a sufficient quantity of empirical data, it is usually possible to build models describing how the uncertain parameters relate to each other and what values they can attain. Then, the description of the uncertainty can be utilized during the optimization, following various paradigms, for instance, Robust Optimization (RO, optimizing for the worst-case realization),

*Corresponding author
Email address: antonin.novak@cvut.cz (Antonin Novak)

Stochastic Optimization (SO, optimizing the expected value), or DRO (distributionally robust optimization, optimizing for the worst-case expectation) that we address in this paper.

An example of such a process is unit testing, a crucial part of modern software development [33]. Each time the unit tests are executed, they cover slightly different parts of the source code, resulting in random pass/failure ratios and the run times. Since the test batches are performed repeatedly, one can build empirical distributions of the aforementioned parameters. Moreover, the tests are usually not independent—a failure of one test might lead to an automatic failure of an entire batch. Thus, it is beneficial to model the uncertainty, e.g., with multivariate distributions. When the number of tests is large, they are scheduled and performed in parallel, using a cluster of servers. When the computing nodes are identical, the problem can be modeled as parallel identical machines scheduling with uncertain job durations. The total flow time is usually chosen as the objective function [20], as minimizing average user waiting time ensures the tests can be executed frequently.

1.1. Problem statement

In this work, we focus on a distributionally robust scheduling introduced in [8]. The problem considers n jobs $\mathcal{J} = \{1, 2, \ldots, n\}$ that need to be scheduled on m identical machines $\mathcal{M} = \{1, 2, \ldots, m\}$. Each job is available at time 0 and can be processed on any machine, while preemption is not allowed. Job $j \in \mathcal{J}$ is characterized by uncertain processing time $\tilde{p}_j \in \mathbb{R}_+$. The processing times can be expressed as random vector $\tilde{\boldsymbol{p}} \in \mathbb{R}_+^n$ subject to an ambiguous probability distribution $P \in \mathcal{D}$, $\mathcal{D} \subseteq \mathcal{P}_0(\mathbb{R}_+^n)$, where $\mathcal{P}_0(\mathbb{R}_+^n)$ is the set of all probability distributions on $\mathbb{R}_+^n$. Finally, the objective is to minimize the worst-case expected total flow time f, i.e., a sum of completion times of jobs ($\sum C_j$ or TFT).

A solution to this problem is a schedule that assigns the jobs to the machines and sequences the assigned jobs on each of them. In [8], it was shown that the representation of the solution can completely disregard the assignment of a job to a specific machine. This property leads to a concise representation of the solution by vector $\boldsymbol{\pi} = (\pi_1, \ldots, \pi_n) \in \mathbb{Z}_+^n$ of n positive integers. In this representation, $\pi_j = l$ if and only if job j is scheduled as the l-th job from the end of the schedule (e.g., $l = 1$ means the last position) on *some* machine. The number of available machines is reflected by the property that any feasible assignment $\boldsymbol{\pi}$ contains at most m elements of the same value. In addition, any optimal solution is of a specific structure. That is, having an instance with n jobs and m machines, an optimal solution $\boldsymbol{\pi}^\star$ to the studied problem has exactly m elements with the value of 1, exactly m elements with the value of 2, and so on up to $\lfloor n/m \rfloor$. Finally, when n is not divisible by m, we have additional $n - \lfloor n/m \rfloor \cdot m$ positions, with the value of $\lfloor n/m \rfloor + 1$. Actually, since the assignment of a job to the specific machine is disregarded by $\boldsymbol{\pi}$, then whenever $m > 1$, a single $\boldsymbol{\pi}$ defines more solutions identical up to the permutations of machines with the identical objective values (see Example 1 bellow). Nevertheless, since the solutions are identical from the objective function point of view, we treat $\boldsymbol{\pi}$ as a single solution.

Considering the structure of solutions explained above, we have that the set of (potentially) optimal assignments $\boldsymbol{\pi}$ is given as

$$\boldsymbol{\Pi} = \left\{ \boldsymbol{\pi} \in \{1, \ldots, \lfloor n/m \rfloor + 1\}^n \;\middle|\; \begin{array}{l} c^{\boldsymbol{\pi}}(\lfloor n/m \rfloor + 1) = n - \lfloor n/m \rfloor \cdot m, \\ \forall l \in \{1, \ldots, \lfloor n/m \rfloor\} : \quad c^{\boldsymbol{\pi}}(l) = m \end{array} \right\}, \tag{1.1}$$

where $c^{\boldsymbol{\pi}}(l) = |\{j \in \mathcal{J} : \pi_j = l\}|$ is the number of jobs assigned to l-th position from the end. The positions of jobs are indexed in the reversed order, as it leads

to a simplified representation of the objective function. Subsequently, the objective function f of the problem can be written as

$$f \equiv f(\boldsymbol{\pi}, \tilde{\boldsymbol{p}}) = \boldsymbol{\pi}^\intercal \tilde{\boldsymbol{p}}. \tag{1.2}$$

When the probability distribution P of the processing times $\tilde{\boldsymbol{p}} \sim P$ is known exactly, then one can utilize Stochastic Programming (SP) solution, which minimizes the expectation of the objective function:

$$\text{SP-PTFT} \equiv \min_{\boldsymbol{\pi} \in \boldsymbol{\Pi}} \mathbb{E}_P\big[\boldsymbol{\pi}^\intercal \tilde{\boldsymbol{p}}\big] = \min_{\boldsymbol{\pi} \in \boldsymbol{\Pi}} \boldsymbol{\pi}^\intercal \mathbb{E}_P\big[\tilde{\boldsymbol{p}}\big]. \tag{1.3}$$

Example 1. Let us have a problem instance with $m = 2$ machines and $n = 5$ jobs with uncertain processing times $\tilde{\boldsymbol{p}} = (\tilde{p}_1, \ldots, \tilde{p}_5)$. Suppose that we have a feasible solution $\boldsymbol{\pi} = (1, 1, 2, 3, 2)$ to the problem (1.3). The solution $\boldsymbol{\pi}$ represents $2^3 = 8$ different job orders. One of these orders is illustrated in Figure 1 (a), where job 4 is scheduled as the first job on the first machine, followed by job 3 and job 1 on the same machine. The remaining jobs are allocated to the second machine, where job 5 is followed by job 2. Using the linearity of the expected value, the objective for order (a) can be rewritten as

$$\mathbb{E}_P\big[\boldsymbol{\pi}^\intercal \tilde{\boldsymbol{p}}\big] = 3 \cdot \mathbb{E}_P\big[\tilde{p}_4\big] + 2 \cdot \mathbb{E}_P\big[\tilde{p}_3\big] + 1 \cdot \mathbb{E}_P\big[\tilde{p}_1\big] + 2 \cdot \mathbb{E}_P\big[\tilde{p}_5\big] + 1 \cdot \mathbb{E}_P\big[\tilde{p}_2\big].$$

Note that the order in Figure 1 (b) leads to the same result, as the multipliers of $\tilde{\boldsymbol{p}}$ are identical.

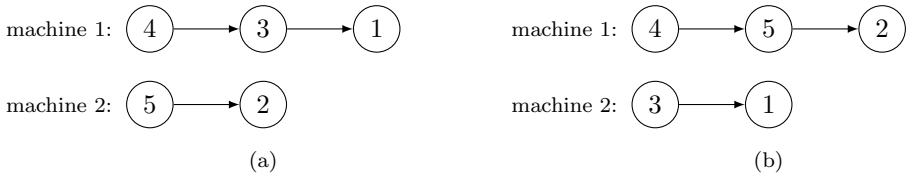

Figure 1: Two (out of eight) different job orders on two machines represented by $\boldsymbol{\pi} = (1, 1, 2, 3, 2)$.

1.2. Distributionally robust solution

Although tt can be seen that the SP solution is optimal in the sense of expected (or long term) performance, it does not hedge against variances in solution quality, thus, it may not be suitable for the risk-averse decision maker. Moreover, the probability distribution P is often not precisely known, therefore, it is advantageous to protect ourselves from sudden disturbances in solution quality caused by the changes in the distribution parameters. This is the reason why it is often useful to assume a broader concept, i.e., a set of probability distributions called an ambiguity set $\mathcal{D}$. Such a set can be in practice build with historical data, model assumptions, or problem constraints, using specific rules. Therefore, the problem is seen as *distributionally robust optimization* (DRO), which aims to find a solution that yields the best expected value of the objective function f for the worst-case distribution in $\mathcal{D}$. Thus, the probability distribution acts as a decision variable and the goal is to find solution $\boldsymbol{\pi}$ that minimizes its expected objective value with respect to the worst-case realization of the probability distribution.

Using the introduced notation, the distributionally robust problem (denoted as *DR-PTFT*) becomes

$$\text{DR-PTFT} \equiv \min_{\boldsymbol{\pi} \in \boldsymbol{\Pi}} \max_{P \in \mathcal{D}} \mathbb{E}_P\big[\boldsymbol{\pi}^\intercal \tilde{\boldsymbol{p}}\big]. \tag{1.4}$$

There are numerous ways to define an ambiguity set. The ambiguity set used in paper [8], as well as in this paper, constrains the first two moments of the distribution, and it is defined as

$$\mathcal{D} = \left\{ P \;\middle|\; \begin{array}{l} \mathbb{P}_P\left[\tilde{\boldsymbol{p}} \in \mathbb{R}^n_+\right] = 1 \\ \left(\mathbb{E}_P[\tilde{\boldsymbol{p}}] - \hat{\boldsymbol{\mu}}\right)^\intercal \hat{\boldsymbol{\Sigma}}^{-1} \left(\mathbb{E}_P[\tilde{\boldsymbol{p}}] - \hat{\boldsymbol{\mu}}\right) \leq \gamma_1^2 \\ \mathbb{E}_P\left[(\tilde{\boldsymbol{p}} - \hat{\boldsymbol{\mu}})(\tilde{\boldsymbol{p}} - \hat{\boldsymbol{\mu}})^\intercal\right] \preceq \gamma_2^2 \hat{\boldsymbol{\Sigma}} \end{array} \right\}, \tag{1.5}$$

where $\hat{\boldsymbol{\mu}} \geq \mathbf{0}$ is an estimate of the mean vector, and $\hat{\boldsymbol{\Sigma}} \succeq \mathbf{0}$ is an estimate of the covariance matrix. The parameters γ_1 and γ_2 ($\gamma_1 \geq 0$, $\gamma_2 \geq 1$) define confidence in the estimates. The set can be interpreted such that the mean vector $\mathbb{E}_P[\tilde{\boldsymbol{p}}]$ is restricted in an ellipsoid of size γ_1; centered at its estimate $\hat{\boldsymbol{\mu}}$. The covariance of the distribution P, in turn, lies in a positive semidefinite cone defined by $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$. The reason why it is convenient to model the ambiguity set with the two first central moments follows from the difficulty of estimating higher-order moments in case of a lack of data. Indeed, the statistical estimators of covariance matrices (e.g., sample covariance) have higher variance than the estimators of the expected value (e.g., sample mean). Thus, the higher moment one wants to estimate, the more data for its reliable estimation is needed. Furthermore, in the case of multivariate distributions (joint distributions), there are several different methods how to measure and interpret skewness (i.e., the standardized third moment), even for distribution from skewed-normal family [2]. Thus, different measures might be suitable for different applications, which makes the estimation of higher-order moments rather complicated for multivariate distributions.

Using the above notation, the problem studied in this paper can be denoted with three-field notation as $\mathrm{P}|\mathbb{P}[\tilde{\boldsymbol{p}}] \in \mathcal{P}^{DY}|\sum C_j$, where $\mathcal{P}^{DY}$ stands for Delange and Ye's ambiguity set [8, 27]. To keep the notation short, we refer to the studied problem as to DR-PTFT. Note that one of the advantageous properties of ambiguity set $\mathcal{D}$ is that when $\gamma_1 = 0$, then the worst-case expectation problem of the DRO formulation reduces exactly to the ordinary expectation of the objective function which matches SP formulation (1.3). Thus, the DRO formulation (1.4) contains an SP solution (1.3) as a special case.

1.3. Contributions and paper organization

In this paper, we revisit DR-PTFT problem from the perspective of the design and analysis of the algorithms. We demonstrate that its solution for large problem instances is computationally intractable in practice, especially with dependent jobs. Thus, we aim to design algorithms with a (pseudo) polynomial complexity, and simultaneously, to provide solutions with the same or almost the same desired properties as the optimal solutions to formulation (1.4). We achieve this aim by expressing the variance of a solution with a robust term, and by considering its different forms. Namely, the main contributions of this paper are:

1. we reformulate DR-PTFT as a minimization of a linear function plus a robust term in the sense of ℓ_2 norm (see Section 3.2);

2. we investigate the effect of the form of the robust term on the computational complexity and, as a special case of our theorem, we improve the best-known upper bound of [8] on the complexity for the problem with independent jobs (see Section 3.3);

3. we extend our methods to the case when processing times of jobs are dependent and we show that the source of the hardness of the problem arises from the

presence of large negative correlations, not from the simple fact that jobs are dependent (see Section 3.4 and Section 3.5);

4. we show that the robust term in the sense of ℓ_1 norm allows to solve the problem in polynomial time, and we provide the explicit definition of the corresponding ambiguity set (see Section 3.5);

5. we relate the proposed methods to multi-objective optimization setting in terms of expected quality and the solution variance; we significantly improve a method for uniform sampling of the solution Pareto set (see Section 4);

6. the experimental results show that our polynomial approximations have nearly identical performance to the formerly known second-order cone integer programming formulation from [8] while being much faster (see Section 5).

The rest of the paper is organized as follows. In Section 2, we survey the related work. In Section 3, we study the computational complexity of the problem in terms of ℓ_p norm with independent jobs. Then, we focus on a particular case of ℓ_1 norm, for which we propose a polynomial-time algorithm with the extension for the case of dependent jobs. In Section 4, we point out the relation between the form of the objective function of the problem and the multi-objective optimization in terms of solution quality and its robustness and discuss some practical concerns for solving the problem. Finally, in Section 5, we perform numerical experiments with our algorithms, and we provide a comparison to the state-of-the-art methods. Section 6 concludes the work.

Notation. Generally we use calligraphic letters ($\mathcal{A}$) to denote sets, for vectors and matrices we use bold ($\boldsymbol{a}$, $\boldsymbol{A}$), tilde ($\tilde{a}$) for random variables, and for the estimates the hat ($\hat{a}$). By $\mathbf{0}$ and $\mathbf{1}$ we denote, respectively, vectors of zeros and ones of appropriate sizes. The diagonal matrix with vector $\boldsymbol{\lambda}$ on its diagonal is denoted as $\mathrm{diag}(\boldsymbol{\lambda})$. The set of all probability distributions on $\mathbb{R}^n$ is written as $\mathcal{P}_0(\mathbb{R}^n)$. Element-wise comparison of vectors $\boldsymbol{a}$ and $\boldsymbol{b}$ is defined as $\boldsymbol{a} \circ \boldsymbol{b} \iff \forall i : \; a_i \circ b_i$, where $\circ \in \{>, \geq, <, \leq, =\}$. We define ℓ_p norm of a vector $\boldsymbol{x} \in \mathbb{R}^n$ as $\|\boldsymbol{x}\|_p = \left(\sum_{i=1}^n |x_i|^p\right)^{1/p}$. Furthermore, we denote the set of all symmetric real positive semidefinite matrices of size $n \times n$ as $\mathbb{S}_+^n$, the set of non-negative reals as $\mathbb{R}_+$, the set of non-negative integers as $\mathbb{Z}_+$, and positive integers as $\mathbb{N}$.

2. Related work

Distributionally robust optimization (DRO) was introduced by Scarf [28] back in 1958. The aim of DRO is to minimize the worst-case expectation with respect to the uncertainty of the underlying distribution of the parameters, i.e., the so-called ambiguity set. The new wave of interest in DRO was sparked namely by recent advancements of mathematical programming solvers and tractable formulations of ambiguity sets [6]. A DRO problem is typically reformulated to a deterministic mathematical programming problem, whose complexity depends on the used formulation of the ambiguity set. Often, such reformulation is more computationally attractive than stochastic and robust optimization counterparts.

2.1. DRO in the scheduling literature

The majority of the existing works dealing with scheduling problems and DRO have applied ambiguity sets defined by estimates of the first two moments. Wang *et*

al. [36] solve the assignment of surgery blocks to operating rooms, which leads to the objective function containing a non-linear term $\sum_i \max\{0, \boldsymbol{d}_i^\intercal \boldsymbol{x} - T\}$ ($\boldsymbol{d}_i$ is a random vector of surgery durations, $\boldsymbol{x}$ are decision variables, and constant T is the regular operating room opening hours). Processing times of surgeries $\boldsymbol{d}_i$ are subject to a probability distribution contained in the ambiguity set defining bounds on mean values and mean absolute deviations. The proposed reformulation of the DRO problem formulation leads to a mixed-integer linear program (MILP) of exponential size in the number of operating rooms. The approach is able to solve problems with about 15 surgery blocks within an hour.

A DRO variant of a single machine total tardiness problem with uncertain processing times was addressed in [26]. The authors used an ambiguity set enforcing equality of the first two moments. The exact reformulation has high complexity, with the inner problem being an exponential-sized SDP (semidefinite programming) problem. Therefore, they solved a surrogate SOCP (second-order cone programming) problem instead by a custom branch-and-bound algorithm. They have been able to solve instances with 30 jobs within 40 seconds. Shang *et al.* [29] use the generalized moment functions with a piece-wise linear form, given by so-called truncation points, to define the ambiguity set. They can be used to constrain the first-order deviation projected along the selected direction. The authors also show a problem reformulation leading to a MILP. Furthermore, they propose a data-driven procedure based on principal component analysis to construct an ambiguity set from the historical data, and they apply the framework to a process scheduling problem.

A problem with bimodal distributions was studied in [32] in the context of outpatient colonoscopy scheduling. Colonoscopy duration is uncertain, and it is conditioned by the bowel preparation quality, which is uncertain as well. Moreover, uncertainty in the time when the patient will show up for the procedure is considered as well. The goal is to sequence patients such that the worst-case expectation of the weighted sum of patient and provider waiting with the overtimes is minimized. The authors use an ambiguity set that enforces support (i.e., lower and upper bounds) and the mean value for all uncertain parameters. The problem is translated into a MILP and solved by CPLEX solver.

In this paper, we build on the work of [8]. They have proposed a distributionally robust variant of the parallel identical machine scheduling problem with the minimization of the worst-case expected total flow time. The processing times of jobs are subject to uncertainty belonging to the ambiguity set constraining the first two moments. The problem is reduced to an integer SOCP, and the case with independent jobs is solved by an exact algorithm that explores all solutions satisfying necessary optimality conditions. The proposed approach for independent jobs was able to solve instances with 100 jobs and 5 machines within several seconds whereas the integer SOCP formulation does not scale well. In our work, we address this problem from the perspective of surrogate problems and their complexity. That is, we classify related problems with respect to their complexity and we show when it is possible to obtain identical quality and robustness of solutions at a much lower computational cost. What is more, we extend the proposed methodology for the case of dependent jobs which displays excellent scaling capabilities.

2.2. Ambiguity set expressivity and robustness evaluation

A known shortcoming of the moment-based ambiguity sets is that the worst-case distribution might have an unrealistic form [37]. For example, it is known that under mild assumptions on the objective function, its worst-case expectation is attained at the distribution having support in at most $m + 1$ points, if m moments

are constrained [31, 4]. If such distribution leads to overly conservative solutions for the target application, then it is better to use, e.g., likelihood or phi-divergence ambiguity sets [3].

The above points raise a question of whether it is appropriate to solve problems with certain ambiguity sets optimally when neither the protection against the (unrealistic) worst-case distribution is not required nor is in the interest of the decision maker. Indeed, the majority of DRO applications in scheduling do not evaluate solutions with respect to the worst-case distribution which was chosen by their DRO algorithm. Instead, the authors assume various selected distributions or they choose different evaluation protocols that are suited to the target application [36, 29, 10]. Nevertheless, the way a DRO algorithm is tested may result in a situation where, sometimes, even a heuristic solution can achieve a better performance than the exact approach under some sensible evaluation protocol, e.g., as in [36]. Therefore, this paper tries to answer a question, that is, given an evaluation protocol of out-of-sample performance, is it necessary to solve the original problem optimally, or can a comparable performance be achieved by solving a related problem, perhaps at a much-reduced computation cost?

2.3. Total flow time scheduling and other problems

From the perspective of the scheduling problem solved in this work, several works on total flow time scheduling are closely related. For example, in [19], the authors study a deterministic parallel identical machines scheduling problem with weighted completion time. With their enhanced arc-flow MILP formulation, they have been able to solve instances having up to 400 jobs, whereas former approaches were limited to around 100 jobs. A robust approach for parallel machines total flow time problem is studied in [1]. The authors treat the problem with normally distributed processing times as a β-robust optimization problem, where the objective is to maximize the probability that the total flow time does not exceed the given level. They developed a branch-and-bound algorithm that was able to solve instances of up to 45 jobs and 5 machines. Another total flow time scheduling problem with sequence-dependent setups is addressed by [21]. The uncertain processing times and setups are represented by interval data. The goal is to minimize the worst-case absolute deviation of the total flow time from the optimal scenario. They have formulated the problem as a resource-constrained shortest path and devised a simulated annealing algorithm to solve it. They were able to solve instances with 200 jobs in about 20 seconds.

From the perspective of the used approaches, our methods share similarities with linear optimization problems containing absolute values of variables. Problems such as *absolute value equations* (AVE) [23] or *linear complementarity problem* (LCP) [11] are known to be $\mathcal{NP}$-hard even over real variables. We further consider problems related to ℓ_p norm minimization [15, 38], which are typically studied in the context of robust estimation and fitting. However, the current results are not directly applicable to our problem as we optimize over a set of constraints having a form of a totally unimodular matrix rather than an unconstrained case.

3. Solving methods for ℓ_p norm formulations

In Section 3.1, we outline the second-order cone program (SOCP) formulation of problem (1.4) from [8]. In Section 3.2, we express the objective function as a sum of a linear term and a robust term in the sense of some ℓ_p norm. Section 3.3 investigates the properties and complexity of the formulation with respect to the used norm for independent jobs. Finally, Section 3.4 focuses on the reformulation with dependent

jobs, while Section 3.5 on ℓ_1 norm specifically. Our analysis shows that the problem with dependent jobs is hard only when the large negative correlations are present. Furthermore, we give a polynomial algorithm for a tractable subclass of the problem, and we investigate its robustness, given by the corresponding ambiguity set.

3.1. Deterministic reformulation of the stochastic problem

In [8], it was shown that when parameters $\gamma_1 \geq 0$ and $\gamma_2 \geq 1$, defining the ambiguity set (1.5), satisfy $\gamma_2 \geq \gamma_1$, then the stochastic problem in the form of (1.4) is equivalent to the following deterministic integer second-order cone program

$$\min_{\boldsymbol{\pi} \in \boldsymbol{\Pi}} \boldsymbol{\pi}^\intercal \hat{\boldsymbol{\mu}} + \gamma_1 \cdot \sqrt{\boldsymbol{\pi}^\intercal \hat{\boldsymbol{\Sigma}} \boldsymbol{\pi}}. \tag{3.1}$$

Interestingly, the resulting formulation does not depend on the particular value of γ_2, as long as $\gamma_2 \geq \gamma_1$. For more details, we refer the reader to [8]. Furthermore, note that when $\gamma_1 = 0$, then (3.1) matches SP solution (1.3).

In [8], authors have dealt with the special case of the problem with independent random variables, i.e., $\hat{\boldsymbol{\Sigma}} = \operatorname{diag}(\hat{\sigma}_1^2, \hat{\sigma}_2^2, \ldots, \hat{\sigma}_n^2)$ where $\hat{\sigma}_j$, $j \in \mathcal{J}$ is a standard deviation of $\tilde{p}_j$. We note that when $\hat{\boldsymbol{\Sigma}} = \mathbf{0}$ (i.e., when the processing times are deterministic), then the formulation (3.1) reduces to the classical, deterministic parallel identical machines total flow time problem ($P||\sum C_j$), with processing times given by $\hat{\boldsymbol{\mu}}$. The deterministic problem is solvable in $\mathcal{O}(n \log n)$ time by sorting the jobs according to $\hat{\boldsymbol{\mu}}$ values in non-increasing order. The optimal $\boldsymbol{\pi}$ is obtained by setting $\pi_j = 1$ to the first m sorted jobs, $\pi_j = 2$ to the next m jobs until all the elements in $\boldsymbol{\pi}$ are assigned (see, e.g., [7, p. 133–134] for a similar algorithm applicable to a more general $Q||\sum C_j$ problem).

Finally, note that since $\hat{\boldsymbol{\mu}}$ (i.e., sample mean) is unbiased estimator of $\mathbb{E}[\tilde{\boldsymbol{p}}]$, we can build a direct connection between the Stochastic Programming formulation SP-PTFT and the considered DRO formulation. That is, the solution of (3.1) with $\gamma_1 = 0$ is equivalent to SP-PTFT formulation.

3.2. Robustness as a norm of solution variance

In this section, we express formulation (3.1) using a vector norm of the solution variance, which provides new insights to the problem. Let us assume that the estimate of covariance matrix $\hat{\boldsymbol{\Sigma}}$ is a positive semidefinite (PSD) matrix. Indeed, this is without a loss of generality as the covariance matrix of any distribution is a PSD matrix (if it exists). From the practical standpoint, the covariance matrix is typically estimated from data using the sample covariance, which provably always results in a PSD matrix. Thus, $\hat{\boldsymbol{\Sigma}}$ admits factorization into $\hat{\boldsymbol{\Sigma}} = \boldsymbol{V}\boldsymbol{D}\boldsymbol{V}^{-1}$, where $\boldsymbol{V}$ is an orthogonal matrix and $\boldsymbol{D}$ is a diagonal matrix with eigenvalues $\lambda_j \geq 0$ of $\hat{\boldsymbol{\Sigma}}$. Let us define a square root of $\hat{\boldsymbol{\Sigma}}$ as

$$\hat{\boldsymbol{\Sigma}}^{1/2} \equiv \boldsymbol{V}\boldsymbol{D}^{1/2}\boldsymbol{V}^{-1}, \tag{3.2}$$

where $\boldsymbol{D}^{1/2}$ is a diagonal matrix computed as element-wise square root of $\boldsymbol{D}$.

Lemma 1. *Problem* (3.1) *can be equivalently expressed as*

$$\textit{DR-PTFT}(\ell_2) \equiv \min_{\boldsymbol{\pi} \in \boldsymbol{\Pi}} \boldsymbol{\pi}^\intercal \hat{\boldsymbol{\mu}} + \gamma_1 \cdot \left\| \hat{\boldsymbol{\Sigma}}^{1/2} \boldsymbol{\pi} \right\|_2, \tag{3.3}$$

where $\|\cdot\|_2$ *is* ℓ_2 *(euclidean) norm.*

Proof. Observe that $\|\boldsymbol{x}\|_2 = \sqrt{\boldsymbol{x}^\intercal \boldsymbol{x}}$ and $\hat{\boldsymbol{\Sigma}}^{1/2}$ is a PSD matrix and is symmetric. Then, since $\boldsymbol{V}^{-1} = \boldsymbol{V}^\intercal$, it follows that

$$\left\|\hat{\boldsymbol{\Sigma}}^{1/2}\boldsymbol{\pi}\right\|_2 = \sqrt{\left(\hat{\boldsymbol{\Sigma}}^{1/2}\boldsymbol{\pi}\right)^\intercal \hat{\boldsymbol{\Sigma}}^{1/2}\boldsymbol{\pi}} = \sqrt{\boldsymbol{\pi}^\intercal \boldsymbol{V}\boldsymbol{D}^{1/2}\boldsymbol{V}^{-1}\boldsymbol{V}\boldsymbol{D}^{1/2}\boldsymbol{V}^{-1}\boldsymbol{\pi}} = \sqrt{\boldsymbol{\pi}^\intercal \hat{\boldsymbol{\Sigma}}\boldsymbol{\pi}}. \tag{3.4}$$

By substituting (3.4) into (3.3), we obtain (3.1). □

While the reformulation (3.3) itself does not bring anything novel, it provides an interesting insight into the connections of (1.4) with the related problems. Namely, we see certain similarities with robust regression methods used in machine learning (ML). There, typically one does not have a precise knowledge of the underlying distribution of the data. Instead, one has access to a finite sample set that can be used to estimate the ambiguity. The training of a prediction model is treated as an optimization problem, minimizing a function defined as a mean error on input samples plus a complexity measure of the model, which typically refers to the number of degrees of freedom used in the learned model. The end goal is to find a model that achieves a small error on unseen data. The resulting model (by analogy, here—a solution to the scheduling problem) is chosen such that it does not overfit the training data (sampled processing times) by having some level of generalization to unseen data (robustness with respect to the processing times uncertainty). Viewing the researched problem from the perspective of ML analogy, $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}^{1/2}$ are derived from the training set, $\boldsymbol{\pi}$ represents the model, $\hat{\boldsymbol{\mu}}^\intercal\boldsymbol{\pi}$ is its error on input samples (performance), while an estimate of model complexity (variance) is expressed as $\|\hat{\boldsymbol{\Sigma}}^{1/2}\boldsymbol{\pi}\|_2$ and acts as a regularization term. The level of protection against overfitting is typically controlled by a weight term for the regularization term, in our case, corresponding to γ_1. There are several common methods in ML how to penalize the model complexity, e.g.,: *ridge regression*, *support vector machines (SVM)* (squared ℓ_2 norm), *lasso regression* (ℓ_1 norm), or *smoothing regularization* ($\|\boldsymbol{D}\boldsymbol{\pi}\|_p$ for some suitably chosen matrix $\boldsymbol{D}$) [25]. Frequently, an ℓ_p norm of the model parameters is used. Different choices of penalty terms lead to different models and training (optimization) algorithms [14]. Similar connections between DRO and regularization approaches were observed by other authors as well, see, e.g., [27].

Therefore, the above reformulation stimulates several interesting questions. Namely, we ask whether the ℓ_2 norm used in DR-PTFT(ℓ_2) formulation (3.3) is essential to preserving the quality of solutions, or maybe rather, can it be replaced with a different penalty (e.g., ℓ_1 norm)? What are then the performance guarantees of such a model and how does the change of regularization affects the complexity of the problem? In the following sections, we provide answers to these questions.

3.3. Complexity of ℓ_p formulation with independent jobs

In the paper [8], the computational complexity problem of (3.1) was not studied. Regarding the complexity of their algorithm for independent jobs (called NPSA) was shown that terminates within the finite number of iterations, but no specific time bound was given. In this section, we provide a complexity characterization for the problem formulation with independent jobs in sense of any ℓ_p norm:

$$\text{DR-PTFT}(\ell_p) \equiv \min_{\boldsymbol{\pi}\in\boldsymbol{\Pi}} \hat{\boldsymbol{\mu}}^\intercal\boldsymbol{\pi} + \gamma_1 \cdot \left\|\hat{\boldsymbol{\Sigma}}^{1/2}\boldsymbol{\pi}\right\|_p. \tag{3.5}$$

We show that the particular case $p = 2$ of the following proposition reduces to problem (3.1) with independent jobs which, in turn, establish a new complexity

result and provides a new algorithm for problem (3.1) with independent jobs studied in [8].

In this subsection, we study the case when the processing times of jobs are independent, i.e., $\hat{\boldsymbol{\Sigma}} = \operatorname{diag}(\hat{\sigma}_1^2, \hat{\sigma}_2^2, \ldots, \hat{\sigma}_n^2)$. Provided that the processing times are non-negative, without a loss of generality we assume that all elements of $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\Sigma}}$ are non-negative integers. Finally, the following proposition provides a characterization of the computational complexity and the solution algorithm for DR-PTFT(ℓ_p) with independent jobs.

Proposition 1. *For any fixed integer $p \geq 1$ and a diagonal covariance matrix $\hat{\boldsymbol{\Sigma}} = \operatorname{diag}(\hat{\sigma}_1^2, \ldots, \hat{\sigma}_n^2)$, problem DR-PTFT($\ell_p$) admits a pseudopolynomial algorithm in $\max_{j \in \mathcal{J}} \{\hat{\mu}_j \cdot \lambda, (\hat{\sigma}_j \cdot \lambda)^p\}$, where $\lambda = \min\left\{\hat{\boldsymbol{\mu}}^\intercal \mathbf{1}, \mathbf{1}^\intercal \hat{\boldsymbol{\Sigma}} \mathbf{1}\right\}$.*

Proof. We prove the statement by reducing DR-PTFT(ℓ_p) problem to a non-linear perfect matching problem in a bipartite graph [5] with a suitably defined non-linear objective function. Such problem is given as the maximization of d-dimensional convex function $q(z_1, \ldots, z_d) : \mathbb{R}_+^d \mapsto \mathbb{R}$ over a set of all perfect matchings $\boldsymbol{g} \in \{0,1\}^{n^2}$ in complete bipartite graph $K_{n,n}$. The arguments of the function q are given as mere linear combinations of the characteristic vector $\boldsymbol{g}$ of the matching and weights $z_i = \boldsymbol{w}_i^\intercal \boldsymbol{g}$, $\boldsymbol{w}_i \in \mathbb{Z}_+^{n^2}$ where $i \in \{1, \ldots, d\}$. In other words, each matching $\boldsymbol{g}$ is scored by d different non-negative integer weights, which are aggregated into a single convex scoring function to be maximized. When the number of arguments d of the function to be maximized is fixed to a constant, then such problem can be solved in a pseudopolynomial time in the maximal weight, as shown in [5]. The reduction given below preserves the pseudopolynomial time complexity with respect to the sum of means and variances, and, thus, the algorithm scheme of [5] is applicable to solve the problem.

Let us define complete bipartite graph $K_{n,n} = (\mathcal{J}, \mathcal{L}, \mathcal{E})$, where $\mathcal{J}$ is a set of all jobs and $\mathcal{L}$ is a multiset of all eligible positions (i.e., including their multiplicity, see the definition of $\boldsymbol{\Pi}$ in (1.1)) for any job given as

$$\mathcal{L} = \{ \underbrace{1, \ldots, 1}_{m \text{ elements}}, \underbrace{2, \ldots, 2}_{m \text{ elements}}, \ldots, \underbrace{\lfloor n/m \rfloor, \ldots, \lfloor n/m \rfloor}_{m \text{ elements}}, \underbrace{\lfloor n/m \rfloor + 1, \ldots, \lfloor n/m \rfloor + 1}_{(n - \lfloor n/m \rfloor \cdot m) \text{ elements}} \}.$$

The first part of the graph represents jobs, and the other part represents all possible job positions, including their multiplicity given by the number of machines. We associate each edge $(j, l) \in \mathcal{E}$, $j \in \mathcal{J}, l \in \mathcal{L}$ with two weights $(\hat{\mu}_j \cdot l, |\hat{\sigma}_j|^p \cdot l^p)$ and we collect all first and second weights into vectors $\boldsymbol{a}, \boldsymbol{b} \in \mathbb{Z}_+^{n^2}$.

Next, let us denote a perfect matching in $K_{n,n}$ as $M \subseteq \mathcal{E}$. Such matching M can be represented by the characteristic vector $\boldsymbol{g} \in \{0,1\}^{n^2}$ with i-th entry being one if and only if the corresponding edge is contained in M, and zero otherwise. Finally, let us define function $q : \mathbb{R}_+^2 \mapsto \mathbb{R}$ given as

$$q(z_1, z_2) = -z_1 - \gamma_1 \cdot {z_2}^{1/p}.$$

Note that, as $\gamma_1 \geq 0$, we have that q is convex on $\mathbb{R}_+^2$ for any $p \in \mathbb{N}$. Then, for any perfect matching $\boldsymbol{g}$, we have that

$$\begin{aligned} q(\boldsymbol{a}^\intercal \boldsymbol{g}, \boldsymbol{b}^\intercal \boldsymbol{g}) = -\boldsymbol{a}^\intercal \boldsymbol{g} - \gamma_1 \cdot (\boldsymbol{b}^\intercal \boldsymbol{g})^{1/p} &= -\boldsymbol{\pi}^\intercal \hat{\boldsymbol{\mu}} - \gamma_1 \cdot \left(\sum_{j=1}^{n} |\hat{\sigma}_j|^p \cdot \pi_j^p \right)^{1/p} \\ &= -\boldsymbol{\pi}^\intercal \hat{\boldsymbol{\mu}} - \gamma_1 \cdot \left\| \hat{\boldsymbol{\Sigma}}^{1/2} \boldsymbol{\pi} \right\|_p, \end{aligned} \tag{3.6}$$

where $\pi_j = l$ if and only if edge $(j, l) \in M$. The first equality follows from the definition of q while the second one follows from the definition of $\boldsymbol{a}$ and $\boldsymbol{b}$ and the fact that $\boldsymbol{g}$ is a perfect matching in a bipartite graph. Thus, for each $j \in \mathcal{J}$, exactly one edge to some $l \in \mathcal{L}$ is selected. Finally, any perfect matching $\boldsymbol{g}^\star$ maximizing $q(\boldsymbol{a}^\intercal \boldsymbol{g}^\star, \boldsymbol{b}^\intercal \boldsymbol{g}^\star)$ corresponds to minimizing (3.3) which solves DR-PTFT(ℓ_p) problem.

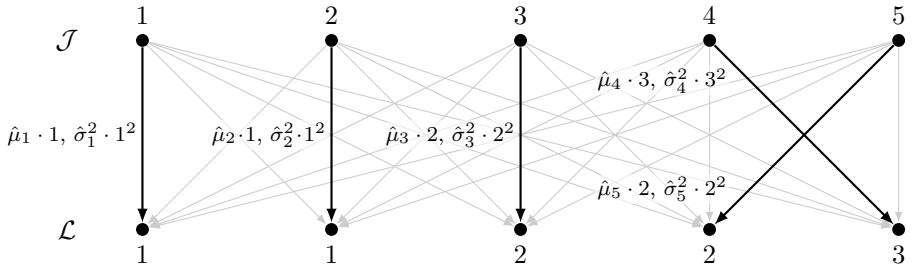

Figure 2: Graph $K_{n,n}$ for instance with $n = 5$, $m = 2$, and optimal solution $\boldsymbol{\pi}^\star = (1, 1, 2, 3, 2)$.

Example 2 (cont.)**.** We continue with the example introduced in Section 1.1. Let $\hat{\boldsymbol{\mu}} = (5, 3, 3, 1, 2)$ be a vector of estimated means and let $\hat{\boldsymbol{\Sigma}} = \operatorname{diag}(1, 2, 1, 4, 3)$ be a diagonal covariance matrix (i.e., the jobs processing times are independent). Finally, let $\gamma_1 = 1$ and $p = 2$. The graph $K_{n,n}$ for this problem instance is depicted in Figure 2. The edges in bold correspond to the optimal solution $\boldsymbol{\pi}^\star$.

The remaining question is, how to find such $\boldsymbol{g}^\star$ in the required time. We see that our function q and weights $\boldsymbol{a}$ and $\boldsymbol{b}$ satisfy assumptions of Theorem 1.2 of [5] for the case with $d = 2$. Their algorithm runs in a polynomial time in n and $\max_i\{\boldsymbol{a}_i, \boldsymbol{b}_i\}$ provided that q is polynomially computable. In our setting, function q can be evaluated in a polynomial time in n and the constants involved can be upper bounded as $\max_i\{\boldsymbol{a}_i, \boldsymbol{b}_i\} \leq \max_j\{\hat{\mu}_j \cdot n, (\hat{\sigma}_j \cdot n)^p\}$ with $n \leq \min\{\hat{\boldsymbol{\mu}}^\intercal \mathbf{1}, \mathbf{1}^\intercal \hat{\boldsymbol{\Sigma}} \mathbf{1}\}$ since the parameters are non-negative integers. Thus, applying the algorithm described in [5] with the above-mentioned setting concludes the proof. □

Remark 1. *We note that when the robust term is p-th power of an ℓ_p norm (e.g., ℓ_2 norm squared), then the problem* (3.5) *with independent jobs becomes an ordinary min-cost perfect bipartite matching problem with a linear objective function. It can be seen from the equation* (3.6)*, when the term $\boldsymbol{b}^\intercal \boldsymbol{g}$ is raised to the p-th power, then the function q becomes separable. Therefore, only a single coefficient $c_{j,l} = \hat{\mu}_j \cdot l + \gamma_1 \cdot |\hat{\sigma}_j|^p \cdot l^p$ for an edge (j, l) suffices to resemble the problem with p-th power of an ℓ_p norm. Such problem can be solved as the ordinary min-cost perfect bipartite matching problem in polynomial time by, e.g.,* HUNGARIAN *algorithm in $\mathcal{O}(n^3)$ [17].*

The difficulty of extending the above proposition to the case of dependent jobs lies in the necessity of having the number of arguments of function q fixed to some constant d. The reason is that an underlying step of the algorithm from [5], which solves the non-linear bipartite matching problem constructs a d-dimensional integer lattice to examine, thus having an exponential complexity in d. When $\hat{\boldsymbol{\Sigma}}$ is a full covariance matrix, then we would need to have $d = 1 + n$ to exploit the same approach as described above. A possible way to go around it would be to compress the information contained in $\hat{\boldsymbol{\Sigma}}$ to a smaller matrix while preserving norms of vectors transformed by the corresponding linear mappings, which is the idea we will explore in the following section.

3.4. Approximate solution for ℓ_p norm with dependent jobs

The purpose of this section is to analyze to which extent the ideas developed in the previous section can be applied to the problem with dependent jobs. In the context of ℓ_1 norm, we show that the difficulty of the problem with dependent jobs lies in the presence of large negative correlations between jobs, not just in the plain fact that jobs are dependent.

To overcome the exponential grow of complexity in n, the trick is to compress the information about the norm of $\hat{\boldsymbol{\Sigma}}^{1/2}\boldsymbol{\pi}$ vector using a different vector $\hat{\boldsymbol{\Sigma}}_k^{1/2}\boldsymbol{\pi}$ of a fixed length k independent from n. However, the same algorithm as in the case with independent jobs cannot be applied. The difficulty is that for dependent jobs, the objective function used in non-linear bipartite perfect matching now loses its convexity. Thus, we will employ a weaker version of the algorithm for non-linear bipartite perfect matching which maximizes an arbitrary function $q : \mathbb{R}^{k+1} \mapsto \mathbb{R}$ over a set of perfect matchings in the complete bipartite graph. In [5], such algorithm is given which runs also in a pseudopolynomial time with the caveat that it is a randomized algorithm — i.e., an algorithm with access to the random bit generator which for any input returns an optimal solution with the probability of at least $1/2$.

The rationale behind the approach with a compressed covariance matrix is that performing computations over a matrix that is similar in some sense to the original one should yield similar results as performing the computation over the original matrix, but with a significantly reduced computational cost. Indeed, this scheme is frequently exploited in numerical linear algebra, e.g., to approximate solutions to problems such as multiplication of large matrices, matrix decompositions, approximate regression problems, and finds many other applications [34].

The general idea of the reduction is similar to the one used in Proposition 1 except for some minor differences, which we describe below. Then, we formulate the task of finding an approximation (compression) of the original matrix and present some solutions to this problem.

The underlying bipartite graph has the same structure as in Proposition 1. We associate each edge (j, l) with $k + 1$ values:

$$(\hat{\mu}_j \cdot l, s_{1,j} \cdot l + v, \ldots, s_{k,j} \cdot l + v),$$

where $s_{i,j}$ is (i, j)-th element of $\hat{\boldsymbol{\Sigma}}_k^{1/2}$ matrix and $v = n \cdot \max_{i,j} |s_{i,j}|$. We collect all $k + 1$ weights along all edges into vectors $\boldsymbol{w}_0, \ldots, \boldsymbol{w}_k \in \mathbb{R}^{n^2}$. Next, we denote the characteristic vector of matching $M \subseteq \mathcal{E}$ in $K_{n,n} = (\mathcal{J}, \mathcal{L}, \mathcal{E})$ as $\boldsymbol{g} \in \{0, 1\}^{n^2}$ with i-th entry being one if and only if the corresponding edge is contained in matching M. Next, we define function $q : \mathbb{R}^{k+1} \mapsto \mathbb{R}$,

$$q(z_0, z_1, \ldots, z_k) = -z_0 - \gamma_1 \cdot \|(z_1 - nv, \ldots, z_k - nv)\|_p \tag{3.7}$$

to be maximized over a set of all perfect matchings $\boldsymbol{g}$ with $z_i = \boldsymbol{w}_i^\intercal \boldsymbol{g}$.

In contrast to the case when $\hat{\boldsymbol{\Sigma}}^{1/2}$ was a diagonal matrix, some entries of $\hat{\boldsymbol{\Sigma}}_k^{1/2}$ might be negative, but the underlying algorithm for non-linear bipartite matching [5] requires weights which are non-negative integers. This is why we add a positive constant v to the last k values of each edge and subtract them back in (3.7). Furthermore, if some $s_{i,j}$ is not an integer, then we need to multiply all weights by a sufficiently large constant. Then, we can use randomized version of the algorithm for non-linear bipartite perfect matching, which performs maximization of an arbitrary function $q(\boldsymbol{w}_0^\intercal \boldsymbol{g}, \ldots, \boldsymbol{w}_k^\intercal \boldsymbol{g})$ given by a polynomial-time comparison oracle. Such a method follows from Theorem 1.3 of [5] with $d = k + 1$ running in a pseudopolynomial time, hence, avoiding an exponential complexity in n.

The question that remains is how to find a suitable approximation of $\hat{\mathbf{\Sigma}}^{1/2}$. Given parameter $k \in \mathbb{N}$, the goal is to find matrix $\hat{\mathbf{\Sigma}}_k^{1/2} \in \mathbb{R}^{k \times n}$ which does not yield to a large error for vectors $\boldsymbol{\pi} \in \mathbf{\Pi}$ in the sense of some ℓ_p norm. That is, one wishes to find

$$\min_{\hat{\mathbf{\Sigma}}_k^{1/2} \in \mathbb{R}^{k \times n}} \max_{\boldsymbol{\pi} \in \mathbf{\Pi}} \left| \|\hat{\mathbf{\Sigma}}^{1/2}\boldsymbol{\pi}\|_p - \|\hat{\mathbf{\Sigma}}_k^{1/2}\boldsymbol{\pi}\|_p \right|. \tag{3.8}$$

We call $\hat{\mathbf{\Sigma}}_k^{1/2} \in \mathbb{R}^{k \times n}$ matrix as *rank-k* approximation of $\hat{\mathbf{\Sigma}}^{1/2}$ and its *distortion* is defined as the maximum absolute difference of the norms of the two vectors over $\mathbf{\Pi}$ in the sense of ℓ_p.

An obvious question to ask is how to look for good approximations of $\hat{\mathbf{\Sigma}}^{1/2}$ with small ranks and how large distortions are incurred. The answer depends on the used norm. This problem is, in fact, very closely related to the *subspace embedding problem* [35], which has many applications, namely in numerical linear algebra. Since the solution of (3.8) in its generality goes well beyond the scope of this paper, we rather briefly describe some particular results related to our application. We provide below some examples of good approximations for some matrices under ℓ_1 norm. The following lemma addresses the case when jobs are positively correlated.

Lemma 2. *For any $\hat{\mathbf{\Sigma}}^{1/2} \in \mathbb{R}_+^{n \times n}$, there exists a rank-1 approximation $\hat{\mathbf{\Sigma}}_1^{1/2}$ with zero distortion in sense of ℓ_1 norm.*

Proof. Set $\hat{\mathbf{\Sigma}}_1^{1/2} = \mathbf{1}^\intercal \hat{\mathbf{\Sigma}}^{1/2}$, i.e., a matrix with column sums. Then, for $k = 1$, we have

$$\|\hat{\mathbf{\Sigma}}_k^{1/2}\boldsymbol{\pi}\|_1 = \|\mathbf{1}^\intercal \hat{\mathbf{\Sigma}}^{1/2}\boldsymbol{\pi}\|_1 = \left| \sum_{i=1}^{n} \sum_{j=1}^{n} \hat{\mathbf{\Sigma}}_{ij}^{1/2} \pi_j \right| = \sum_{i=1}^{n} \left| \sum_{j=1}^{n} \hat{\mathbf{\Sigma}}_{ij}^{1/2} \pi_j \right| = \|\hat{\mathbf{\Sigma}}^{1/2}\boldsymbol{\pi}\|_1,$$

where the third equality follows from the fact that $\hat{\mathbf{\Sigma}}^{1/2} \in \mathbb{R}_+^{n \times n}$. □

The above lemma also suggests how to obtain good approximations for covariance matrices with a small number of negative entries:

Corollary. *For any $\hat{\mathbf{\Sigma}}^{1/2}$ with at most k rows with a negative entry, there exists rank-$(k+1)$ approximation $\hat{\mathbf{\Sigma}}_{k+1}^{1/2}$ with zero distortion in sense of ℓ_1 norm.*

The construction is straightforward — keep all k rows with a negative element and, for the rest, apply Lemma 2, yielding a rank-$(k+1)$ approximation. The above approximations suggest that the complexity of the problem with correlated jobs in the sense of ℓ_1 norm is closely connected to the presence of negative correlations in the covariance matrix. In fact, based on the above construction it can shown that the distortion for a rank-1 approximation is proportional to n, $\mathsf{nne}(\hat{\mathbf{\Sigma}}^{1/2})$ and $\max_j \hat{\mathbf{\Sigma}}_{jj}^{1/2}$, where $\mathsf{nne}(\hat{\mathbf{\Sigma}}^{1/2})$ is the number of negative elements of $\hat{\mathbf{\Sigma}}^{1/2}$. Obviously, one can again trade-off the rank of the approximation for its precision by keeping some of the rows intact, yielding distortion dependant on $\mathsf{nne}(\cdot)$ of the remaining matrix. For other related results on ℓ_1 subspace embedding, we refer the reader to, e.g., [16, 9, 34].

In the following section, we turn our attention to the formulation utilizing ℓ_1 norm from the perspective of robustness and its computational complexity.

3.5. ℓ_1 norm formulation for dependent jobs

In this section, we will focus specifically on the problem with ℓ_1 norm and dependent jobs. As we show below, this particular case leads to favorable computational complexity as well as both theoretical guarantees of the robustness and its empirical performance. These favorable properties make this case the most practical one.

Let us consider the following problem

$$\text{DR-PTFT}(\ell_1) \equiv \min_{\boldsymbol{\pi}\in\boldsymbol{\Pi}} \hat{\boldsymbol{\mu}}^\intercal \boldsymbol{\pi} + \gamma_1 \cdot \left\|\hat{\boldsymbol{\Sigma}}^{1/2}\boldsymbol{\pi}\right\|_1, \tag{3.9}$$

where the robust term is expressed in the sense of ℓ_1 norm. As it will be shown below, the benefit of such formulation is that problem (3.9) can be solved in strongly polynomial time when $\hat{\boldsymbol{\Sigma}}$ fulfills so-called copositivy condition, which is related to the relative magnitude of negative elements in $\hat{\boldsymbol{\Sigma}}^{1/2}$ matrix. At the same time, as it is shown later in Section 5.3, the quality of solutions to problem (3.9) is comparable to the solutions of a more complex ℓ_2 formulation.

Speaking about general PSD covariance matrices, the complexity of problem (3.9) arises from the presence of the absolute value inside ℓ_1 norm. It is known that equations with absolute values are hard to solve even over the domain of real numbers [24], suggesting that solving (3.9) in its generality might be hard as well. Therefore, we will focus on the cases where each element of the vector $\hat{\boldsymbol{\Sigma}}^{1/2}\boldsymbol{\pi}$ is non-negative, which is more general than simply requiring $\hat{\boldsymbol{\Sigma}}^{1/2} \in \mathbb{R}_+^{n\times n}$. We show that such cases of the problem can be solved in polynomial time. For that, we introduce a subclass of matrices which acts as a generalization of strictly positive covariances:

Definition (Copositivity with respect to $\boldsymbol{\Pi}$)**.** *Let us define a set of matrices*

$$\mathcal{C}_+^{\boldsymbol{\Pi}} = \left\{ \boldsymbol{A} \in \mathbb{S}_+^n \,\middle|\, \forall \boldsymbol{\pi} \in \boldsymbol{\Pi} \subset \mathbb{Z}_+^n : \boldsymbol{A}\boldsymbol{\pi} \geq \mathbf{0} \right\},$$

which is the set of PSD matrices that maps $\boldsymbol{\Pi}$ into $\mathbb{R}_+^n$.

Intuitively, the set of matrices $\mathcal{C}_+^{\boldsymbol{\Pi}}$ relates to the notion of diagonally-dominant matrices. Obviously, it follows that any covariance matrix of independent jobs (or strictly positively correlated) is contained in $\mathcal{C}_+^{\boldsymbol{\Pi}}$. Next, when $\hat{\boldsymbol{\Sigma}}^{1/2}$ matrix has diagonal elements that are about a factor $\mathcal{O}((n/m)^2)$ larger than the absolute value of the largest negative off-diagonal element, then it is likely to be contained in $\mathcal{C}_+^{\boldsymbol{\Pi}}$. Such covariance matrices appear, e.g., in distributions of so-called *weakly correlated random variables* [22]. As an example, we list some particular matrices below.

Example 3. Consider the following example covariance matrices:

$$\boldsymbol{A}_1 = \begin{bmatrix} 3 & 1 & 0 & 2 \\ 1 & 3 & 0 & 1 \\ 0 & 0 & 2 & 1 \\ 2 & 1 & 1 & 4 \end{bmatrix}, \quad \boldsymbol{A}_2 = \begin{bmatrix} 3 & -1 & 0 & 2 \\ -1 & 3 & 1 & 1 \\ 0 & 1 & 2 & -1 \\ 2 & 1 & -1 & 4 \end{bmatrix}, \quad \boldsymbol{A}_3 = \begin{bmatrix} 4 & -2 & 0 & 2 \\ -2 & 3 & 0 & 1 \\ 0 & 0 & 2 & 1 \\ 2 & 1 & 1 & 4 \end{bmatrix}.$$

All matrices are PSD. Next, it can be verified, e.g., by enumeration of all $\boldsymbol{\pi} \in \boldsymbol{\Pi}$, that $\boldsymbol{A}_1, \boldsymbol{A}_2 \in \mathcal{C}_+^{\boldsymbol{\Pi}}$ but $\boldsymbol{A}_3 \notin \mathcal{C}_+^{\boldsymbol{\Pi}}$ for $\boldsymbol{\Pi}$ corresponding to set of assignments for a single machine. As it was discussed in Section 3.4, Lemma 2 would suggest approximating $\boldsymbol{A}_1$ with a rank-1 matrix, $\boldsymbol{A}_2$ with a rank-4, and $\boldsymbol{A}_3$ with a rank-3 matrix. Thus, the notions of rank-k approximation and $\mathcal{C}_+^{\boldsymbol{\Pi}}$ are generally incomparable. Finally, note that when $\boldsymbol{\Pi'}$ corresponds to the set of assignments for two machines, then $\boldsymbol{A}_3 \in \mathcal{C}_+^{\boldsymbol{\Pi'}}$.

Another useful property of $\mathcal{C}_+^{\boldsymbol{\Pi}}$ is that it forms a convex cone, meaning that whether $\boldsymbol{A}, \boldsymbol{B} \in \mathcal{C}_+^{\boldsymbol{\Pi}}$, then $\alpha\boldsymbol{A} + \beta\boldsymbol{B} \in \mathcal{C}_+^{\boldsymbol{\Pi}}$ for any $\alpha, \beta \geq 0$. This property will be utilized later. First, we ask the question of whether it is possible to test the membership in $\mathcal{C}_+^{\boldsymbol{\Pi}}$ for a matrix $\boldsymbol{A}$ efficiently. Since $\boldsymbol{\Pi}$ is a finite set for any n and m (although a large one), one could enumerate all its elements and test the inequality for each element of $\boldsymbol{\Pi}$. However, a more efficient way exists. The test whether a given matrix $\boldsymbol{A} \in \mathbb{S}_+^n$ is contained in $\mathcal{C}_+^{\boldsymbol{\Pi}}$ can be performed in $\mathcal{O}(n^2 \log n)$ time. The idea is the following. If the copositivity condition holds, there must not exist a pair of $\boldsymbol{\pi} \in \boldsymbol{\Pi}$ and $i \in \mathcal{J}$, such that $\boldsymbol{e}_i^\intercal \boldsymbol{A}\boldsymbol{\pi} < 0$, where $\boldsymbol{e}_i = (0, \ldots, 0, 1, 0, \ldots, 0)$ is an i-th basis vector. Essentially, it selects i-th row of matrix $\boldsymbol{A}$ and multiplies it with a $\boldsymbol{\pi}$, which as to be a non-negative number. If one wants to know if this holds for the given i for any $\boldsymbol{\pi} \in \boldsymbol{\Pi}$, then it is enough to examine the worst-case $\boldsymbol{\pi}$ vector. That is, the test checks for each $i \in \mathcal{J}$, whether $\min_{\boldsymbol{\pi} \in \boldsymbol{\Pi}} \boldsymbol{e}_i^\intercal \boldsymbol{A}\boldsymbol{\pi} \geq 0$. The minimum can be evaluated by sorting $\boldsymbol{e}_i^\intercal \boldsymbol{A}$ and $\boldsymbol{\pi}$; — assigning the lowest values of $\boldsymbol{e}_i^\intercal \boldsymbol{A}$ with the highest of $\boldsymbol{\pi}$. This step takes $\mathcal{O}(n \log n)$ time, and thus the overall complexity is $\mathcal{O}(n^2 \log n)$.

In the rest of this section, we will analyze the properties of problem DR-PTFT(ℓ_1) with copositive covariance matrices. We will show that solutions of the problem (3.9) have similar robust properties as in ℓ_2 case. When $\hat{\boldsymbol{\Sigma}}^{1/2} \in \mathcal{C}_+^{\boldsymbol{\Pi}}$, the solution of (3.9) corresponds exactly to a distributionally robust solution over a specific ambiguity set.

Proposition 2. *Assuming $\hat{\boldsymbol{\Sigma}}^{1/2} \in \mathcal{C}_+^{\boldsymbol{\Pi}}$, the problem DR-PTFT($\ell_1$) is a distributionally robust formulation for $P||\sum C_j$ with an ambiguity set given by*

$$\mathcal{D}_{\ell_1} = \left\{ P \in \mathcal{P}_0(\mathbb{R}^n) \;\middle|\; \begin{array}{l} \mathbb{P}_P[\tilde{\boldsymbol{p}} \geq \boldsymbol{0}] = 1 \\ \mathbb{E}_P[\tilde{\boldsymbol{p}}] \leq \hat{\boldsymbol{\mu}} + \gamma_1 \cdot \hat{\boldsymbol{\Sigma}}^{1/2}\boldsymbol{1} \end{array} \right\}.$$

Proof. We will start with a high-level sketch of the proof. We substitute the definition of an ambiguity set $\mathcal{D}_{\ell_1}$ into the stochastic formulation of the DRO problem considered (1.4). We focus on the inner expectation problem of finding the worst-case probability distribution. First, we reformulate the inner problem using the definition of the expected value. Then, we derive the dual problem and observe that strong duality holds (both problems have the same optimal solutions). Finally, we transform the dual problem and substitute it back into the outer problem (finding optimal $\boldsymbol{\pi}$), obtaining the DR-PTFT(ℓ_1) problem.

Starting with the definition of the problem in (1.4)

$$\min_{\boldsymbol{\pi} \in \boldsymbol{\Pi}} \max_{P \in \mathcal{D}} \mathbb{E}_P\big[f(\boldsymbol{\pi}, \tilde{\boldsymbol{p}})\big] = \min_{\boldsymbol{\pi} \in \boldsymbol{\Pi}} \max_{P \in \mathcal{D}} \boldsymbol{\pi}^\intercal \mathbb{E}_P\big[\tilde{\boldsymbol{p}}\big]. \tag{3.10}$$

Let us focus on the inner maximum, i.e., finding the worst-case probability distribution P from the ambiguity set $\mathcal{D}$. By letting $\mathcal{D} \equiv \mathcal{D}_{\ell_1}$, the maximum can be calculated from the definition of the expected value

$$\max_{P \in \mathcal{D}_{\ell_1}} \int_{\mathbb{R}_+^n} \mathrm{f}_P(\boldsymbol{p}) \boldsymbol{\pi}^\intercal \boldsymbol{p} \; d\boldsymbol{p} \tag{3.11}$$

$$\text{s.t.} \int_{\mathbb{R}_+^n} \mathrm{f}_P(\boldsymbol{p}) \; d\boldsymbol{p} = 1 \tag{3.12}$$

$$\int_{\mathbb{R}_+^n} \mathrm{f}_P(\boldsymbol{p}) \boldsymbol{p} \; d\boldsymbol{p} \leq \hat{\boldsymbol{\mu}} + \gamma_1 \cdot \hat{\boldsymbol{\Sigma}}^{1/2}\boldsymbol{1} = \boldsymbol{h}, \tag{3.13}$$

where f_P is a probability density function of probability distribution P and $\boldsymbol{p}$ is a value in the support (i.e., the set of possible realizations) of $\tilde{\boldsymbol{p}}$, i.e., $\mathbb{R}^n_+$. Now, we will derive a dual problem for (3.11)–(3.13). Essentially, we put constraints into the objective multiplied with newly introduced multipliers. The Lagrangian of the problem is given by the equation

$$L(P, \alpha, \boldsymbol{\beta}) = \int_{\mathbb{R}^n_+} \mathrm{f}_P(\boldsymbol{p}) \Big(\boldsymbol{\pi}^\intercal \boldsymbol{p} - \alpha - \boldsymbol{\beta}^\intercal \boldsymbol{p} \Big) \, d\boldsymbol{p} + \alpha + \boldsymbol{\beta}^\intercal \boldsymbol{h}, \tag{3.14}$$

where $\alpha \in \mathbb{R}$ and $\boldsymbol{\beta} \in \mathbb{R}^n_+$ are the introduced Lagrange multipliers. The dual Lagrangian function is obtained with taking maximum over the original variables. Thus, the dual for the Lagrangian above is

$$\begin{aligned} g(\alpha, \boldsymbol{\beta}) &= \max_{P \in \mathcal{D}_{\ell_1}} L(P, \alpha, \boldsymbol{\beta}) && (3.15) \\ &= \max_{P \in \mathcal{D}_{\ell_1}} \left(\int_{\mathbb{R}^n_+} \mathrm{f}_P(\boldsymbol{p}) \Big(\boldsymbol{\pi}^\intercal \boldsymbol{p} - \alpha - \boldsymbol{\beta}^\intercal \boldsymbol{p} \Big) \, d\boldsymbol{p} + \alpha + \boldsymbol{\beta}^\intercal \boldsymbol{h} \right) && (3.16) \\ &= \alpha + \boldsymbol{\beta}^\intercal \boldsymbol{h} + \max_{P \in \mathcal{D}_{\ell_1}} \int_{\mathbb{R}^n_+} \mathrm{f}_P(\boldsymbol{p}) \Big(\boldsymbol{\pi}^\intercal \boldsymbol{p} - \alpha - \boldsymbol{\beta}^\intercal \boldsymbol{p} \Big) \, d\boldsymbol{p}. && (3.17) \end{aligned}$$

If there exists $\boldsymbol{p} \in \mathbb{R}^n_+$, such that $\boldsymbol{\pi}^\intercal \boldsymbol{p} - \alpha - \boldsymbol{\beta}^\intercal \boldsymbol{p} \geq 0$, then g is unbounded:

$$g(\alpha, \boldsymbol{\beta}) = \begin{cases} \alpha + \boldsymbol{\beta}^\intercal \boldsymbol{h} & \text{if } \boldsymbol{\pi}^\intercal \boldsymbol{p} - \alpha - \boldsymbol{\beta}^\intercal \boldsymbol{p} \leq 0, \\ +\infty & \text{otherwise.} \end{cases} \tag{3.18}$$

Finally, disregarding the unbounded case, the dual problem for (3.11) is

$$\begin{aligned} &\min_{\alpha, \boldsymbol{\beta}} \alpha + \boldsymbol{\beta}^\intercal \boldsymbol{h} && (3.19) \\ &\text{s.t. } \boldsymbol{\beta} \geq \mathbf{0} && (3.20) \\ &\qquad \alpha + \boldsymbol{\beta}^\intercal \boldsymbol{p} \geq \boldsymbol{\pi}^\intercal \boldsymbol{p} \quad \forall \boldsymbol{p} \in \mathbb{R}^n_+. && (3.21) \end{aligned}$$

The problem satisfies conic duality [30], thus strong duality holds. Therefore, an optimal solution to the dual is also optimal w.r.t. (3.11). By transforming the problem further, we obtain

$$\begin{aligned} &\min_{\alpha, \boldsymbol{\beta}} \alpha + \boldsymbol{\beta}^\intercal \boldsymbol{h} && (3.22) \\ &\text{s.t. } \boldsymbol{\beta} \geq \mathbf{0} && (3.23) \\ &\qquad (\boldsymbol{\pi}^\intercal - \boldsymbol{\beta}^\intercal) \boldsymbol{p} \leq \alpha \quad \forall \boldsymbol{p} \in \mathbb{R}^n_+. && (3.24) \end{aligned}$$

Next, the value of the left hand side expression in (3.24) must be investigated. With respect to variable α, there are two possible cases to consider:

1. $\exists j \in \mathcal{J} : \beta_j < \pi_j$. Then, because for $p_j \to +\infty$, left hand side of (3.24) goes to infinity, and also $\alpha \to +\infty$. Therefore, in this case, (3.22)–(3.24) becomes infeasible.

2. $\boldsymbol{\pi} \leq \boldsymbol{\beta}$. Then $\alpha = 0$ and then (3.22)–(3.24) can be written as

$$\begin{aligned} &\min_{\boldsymbol{\beta}} \boldsymbol{\beta}^\intercal \boldsymbol{h} && (3.25) \\ &\text{s.t. } \boldsymbol{\beta} \geq \mathbf{0} && (3.26) \\ &\qquad \boldsymbol{\pi} \leq \boldsymbol{\beta}, && (3.27) \end{aligned}$$

   and thus $\boldsymbol{\beta} = \boldsymbol{\pi}$.

This investigation shows that the optimal value for the multipliers are: $\alpha = 0$ and $\boldsymbol{\beta} = \boldsymbol{\pi}$, and the solution to the dual problem is $0 + \boldsymbol{\pi}^\intercal \boldsymbol{h} = \boldsymbol{\pi}^\intercal(\hat{\boldsymbol{\mu}} + \gamma_1 \cdot \hat{\boldsymbol{\Sigma}}^{1/2}\mathbf{1})$. Substituting the result into the outer minimization problem from (3.10), we have

$$\min_{\boldsymbol{\pi}\in\boldsymbol{\Pi}} \boldsymbol{\pi}^\intercal(\hat{\boldsymbol{\mu}} + \gamma_1 \cdot \hat{\boldsymbol{\Sigma}}^{1/2}\mathbf{1}) = \boldsymbol{\pi}^\intercal\hat{\boldsymbol{\mu}} + \gamma_1 \cdot \mathbf{1}^\intercal\hat{\boldsymbol{\Sigma}}^{1/2}\boldsymbol{\pi} = \boldsymbol{\pi}^\intercal\hat{\boldsymbol{\mu}} + \gamma_1 \cdot \left\|\hat{\boldsymbol{\Sigma}}^{1/2}\boldsymbol{\pi}\right\|_1, \quad (3.28)$$

where the last equality follows from the fact that $\hat{\boldsymbol{\Sigma}}^{1/2} \in \mathcal{C}_+^{\boldsymbol{\Pi}}$.

□

Ambiguity set $\mathcal{D}_{\ell_1}$ from Proposition 2 is relatively simple as it imposes the upper limit on the first moment of the random variables only. Utilizing ambiguity sets based only on the first moment is not uncommon, see for example, [32]. The main reason why they are being used is to improve the computational tractability of the resulting problem. On the other hand, our ambiguity set does not disregard the second moment. As it was explained in Section 3.2, the second moment is reflected in the same way as ML algorithms treat model complexity via regularization. This was shown in Proposition 2, which explains the link between DR-PTFT(ℓ_1) and ambiguity set $\mathcal{D}_{\ell_1}$ where parameter γ_1 is used to control the robustness of the solution. The principal advantage of ambiguity set $\mathcal{D}_{\ell_1}$ is its favorable computational properties, as reflected by the following complexity characterization:

Proposition 3. *Problem DR-PTFT(ℓ_1) is solvable in $\mathcal{O}(n \log n)$ time when $\hat{\boldsymbol{\Sigma}}^{1/2}$ is a diagonal matrix; and in $\mathcal{O}(n^2)$ when $\hat{\boldsymbol{\Sigma}}^{1/2} \in \mathcal{C}_+^{\boldsymbol{\Pi}}$.*

Proof. Using the fact that $\operatorname{diag}(\hat{\boldsymbol{\mu}}) \in \mathcal{C}_+^{\boldsymbol{\Pi}}$, $\hat{\boldsymbol{\Sigma}}^{1/2} \in \mathcal{C}_+^{\boldsymbol{\Pi}}$, $\gamma_1 \geq 0$ and $\mathcal{C}_+^{\boldsymbol{\Pi}}$ is a convex cone, problem DR-PTFT(ℓ_1) can be reformulated as

$$\begin{aligned}\min_{\boldsymbol{\pi}\in\boldsymbol{\Pi}} \hat{\boldsymbol{\mu}}^\intercal\boldsymbol{\pi} + \gamma_1 \cdot \|\hat{\boldsymbol{\Sigma}}^{1/2}\boldsymbol{\pi}\|_1 &= \min_{\boldsymbol{\pi}\in\boldsymbol{\Pi}} \left\| \left(\operatorname{diag}(\hat{\boldsymbol{\mu}}) + \gamma_1 \cdot \hat{\boldsymbol{\Sigma}}^{1/2}\right)\boldsymbol{\pi} \right\|_1 \\ &= \min_{\boldsymbol{\pi}\in\boldsymbol{\Pi}} \mathbf{1}^\intercal \left(\operatorname{diag}(\hat{\boldsymbol{\mu}}) + \gamma_1 \cdot \hat{\boldsymbol{\Sigma}}^{1/2}\right)\boldsymbol{\pi}. \end{aligned} \quad (3.29)$$

Next, let us denote $\boldsymbol{h} = \mathbf{1}^\intercal \left(\operatorname{diag}(\hat{\boldsymbol{\mu}}) + \gamma_1 \cdot \hat{\boldsymbol{\Sigma}}^{1/2}\right) \in \mathbb{R}^n$. We can see that the problem (3.29) is tantamount to deterministic $P||\sum C_j$ with job durations given by $\boldsymbol{h}$. There are known, efficient polynomial exact algorithms for the problem, based on sorting (for more details, refer to, e.g., [7, p. 133–134]). For the convenience of the reader, we present the full procedure in Algorithm 1. Vector $\boldsymbol{h}$ can be computed in $\mathcal{O}(n^2)$ (line 1), and as a result, the overall complexity is $\mathcal{O}(n^2 + n \log n) = \mathcal{O}(n^2)$. When $\hat{\boldsymbol{\Sigma}}^{1/2}$ is diagonal, the complexity is just $\mathcal{O}(n \log n)$. Note that when $\hat{\boldsymbol{\Sigma}}^{1/2}$ is not part of the input, then it needs to be computed from covariance matrix $\hat{\boldsymbol{\Sigma}}$ first, which can be done in $\mathcal{O}(n^3)$ time. □

The algorithm given by Proposition 3 is formulated in Algorithm 1. At line 2, the jobs are sorted in non-increasing order, by the weight defined in $\boldsymbol{h}$, which takes $\mathcal{O}(n \log n)$ time. Then, in the loop at lines 3–5, they are sequentially inserted into the solution. Each time a job is assigned to a machine with the least jobs assigned so far. In $\boldsymbol{\pi}$ representation, we only store the number of jobs preceding the job J on the machine, so the exact machine number is not computed. Each operation in the loop takes $\mathcal{O}(1)$ time, thus the entire loop takes $\mathcal{O}(n)$ time. In conclusion, line 1 determines the overall time complexity, depending on the form and values of

the covariance matrix. We refer to this algorithm as *Sort Optimizer with Ravishing Technique* (i.e., SORT(ℓ_1)).

Algorithm 1: Sort Optimizer with Ravishing Technique (SORT(ℓ_1))

input : Estimates of the parameters: $\hat{\boldsymbol{\mu}}$, $\hat{\boldsymbol{\Sigma}}^{1/2}$, trade-off parameter γ_1.
output: Optimal solution $\boldsymbol{\pi}$.

$\boldsymbol{h} \leftarrow \mathbf{1}^\intercal \left(\operatorname{diag}(\hat{\boldsymbol{\mu}}) + \gamma_1 \cdot \hat{\boldsymbol{\Sigma}}^{1/2}\right)$
jobs $\leftarrow$ sort jobs in non-increasing order, job $j \in \mathcal{J}$ has weight h_j
for $j = 1, 2, \ldots, n$ **do**
    J $\leftarrow$ jobs(j) `// Take the next job.`
    $\pi_{\mathsf{J}} \leftarrow \lceil \frac{j}{m} \rceil$
return $\boldsymbol{\pi}$

Obviously, when $\hat{\boldsymbol{\Sigma}}^{1/2} \notin \mathcal{C}_+^{\boldsymbol{\Pi}}$, then Proposition 3 does not apply, and it is an open question whether a polynomial algorithm for this case exists as well. In any case, DR-PTFT(ℓ_1) can be still expressed as a mixed-integer linear program (3.29) and solved by a general-purpose solver. As it will be shown in experiments in Section 5, even this method is very efficient, while allowing to tackle any covariance matrix.

4. Multi-objective optimization perspective

The purpose of this section is to point out a relation between the form of the objective function of the problem and the multi-objective optimization. In contrast to the previous sections, the aim here is to discuss some practical concerns related to the solution of the problem (3.5). One of them the obvious questions that the decision maker faces is, how to set the value of γ_1 parameter. Although there are some methods of how to set γ_1, e.g., to which extent we want to cover the target distribution [12], these will not provide the price to be paid for such a solution beforehand. Thus, we will argue that obtaining a single solution for some γ_1 is not likely to be very useful in practice. Instead, we will provide a method for uniform sampling of solutions in the Pareto front, which exhibits different optimal trade-offs between the mean and variance.

Finally, we reveal that the objective function of problem (3.3) directly optimizes the metrics used to assess its out-of-sample performance. We identify this as a striking difference from many other scheduling problems and their evaluation in DRO scheduling literature, where this correspondence is often not present.

4.1. Relation to multi-objective optimization

First, let us introduce the two solution quality metrics of a robust solution used in [8] — *robust price* (RP) and *robust benefit* (RB). They are defined with respect to some testing probability distribution $P \in \mathcal{P}_0(\mathbb{R}^n)$ of processing times $\tilde{\boldsymbol{p}} \sim P$:

$$\mathrm{RP}(\boldsymbol{\pi}^R) = \Big(\mathbb{E}_P\big[f(\boldsymbol{\pi}^R, \tilde{\boldsymbol{p}})\big] - \mathbb{E}_P\big[f(\boldsymbol{\pi}^D, \tilde{\boldsymbol{p}})\big]\Big) / \mathbb{E}_P\big[f(\boldsymbol{\pi}^R, \tilde{\boldsymbol{p}})\big], \tag{4.1}$$

$$\mathrm{RB}(\boldsymbol{\pi}^R) = \Big(\mathrm{Var}_P\big[f(\boldsymbol{\pi}^D, \tilde{\boldsymbol{p}})\big]^{1/2} - \mathrm{Var}_P\big[f(\boldsymbol{\pi}^R, \tilde{\boldsymbol{p}})\big]^{1/2}\Big) / \mathrm{Var}_P\big[f(\boldsymbol{\pi}^R, \tilde{\boldsymbol{p}})\big]^{1/2}. \tag{4.2}$$

The solution of the robust formulation (3.5) is denoted as $\boldsymbol{\pi}^R$, while $\boldsymbol{\pi}^D$ is an optimal solution of the problem with deterministic processing times, i.e., (3.5) with $\gamma_1 = 0$, with the processing times set as their true (or estimated) means. Thus, $\mathrm{RP}(\boldsymbol{\pi}^R)$ measures the relative difference between the expected quality of the robust

and deterministic solutions $\boldsymbol{\pi}^R$ and $\boldsymbol{\pi}^D$, respectively. Similarly, $\mathrm{RB}(\boldsymbol{\pi}^R)$ is the relative difference of standard deviations of solutions under the distribution P. Note that the testing distribution P may or may not be known; nevertheless, we assume that one has access to a finite sample set from P.

Example 4. Assume that for an instance with 4 jobs and a single machine, the parameters of the jobs are estimated as $\hat{\boldsymbol{\mu}} = (1.96, 1.39, 1.39, 1.39)$ and $\hat{\boldsymbol{\Sigma}} = \mathrm{diag}\,(0, 0, 0.072, 0.209)$. For this instance, consider two solutions: the deterministic (or SP, they are equivalent) one $\boldsymbol{\pi}^D = (1, 2, 3, 4)$ and the robust one $\boldsymbol{\pi}^R = (4, 3, 2, 1)$ (not necessarily optimal). As an example to demonstrate the above-defined quantities, let us pick a testing distribution $P = \mathcal{N}(\hat{\boldsymbol{\mu}}, \hat{\boldsymbol{\Sigma}})$ to display the empirical densities of the objective values of the aforementioned solutions (shown in Figure 3). Under this test distribution, $\boldsymbol{\pi}^D$ is an optimal solution for SP formulation. However, we see that for a price measured in terms of $\mathrm{RP}(\boldsymbol{\pi}^R)$, robust solution $\boldsymbol{\pi}^R$ achieves a smaller variance and also a smaller worst-case objective value, thus being a more stable solution for a risk-aware decision maker.

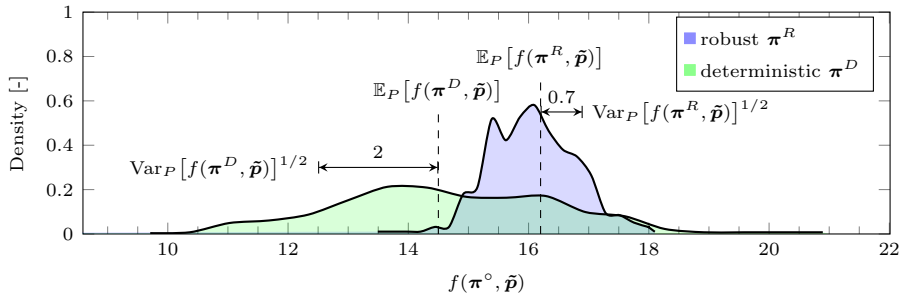

Figure 3: Illustration of robust price and robust benefit for different solutions $\boldsymbol{\pi}^\circ$, $\circ \in \{R, D\}$.

Obviously, one would like to have RP the smallest possible and RB the largest. Different robust solutions $\boldsymbol{\pi}^R$ can perform differently under these two, generally conflicting criteria. Thus, such a problem can be formulated from the perspective of the multi-objective optimization with the two criteria $g(\boldsymbol{\pi}) \in \mathbb{R}^2$:

$$\boldsymbol{\pi}^R = \arg\min_{\boldsymbol{\pi}\in\boldsymbol{\Pi}} g(\boldsymbol{\pi}) = \arg\min_{\boldsymbol{\pi}\in\boldsymbol{\Pi}} \big(\mathrm{RP}(\boldsymbol{\pi}), -\mathrm{RB}(\boldsymbol{\pi})\big). \tag{4.3}$$

What will be shown next is that formulation (3.3), in fact, solves this multi-objective optimization problem with the *scalarization approach* [13]. To show that, let us analyze the two criteria separately. For the RP, we have that

$$\begin{aligned}\boldsymbol{\pi}^\star_{RP} = \arg\min_{\boldsymbol{\pi}^R\in\boldsymbol{\Pi}} \mathrm{RP}(\boldsymbol{\pi}^R) &= \arg\min_{\boldsymbol{\pi}^R\in\boldsymbol{\Pi}} \left\{1 - \frac{\mathbb{E}_P\big[f(\boldsymbol{\pi}^D, \tilde{\boldsymbol{p}})\big]}{\mathbb{E}_P\big[f(\boldsymbol{\pi}^R, \tilde{\boldsymbol{p}})\big]}\right\} = \arg\min_{\boldsymbol{\pi}^R\in\boldsymbol{\Pi}} \mathbb{E}_P\big[f(\boldsymbol{\pi}^R, \tilde{\boldsymbol{p}})\big] \\ &= \arg\min_{\boldsymbol{\pi}^R\in\boldsymbol{\Pi}} \boldsymbol{\mu}^\intercal \boldsymbol{\pi}^R \approx \arg\min_{\boldsymbol{\pi}^R\in\boldsymbol{\Pi}} \hat{\boldsymbol{\mu}}^\intercal \boldsymbol{\pi}^R,\end{aligned}$$

where $\boldsymbol{\mu}$ is (potentially unknown) mean value of $\tilde{\boldsymbol{p}}$ and $\hat{\boldsymbol{\mu}}$ is the sample mean obtained from P. The second equality follows from the fact that $\mathbb{E}_P[f(\boldsymbol{\pi}^D, \tilde{\boldsymbol{p}})]$ is a constant as long as P is fixed. Analogously for RB, we obtain

$$\begin{aligned}\boldsymbol{\pi}^\star_{RB} = \arg\max_{\boldsymbol{\pi}^R\in\boldsymbol{\Pi}} \mathrm{RB}(\boldsymbol{\pi}^R) &= \arg\max_{\boldsymbol{\pi}^R\in\boldsymbol{\Pi}} \left\{\frac{\mathrm{Var}_P[f(\boldsymbol{\pi}^D, \tilde{\boldsymbol{p}})]^{1/2}}{\mathrm{Var}_P[f(\boldsymbol{\pi}^R, \tilde{\boldsymbol{p}})]^{1/2}} - 1\right\} = \arg\min_{\boldsymbol{\pi}^R\in\boldsymbol{\Pi}} \mathrm{Var}_P[f(\boldsymbol{\pi}^R, \tilde{\boldsymbol{p}})]^{1/2} \\ &= \arg\min_{\boldsymbol{\pi}^R\in\boldsymbol{\Pi}} \|\boldsymbol{\Sigma}^{1/2}\boldsymbol{\pi}^R\|_2 \approx \arg\min_{\boldsymbol{\pi}^R\in\boldsymbol{\Pi}} \|\hat{\boldsymbol{\Sigma}}^{1/2}\boldsymbol{\pi}^R\|_2,\end{aligned}$$

where $\boldsymbol{\Sigma}$ is (potentially unknown) covariance and $\hat{\boldsymbol{\Sigma}}$ is the sample covariance of $\tilde{\boldsymbol{p}}$. Similarly, term $\mathrm{Var}_P[f(\boldsymbol{\pi}^D, \tilde{\boldsymbol{p}})]^{1/2}$ acts as a constant. Thus, it can be seen that $\boldsymbol{\pi}^{\star}_{RP} \in \boldsymbol{\Pi}$ minimizing $\hat{\boldsymbol{\mu}}^{\intercal}\boldsymbol{\pi}$ also minimizes RP$(\cdot)$. Similarly, $\boldsymbol{\pi}^{\star}_{RB} \in \boldsymbol{\Pi}$ minimizing $\|\hat{\boldsymbol{\Sigma}}^{1/2}\boldsymbol{\pi}\|_2$ minimizes $-$RB$(\cdot)$. Thus, it implies that (3.3) can be viewed as a scalarization method for the multi-objective optimization applied to (4.3). What else can be concluded is that formulation (4.3) suggests that a solution in the sense of ℓ_1 norm obtained from (3.9) is also likely to work well as a solution for multi-objective problem since $\|\boldsymbol{x}\|_1 \geq \|\boldsymbol{x}\|_2$ for any $\boldsymbol{x} \in \mathbb{R}^n$. Thus, its robust term acts as an upper bound on $\mathrm{Var}_P\left[f(\tilde{\boldsymbol{p}}, \boldsymbol{\pi})\right]^{1/2}$, which in turn leads to optimization of RB as well. Note that when one measures the out-of-sample performance in terms of RB and RP, then the optimization criterion matches the performance metric. This is not necessarily always the case with DRO scheduling problems described in the literature since they often just draw several samples from a mix of distribution (whose does not necessarily correspond to the worst-case distribution). After that, some quantities of the objective function are reported, such as the expected value, maximum value or different quantiles. This leads us again back to the discussion in Section 2.2, where we have outlined some challenges when evaluating a DRO solution. Thus, we believe that this aspect of our problem worths pointing out.

In the following section, we describe an improved RP/RB trade-off parametrization, which provides a more uniform sampling of the Pareto front.

4.2. Pareto front sampling

As it was shown in the above section, parameter γ_1 in (3.1) can be used by the decision maker to control the trade-off between RP and RB of the resulting solution. However, an obvious disadvantage of such parametrization is that the actual value of γ_1 needed to achieve certain RB depends on the numerical scale of sample covariance matrix $\hat{\boldsymbol{\Sigma}}$. Concerning the practical use of such parametrization, it is difficult to guess desired values for γ_1. For a decision maker, an ideal parametrization would allow to choose any point on the RP/RB trade-off curve and obtain the desired balance between robustness and average performance of the system without having to re-run the solving procedure multiple times.

Unfortunately, a computationally efficient exact approach to this task might be hard to find. Instead, we propose a simple—yet useful—heuristic. The idea is to normalize the effect of parameter $\gamma_1 \geq 0$ with respect to the sample mean $\hat{\boldsymbol{\mu}}$ and covariance $\hat{\boldsymbol{\Sigma}}$. We introduce a single parameter $r \in [0, 1]$, that controls the emphasis between RP and RB. The problem with r becomes

$$\text{DR-PTFT}(\ell_p, r) \equiv \min_{\boldsymbol{\pi}\in\boldsymbol{\Pi}} \frac{1-r}{0.5n \cdot \hat{\boldsymbol{\mu}}^{\intercal}\mathbf{1}} \cdot \hat{\boldsymbol{\mu}}^{\intercal}\boldsymbol{\pi} + \frac{r}{\left\|0.5n \cdot \hat{\boldsymbol{\Sigma}}^{1/2}\mathbf{1}\right\|_p^a} \cdot \left\|\hat{\boldsymbol{\Sigma}}^{1/2}\boldsymbol{\pi}\right\|_p^a, \qquad (4.4)$$

where $a \in \mathbb{N}$ is used when the ℓ_p norm is raised to the a-th power (e.g., ℓ_2 norm squared, see Remark 1). The denominators play the role of normalization constants, by the values the both terms might likely attain. Technically, these values also depend on the number of machines m; however, we have observed that for small values of m it does not affect it heavily. Equation (4.4) is designed such that for $\boldsymbol{\pi} = \frac{n}{2} \cdot \mathbf{1}$ (note that such $\boldsymbol{\pi}$ is infeasible, yet it may represent an "averaged" solution for a small m), both terms are normalized and the overall value is invariant in respect to $r \in [0, 1]$, i.e.,

$$\frac{1-r}{0.5n \cdot \hat{\boldsymbol{\mu}}^{\intercal}\mathbf{1}} \cdot \hat{\boldsymbol{\mu}}^{\intercal} \cdot 0.5n \cdot \mathbf{1} + \frac{r}{\left\|0.5n \cdot \hat{\boldsymbol{\Sigma}}^{1/2}\mathbf{1}\right\|_p^a} \cdot \left\|\hat{\boldsymbol{\Sigma}}^{1/2} \cdot 0.5n \cdot \mathbf{1}\right\|_p^a = 1 \quad \forall r \in [0, 1]\,.$$

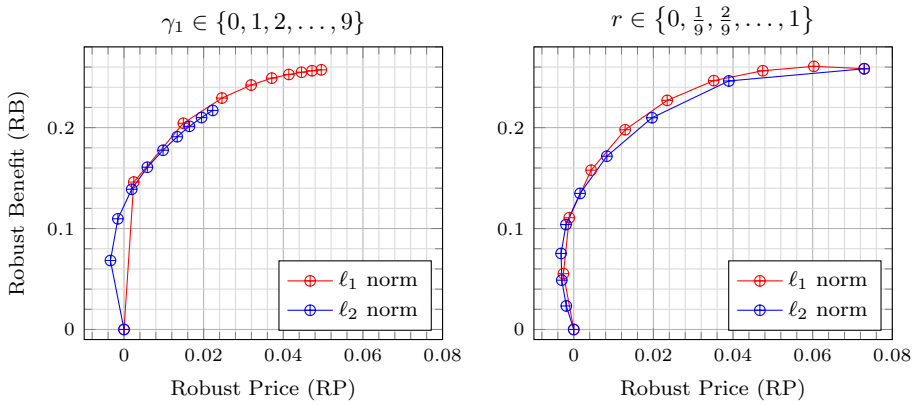

Figure 4: Distribution of solutions on RP/RB trade-off curves for different parametrizations.

Note that DR-PTFT($\ell_p, r = 0$) is equivalent to DR-PTFT($\ell_p, \gamma_1 = 0$), thus the solution of (4.4) with $r = 0$ resembles a Stochastic Programming solution as well.

To assess the benefits of the new parametrization, we have performed the following experiment. We generated 100 random instances, for each combination of $n \in \{10, 15, 20, 30, 50, 100, 150\}$ and $m \in \{3, 4, 5\}$. Each instance was solved both with ℓ_1 and ℓ_2 norms and with parametrizations using γ_1 and r. For each solution, RP and RB were calculated according to (4.1)–(4.2) and averaged over all instances. See the results in Figure 4. On the left-hand side, we can see RP/RB trade-off curves with the default parametrization using 10 values of $\gamma_1 \in \{0, 1, 2, \ldots, 9\}$, applied to both problems with ℓ_1 and ℓ_2 norm. We can see that in the case of ℓ_1 and ℓ_2 norm, the points on the Pareto front tend to get dense with the increasing value of γ_1 (i.e., the greater γ_1, the greater RB). On the right-hand side, we see the results for parametrization using $r \in \{\frac{0}{9}, \frac{1}{9}, \frac{2}{9}, \ldots, \frac{9}{9}\}$ (i.e., also 10 different values). We can observe that for ℓ_1 norm, this parametrization leads to solutions distributed evenly along the Pareto front. Similarly, for ℓ_2 norm, the parametrization using r also leads to a more even distribution of the solutions, than in the case of γ_1. Hence, in subsequent experiments, the parametrization with r will be utilized instead of γ_1.

5. Numerical experiments

In this section, we perform the experimental evaluation of the proposed methods. Specifically, for problem DR-PTFT(ℓ_1) we benchmark the sort-based method from Proposition 3 (denoted as SORT(ℓ_1)), and the MILP model given by Equation (3.28) (i.e., MILP(ℓ_1)). For problem DR-PTFT(ℓ_2), we evaluate the SOCP given by Equation (3.1) (i.e., SOCP(ℓ_2)), and the NOC-points Search Algorithm (denoted as NPSA(ℓ_2) introduced by Chang *et al.* [8]. Algorithm NPSA(ℓ_2) inspects a finite number of so-called NOC-points (*necessary optimality condition* points) that are suspected of being extreme. For more details, we refer the reader to [8].

In the case of problem DR-PTFT(ℓ_2^2), we assume the min-cost bipartite perfect matching with Hungarian algorithm [17] mentioned in Remark 1 (i.e., HUNGARIAN(ℓ_2^2)). We study the quality, robustness of the solutions, and computation times of the algorithms, for problems with respect to different norms, both for independent and dependent jobs.

5.1. Experimental setup

For experimental evaluation, we have used a workstation equipped with AMD Ryzen Threadripper 3990X @2.90GHz (during computations boosted to about 4.00 GHz), 64 GB RAM, running Windows 10 Pro. Due to the operating system shortcomings (limited support for more than 64 cores per CPU), Simultaneous Multi-Threading (SMT) was disabled. All algorithms have been implemented in Python 3. As a solver, we have used Gurobi 9.0 with default parameter configuration. The source codes and test instances can be found at [OMITTED FOR THE REVIEW PROCESS]. When measuring time, a single run of an algorithm utilizes a single CPU core. Otherwise, all 64 physical cores are available for Gurobi solver (in case of solving MILP(ℓ_1) and SOCP(ℓ_2)); however, it rarely utilizes more than 12 and never more than 32.

5.2. Evaluation protocol

To test the robustness and quality of solutions, we adopted the evaluation protocol proposed in [8]. The protocol focuses on independent jobs and compares robust and deterministic solutions from the perspective of average quality and stability. It proceeds as follows:

1. Generate 1,000 random $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, representing the true moments of distributions. The mean duration for each job μ_j is generated as $\mu_j \sim \mathcal{U}(10, 60)$ and its standard deviation is distributed $\sigma_j \sim \mathcal{U}(0.1\mu_j, 0.9\mu_j)$.

2. For each $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$, the protocol generates 10,000 random samples (i.e., one sample is a single realization of $\tilde{\boldsymbol{p}}$) from the mix of distributions: Gamma, uniform, normal and Laplace, with the given fixed mean and variance. From each distribution, 2,500 samples are taken. Then, the samples are shuffled randomly to simplify the next step.

3. From each set of 10,000 random samples, the protocol creates several sets of subsamples by selecting the given samples. When $n \leq 20$ it creates 100 subsample sets and when $n > 20$ it is only 20 of them. The size of the subsample set is determined by a sample rate (referred to as *S-rate* in [8]). If it is not specified otherwise, we assume 10 subsamples in each subsample set. Each subsample set is used to define the ambiguity set by estimates $\hat{\boldsymbol{\mu}}$ and $\hat{\boldsymbol{\sigma}}$ of the given true moments $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. Solution of (3.5) and its deterministic counterpart result in $\boldsymbol{\pi}^R$ and $\boldsymbol{\pi}^D$ respectively.

4. For each $\boldsymbol{\pi}^\circ$, $\circ \in \{R, D\}$, their expectations $\mathbb{E}_P[f(\boldsymbol{\pi}^\circ, \tilde{\boldsymbol{p}})]$ and standard deviations $\mathrm{Var}_P[f(\boldsymbol{\pi}^\circ, \tilde{\boldsymbol{p}})]^{1/2}$ are estimated from $500,000$ samples from the mix of distributions with the given fixed true moments $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$. The samples are generated in the same way as in the third step.

5. Estimate RP and RB for each distribution given by its true moments $\boldsymbol{\mu}$ and $\boldsymbol{\sigma}$ via sample mean along the subsamples.

Note that since [8] dealt with independent jobs only, it did not provide a protocol for dependent jobs' durations. Hence, in this paper, we propose the following modification for dependent jobs. True expected values of job durations were taken from the uniform distribution $\mathcal{U}(10, 50)$, but we replace the generation of the true $\boldsymbol{\sigma}$, with a full true covariance matrix $\boldsymbol{\Sigma}$. In order to cover a wide range of distributions, we consider covariance matrices to be samples from the Wishart distribution. Wishart distribution is a distribution over symmetric positive definite matrices, possibly with

some negative off-diagonal elements. It appears as a distribution over sample covariance matrices produced by samples from a multivariate normal distribution. In our experiments, the distribution of covariances is given as $\hat{\boldsymbol{\Sigma}} \sim W_n(\nu, \lambda \cdot \boldsymbol{I}) + \operatorname{diag} \boldsymbol{d}$ where $\boldsymbol{d} \sim \mathcal{N}(\boldsymbol{\mu'}, 1)$. Hence, they are samples from Wishart distribution $W_n(\,\cdot\,,\,\cdot\,)$ with ν degrees of freedom and a scale matrix given as $\lambda \cdot \boldsymbol{I}$ of $n \times n$ real symmetric PSD matrices. Furthermore, we add a normally distributed vector $\boldsymbol{d}$ to the diagonal to control the likelihood that $\hat{\boldsymbol{\Sigma}}$ is copositive with respect to $\boldsymbol{\Pi}$. We have chosen $\boldsymbol{\mu'} = 5 \cdot \mathbf{1}$, $\lambda = 1$ and $\nu = n + 40$. After a true covariance matrix $\boldsymbol{\Sigma}$ is generated, an algorithm has then access to a limited number of samples from a mix of multivariate normal and multivariate uniform distributions with covariance $\boldsymbol{\Sigma}$. The reason behind using just these two, and not all four types as in the independent case, is that for many distributions, there is no single, well-established multivariate extension. It follows from the fact, that the covariance has a good meaning as a measure of variability for symmetric, elliptically contoured distributions. That is, it is not straightforward to define a natural extension of these distributions, and—as a result—multiple possible generalizations exist, often emerge from different applications (see, e.g., [18]). Therefore, in our experiments addressing dependent jobs' durations, we have limited ourselves to multivariate uniform and multivariate Gaussian distributions. The details regarding generating covariance matrices are covered in Sections 5.4 and 5.5.

In the following sections, we utilize RP and RB obtained via the above evaluation protocol as the primary performance metrics based on the out-of-sample evaluation.

5.3. Independent jobs: quality, stability and performance

In this section, we compare the quality of solutions obtained by assuming different objective functions, for the case of independent jobs. In addition, we present computational times of different algorithms. All the variants of the problem are solved exactly with respect to their objective functions, i.e., ℓ_1 norm is solved with our SORT(ℓ_1) method, ℓ_2 norm with SOCP(ℓ_2) and ℓ_2^2 norm with HUNGARIAN(ℓ_2^2). In the evaluation of the computational times, we also present the state-of-the-art method NPSA(ℓ_2) proposed in [8].

Effect of the used ℓ_p norm. The first experiment studies the trade-off between RP and RB when different norms are used. The instances are generated and evaluated according to the protocol described in Section 5.2. The results for instances with $n \in \{10, 15, 20, 30, 50, 100, 150\}$ and $m \in \{3, 4, 5\}$ are displayed in Figure 5 altogether, as we have observed that the general shape of trade-off curves do not depend heavily on the particular values of m and n. A more detailed comparison of the differences between the individual curves with particular values of m and n for ℓ_1 norm is displayed in Figure 6.

The coordinates of each point on the curve in Figure 5 correspond to RP and RB of the solution averaged over $100 \times 7 \times 3 = 2100$ instances. Each point is obtained with a different value of r parameter, i.e., the position of a point on the curve is completely parameterized by r. We have taken 100 values of r distributed uniformly on $[0, 1]$ for each method. In Figure 5, it can be seen that all the methods achieve comparable trade-off curves in terms of ℓ_1, ℓ_2 and ℓ_2^2 (although technically speaking ℓ_2^2 is not a norm), yet each norm gets more advantage in different parts of the curve. Thus, they are incomparable but essentially identical. What is particularly interesting, is that all methods allow obtaining solutions with a positive RB while having a negative RP. Thus, in this setting, a free lunch is possible and one can get a more stable and cheaper solution than the deterministic one. It is likely due to the fact that the deterministic solution completely lacks the information about variances, which

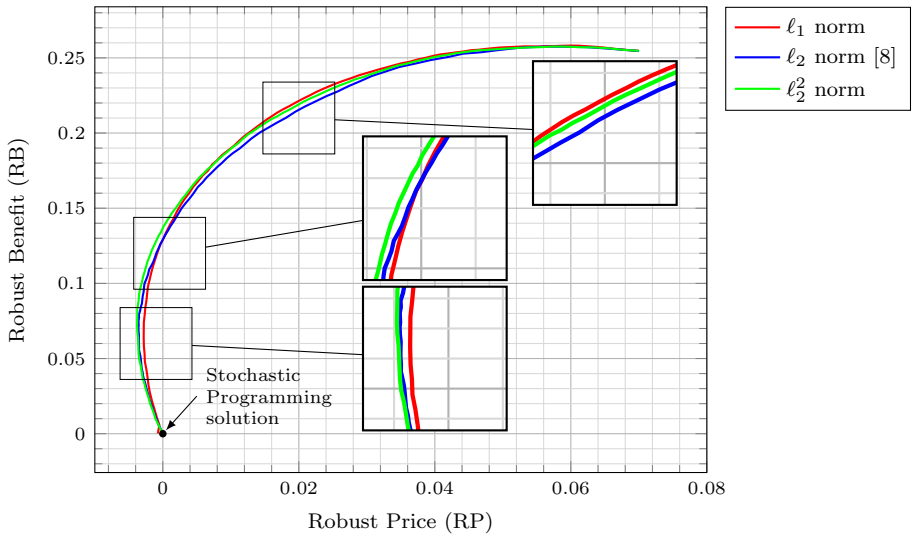

Figure 5: Trade off between RP and RB for ℓ_1, ℓ_2 and ℓ_2^2 formulations with independent jobs.

robust solutions can take advantage of. An interesting question to ask is, how the choice of the particular norm affects the decision maker. As we have seen in Figure 5, ℓ_1 norm achieves similar trade-offs between RP and RB as ℓ_2 norm. On one hand, the advantage of ℓ_2 norm may be seen by the fact that it exactly resembles the standard deviation of the solution objective, thus the weight of the norm term can be directly interpreted. On the other hand, ℓ_1 norm does not offer this, but its advantage for the decision maker lies in its practical computational tractability. In other words, with ℓ_1 norm it is affordable to compute the whole RP/RB curve and pick any solution that suits the requirements of the decision maker.

Computational times. In this experiment, we provide a measuring of time needed to solve the problem depending on the used norm and algorithm. In Figure 7 and Table 1, we display comparison of computational times SOCP(ℓ_2), SORT(ℓ_1), MILP(ℓ_1), and NPSA(ℓ_2). The x axis in Figure 7 represents instances with different values of n and m. Axis y depicts the computational time in seconds. Note the logarithmic scale of the y axis. Each data point is given by an average of over 100 instances. One can see that SOCP(ℓ_2) is computationally the most expensive, while MILP(ℓ_1) and SORT(ℓ_1) are far less demanding. Indeed, MILP(ℓ_1) is 10 times faster than SOCP(ℓ_2) while SORT(ℓ_1) is even more than three orders of magnitude faster than SOCP(ℓ_2) for the largest instances. In addition, all the methods solving DR-PTFT(ℓ_1) have consistent running times — the error bars (± 1 sigma) are virtually non-existent. While it is not surprising for SORT(ℓ_1), in the case of MILP(ℓ_1) this phenomenon is explained by the observation that the solver has solved every problem instance in the root node, yielding to consistent times. For more detailed results see Table 1. There, we can see that for a larger number of machines m, the problem becomes simpler. For the largest instances with $n = 150$ jobs, SOCP(ℓ_2) has lower average number of visited nodes than for $n = 100$. This is due to occasional timeouts that have occurred on the largest instances, especially for larger values of r which put more emphasis on the robust term. Nevertheless, the average optimality gaps reported by the solver were in these cases very small.

Next, we have compared our algorithms to the state-of-the-art method NPSA(ℓ_2)

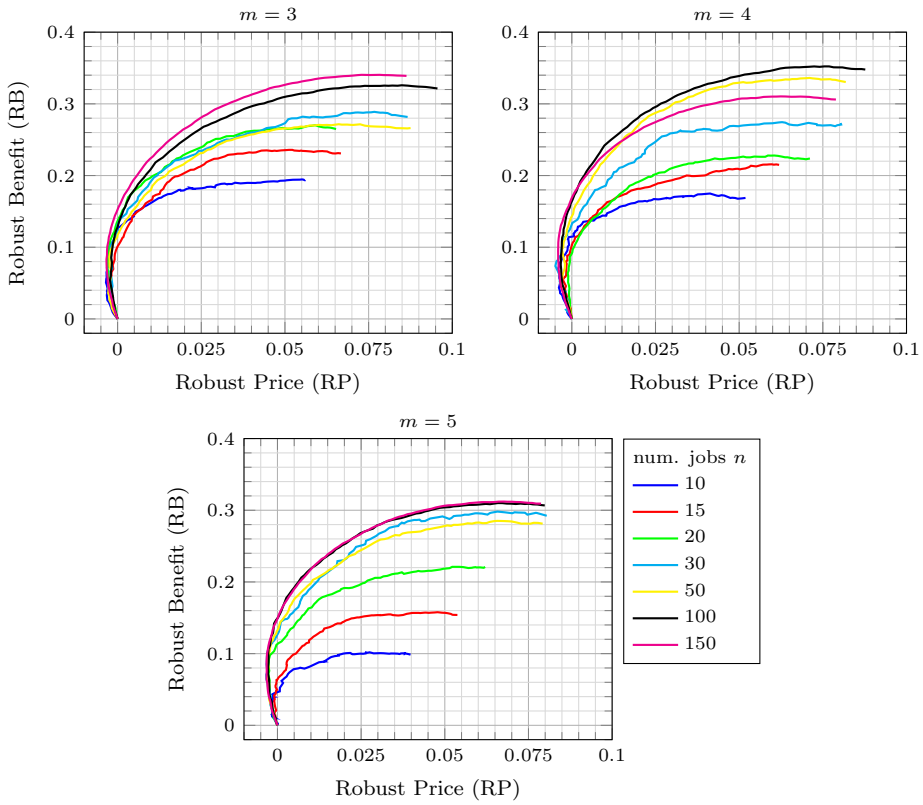

Figure 6: Relation between RP/RB curve and instance parameters for ℓ_1 formulation.

proposed in [8]. It is an exact algorithm solving the same problem as SOCP(ℓ_2), but with independent jobs only. For the sake of comparison, we have scaled their runtimes by the relative single-core performance of their and our CPU (approximately 1.5 times). It can be observed that even though NPSA(ℓ_2) outperforms SOCP(ℓ_2), it is still much slower than SORT(ℓ_1). The error bars are not displayed, as they were not reported in their paper.

To summarize, the results show that (i) one can obtain comparable RP/RB trade-offs for the problem with ℓ_1 norm as for ℓ_2 norm, and (ii) the computational time for the problem with ℓ_1 is much shorter than with ℓ_2 norm. Moreover, ℓ_1 formulation enjoys polynomial-time exact algorithm, while for ℓ_2 there is no such guarantee on the time required for calculations. The likely non-existence of this polynomial time bound is reflected in a higher spread of computation times observed for SOCP(ℓ_2), with some instances taking over 1000 times longer to be solved than the average. Although it may not be that dramatic for instances benchmarked in Figure 7, this difference even increases with the size of the instance, turning the solution to instances with more than hundreds of jobs nearly intractable with SOCP(ℓ_2). Such large instances occur, e.g., when scheduling unit test batches (as described in the introduction), with tens of thousands of jobs possible.

5.4. Dependent jobs: quality, stability and performance

In this section, we study the computational properties of the problem with dependent jobs. Specifically, we investigate: (i) what are the benefits of using information

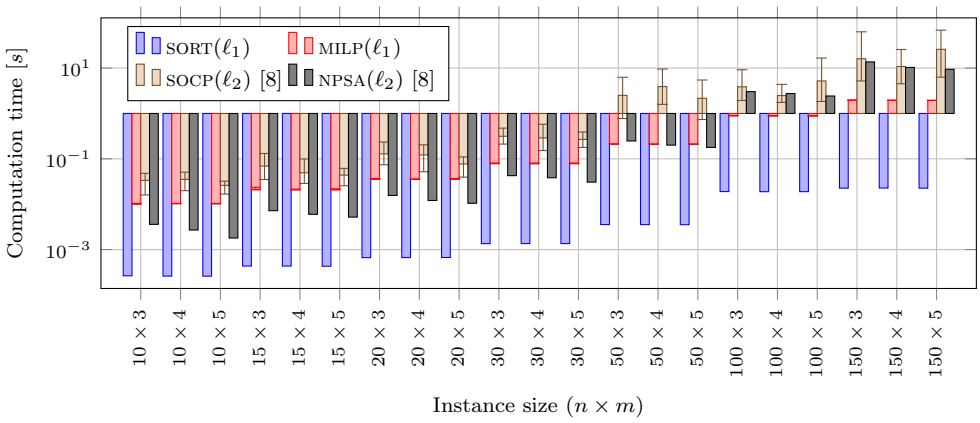

Figure 7: Comparison of averaged computational times for different methods and instance sizes for problem with independent jobs and $\gamma_1 = 4$.

about the correlations between jobs to quality/stability of the schedule, and (ii) what amount of data is needed to reliably estimate covariance matrix such that it brings a meaningful benefit over just a diagonal covariance. Furthermore, we report computational times needed to solve such a problem, depending on the properties of the covariance matrix and the used norm.

Effect of the used ℓ_p norm. In the first experiment, we compare RP/RB trade-off curves obtainable with ℓ_1 and ℓ_2 norms. The experiment assumes perfect information about the covariance, i.e., algorithms have access to the true covariance matrix. Since the solution of a single instance with $n = 10$ jobs with full covariance takes more than one hour to compute with SOCP(ℓ_2) model, we have limited ourselves to instances only with $n = 10$ and $m = 3$ in this experiment. As the solution of ℓ_1 formulation is much faster, we took 100 values of r uniformly distributed on $[0, 1]$ while for ℓ_2 formulation, we have used 25 values of r. The results can be seen in the left part of Figure 8.

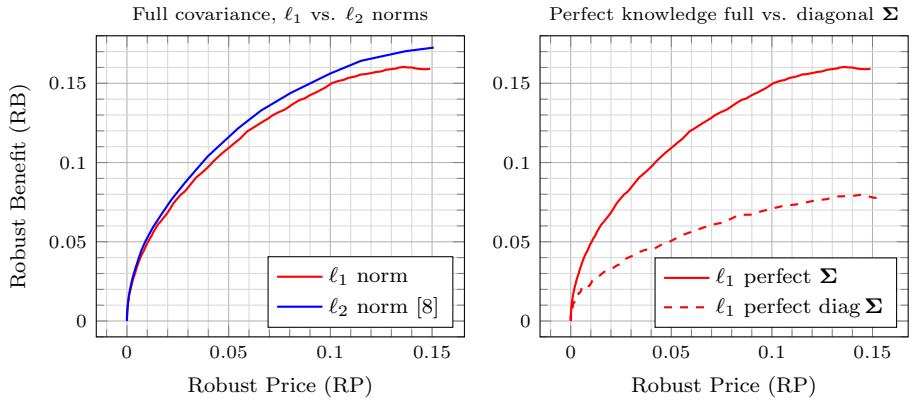

Figure 8: Trade-off between RP and RB with ℓ_1 and ℓ_2 formulations for dependent jobs with perfect covariance knowledge and its diagonal part.

Similarly, as for the independent case, the trade-off curves are very similar to each other, but here the solution with ℓ_2 achieves slightly better RB values. We believe

Table 1: Detailed comparison of performance indicators for independent jobs.

		SORT(ℓ_1)	NPSA(ℓ_2) [8]	MILP(ℓ_1)		SOCP(ℓ_2) [8]		
n	m	time [s]	time [s]	time [s]	nodes [$-$]	time [s]	nodes [$-$]	gap [%]
10	3	0.0003	0.0036	0.010	0.0	0.03	2.76	0.001
	4	0.0003	0.0027	0.010	0.0	0.04	2.36	0.000
	5	0.0003	0.0018	0.010	0.0	0.03	1.94	0.000
15	3	0.0004	0.0072	0.021	0.0	0.07	5.64	0.001
	4	0.0004	0.0060	0.021	0.0	0.05	5.93	0.001
	5	0.0004	0.0052	0.021	0.0	0.04	4.97	0.001
20	3	0.0007	0.0156	0.036	0.0	0.13	21.62	0.002
	4	0.0007	0.0121	0.036	0.0	0.12	11.34	0.002
	5	0.0007	0.0105	0.036	0.0	0.08	9.88	0.001
30	3	0.0014	0.0426	0.079	0.0	0.31	399.74	0.003
	4	0.0014	0.0383	0.079	0.0	0.29	139.16	0.003
	5	0.0014	0.0307	0.079	0.0	0.27	47.28	0.002
50	3	0.0036	0.2477	0.212	0.0	2.50	2377.50	0.003
	4	0.0035	0.2006	0.210	0.0	3.91	1885.92	0.007
	5	0.0035	0.1784	0.210	0.0	2.16	449.87	0.004
100	3	0.0190	3.0238	0.892	0.0	3.86	36029.88	0.015
	4	0.0189	2.7509	0.888	0.0	2.48	24225.20	0.010
	5	0.0189	2.4252	0.885	0.0	5.21	20566.85	0.012
150	3	0.0226	13.6820	1.964	0.0	16.07	12858.49	0.033
	4	0.0227	10.3278	1.953	0.0	10.84	13555.14	0.020
	5	0.0226	9.4144	1.942	0.0	25.95	15514.36	0.024

that the reason for it is related to the fact that PSD matrices are closely connected to ℓ_2 norm, which also takes a unique role among the different ℓ_p norms. That is, ℓ_2 is the only norm among ℓ_p norms which is induced by a scalar product, and by Riesz representation theorem, its dual norm is also ℓ_2. Actually, every scalar product on $\mathbb{R}^n$ corresponds to exactly one positive-definite $n \times n$ matrix. We believe the ℓ_2 norm is, therefore, more intuitive to work with when working with PSD matrices, and hence, leads to slightly better results. These differences did not play a significant influence for the case of independent jobs, but it seems to have a bigger impact for the dependent case. However, what will be shown in a subsequent experiment, when one does not have the perfect knowledge of covariance (i.e., it has to be estimated from a finite sample set), then the differences between ℓ_1 and ℓ_2 norms become negligible again. Thus, even with dependent jobs, both norms allow obtaining solutions of comparable quality/stability, especially considering the fact that the solution with ℓ_1 norm is much faster. Again, we see this as a benefit for the decision maker which can afford to compute the whole RP/RB trade-off curve and choose the desired balance between these two.

Full and diagonal covariances with perfect information. The second experiment assesses the effect of using the full covariance matrix, assuming the perfect knowledge of it. Hence, in this setting, the algorithm has access to the true covariance matrix and we measure which values of RP/RB are achievable compared to the case when one uses just its diagonal part. The results are displayed in the right part of Figure 8. There, we have generated 100 instances with $n = 10$ jobs and $m = 3$ machines. Each curve is obtained with 100 values of r parameter controlling the trade-off between

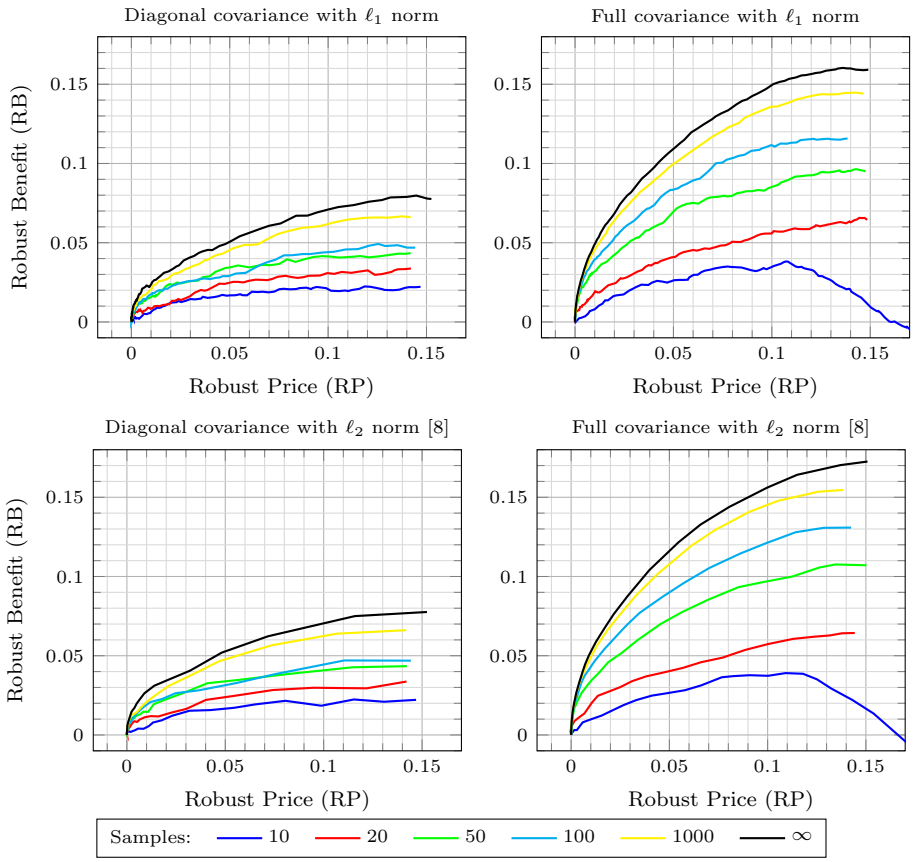

Figure 9: Effect of S-rate to RP/RB curve with dependent jobs.

RP and RB. The solid curve is obtained using full covariance with the perfect information; hence, it represents an upper bound on the RP/RB curve. The dashed curve represents the performance of a solution when just the diagonal part of the covariance matrix is used, i.e., when (potential) dependency between jobs is ignored (but being tested against distributions with non-diagonal covariance). It can be seen that using the information from the full covariance matrix allows obtaining about two times larger RB for the same RP than using just its diagonal part.

Full covariance with imperfect information. In practice, one might not have access to the true covariance, but rather the covariance needs to be estimated from empirical data. Hence, in the following experiment, we study how many data samples are needed to obtain an estimate of the covariance matrix, which actually produces an additional benefit over the ignorance of mutual correlations.

The setup is the following. We generate 100 instances with $n = 10$ and $m = 3$. Then, we follow the evaluation protocol described in Section 5.2, with the exception that S-rate is now a parameter whose effect is investigated. The tested values of S-rate are $0.001, 0.002, 0.005, 0.01$ and 0.1, which corresponds to $10, 20, 50, 100$ and 1000 samples used for the estimation of mean $\hat{\boldsymbol{\mu}}$ and covariance $\hat{\boldsymbol{\Sigma}}$.

The results are shown in Figure 9, where the results on the left side ignore the mutual correlations and the results on the right assume the full covariance. Curves for ℓ_1 norm were obtained using 100 values of r parameter wheres the curves for ℓ_2

Table 2: Computational times of different solving methods and instance sizes for dependent jobs.

Instance	MILP-DIAG(ℓ_1)		MILP(ℓ_1)		SOCP(ℓ_2) [8]	
$n \times m$	time [s]	std [s]	time [s]	std [s]	time [s]	std [s]
8×3	6.44×10^{-3}	1.7×10^{-5}	7.29×10^{-3}	1.8×10^{-4}	0.14	0.08
9×3	7.83×10^{-3}	1.1×10^{-5}	8.66×10^{-3}	8.1×10^{-5}	0.30	0.22
10×3	9.41×10^{-3}	5.6×10^{-5}	1.04×10^{-2}	1.1×10^{-4}	3.09	3.88
11×3	1.10×10^{-2}	2.4×10^{-5}	1.22×10^{-2}	1.3×10^{-4}	100.13	787.20
12×3	1.29×10^{-2}	3.9×10^{-5}	1.46×10^{-2}	9.9×10^{-4}	488.95	1087.30

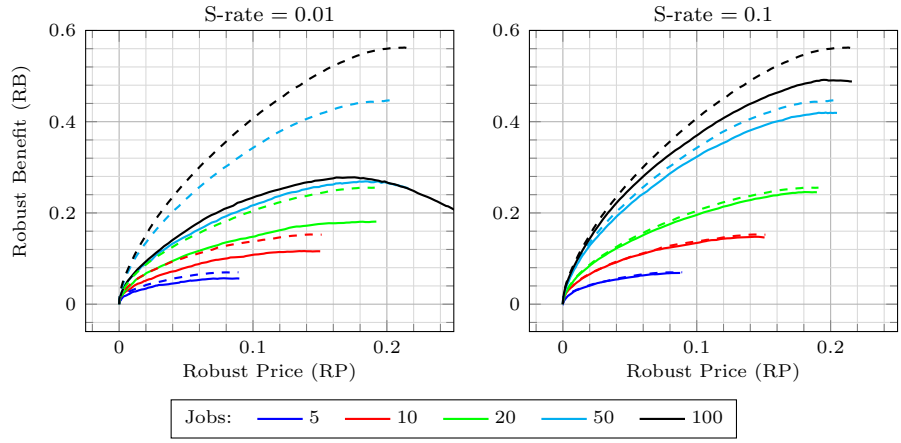

Figure 10: Effect of the number of dependent jobs n to RP/RB curve under ℓ_1 norm with variable S-rate.

with 25 values (since its solution is much more computationally demanding). The top curve, denoted as ∞ samples, corresponds to results when the algorithm has the perfect knowledge of the covariance. When the number of samples is decreasing, the achievable RB for a fixed RP is decreasing as well. However, it can be seen that when the number of samples drops below a certain level, then we may obtain a solution that performs even worse than the solution ignoring mutual correlations, i.e., assuming independence. This effect can be observed in Figure 9 for the full covariance estimated with just 10 samples, where for some RP values the solution has worse RB than when just the diagonal is estimated. Again, the trade-off curves are comparable for both ℓ_1 and ℓ_2 norms. Moreover, the computational results displayed in Table 2 show that both the solution in terms of ℓ_1 norm with full covariance (i.e., MILP(ℓ_1)) and just its diagonal part (denoted as MILP-DIAG(ℓ_1)) are much faster than SOCP(ℓ_2). As a result, the above experiment suggests that when enough data is available, then it is advantageous for the decision maker to solve the problem with full covariance rather than assuming the independence between jobs as it provides more protection against solution variance.

Scaling with respect to the sample quantity. Evidently, the amount of data that is needed to achieve the required RB depends on the size of the problem instance, i.e., the number of jobs n. Therefore, we have performed an experiment, where we fixed S-rate to 0.01 and 0.1, and we change the number of jobs $n \in \{5, 10, 20, 50, 100\}$ while we keep the number of machines $m = 3$. The results are displayed in Figure 10, where

Table 3: Computational times of different solving methods and instance sizes for dependent jobs with copositive covariance matrices.

Instance	SORT(ℓ_1)		MILP(ℓ_1)		SOCP(ℓ_2) [8]	
$n \times m$	time [s]	std [s]	time [s]	std [s]	time [s]	std [s]
8×3	2.21×10^{-4}	1.03×10^{-6}	6.0×10^{-3}	7.02×10^{-5}	0.13	0.08
9×3	2.45×10^{-4}	1.04×10^{-6}	7.3×10^{-3}	3.18×10^{-5}	0.29	0.40
10×3	2.72×10^{-4}	1.54×10^{-6}	9.0×10^{-3}	5.44×10^{-5}	1.29	1.78
11×3	3.07×10^{-4}	1.97×10^{-6}	1.1×10^{-2}	2.53×10^{-5}	7.07	13.79
12×3	3.39×10^{-4}	1.97×10^{-6}	1.2×10^{-2}	3.44×10^{-5}	25.88	59.59

the RP/RB curves are obtained by MILP(ℓ_1) with a varying number of jobs n, but with a fixed S-rate (0.01 on the left side, 0.1 on the right side). For each n, two curves are reported — the dashed curve is the one with perfect information, whereas the solid curve corresponds to the case when the limited number of samples (given by S-rate) is available. Therefore, the absolute values of RB are not that important (as it changes with the number of jobs n), but the difference between the two curves matters. We can see that, e.g., for $n = 50$ jobs, the S-rate equal to 0.1 (i.e., 1000 samples) is essentially enough to obtain a theoretically optimal trade-off curve (see the right plot in Figure 10). On the other hand, with S-rate equal to 0.01 (i.e., 100 samples), a similar level of discrepancy between the two curves is achieved for just $n = 10$ jobs (see the left plot in Figure 10). These values for S-rate correspond to the number of free parameters of a PSD matrix that need to be estimated, which is roughly quadratic in n.

5.5. Copositive covariance matrices

In this section, we focus on copositive covariance matrices, which is a special class of covariance matrices allowing us to solve the problem in polynomial time, as shown in Section 3.5. There are several natural questions connected with this class of covariance matrices. For example, do they bring any additional benefits in terms of RP/RB curve in comparison to using just their diagonal part? How often these matrices appear among the ones generated by the evaluation protocol, and is the solution of the problem in terms of ℓ_1 norm still comparable to ℓ_2 norm?

The setup is similar to the experiments in Section 5.4. Due to the limited performance of SOCP(ℓ_2) with dependent jobs, we have restricted the comparison to the instances with $m = 3$ machines and the maximum of $n = 12$ jobs. The true covariances are drawn from a distribution over PSD matrices described in Section 5.4, with the same parameters, i.e., $\nu = n + 40$, $\lambda = 1$ and $\boldsymbol{\mu'} = 5 \cdot \mathbf{1}$. With the given parameters, we have observed that the sampled matrices are copositive with respect to the used values of m and n in about 30% of cases and these were used in the following experiments.

Effect of the used ℓ_p norm. First, we discuss achievable trade-off curves. The results are displayed in Figure 11. There, we can see several essential differences from the results for general covariance matrices presented in Figure 9. First, we see that differences between ℓ_1 and ℓ_2 are much smaller for copositive covariance matrices in comparison to general ones (displayed in the left plot of Figure 8). Next, it can be seen that the achievable RB for a fixed RP is about 0.02 (for $n = 10$ jobs) smaller than in the case of general covariance matrices. Both these observations are explained by the fact that the copositive matrices tend to be more diagonally

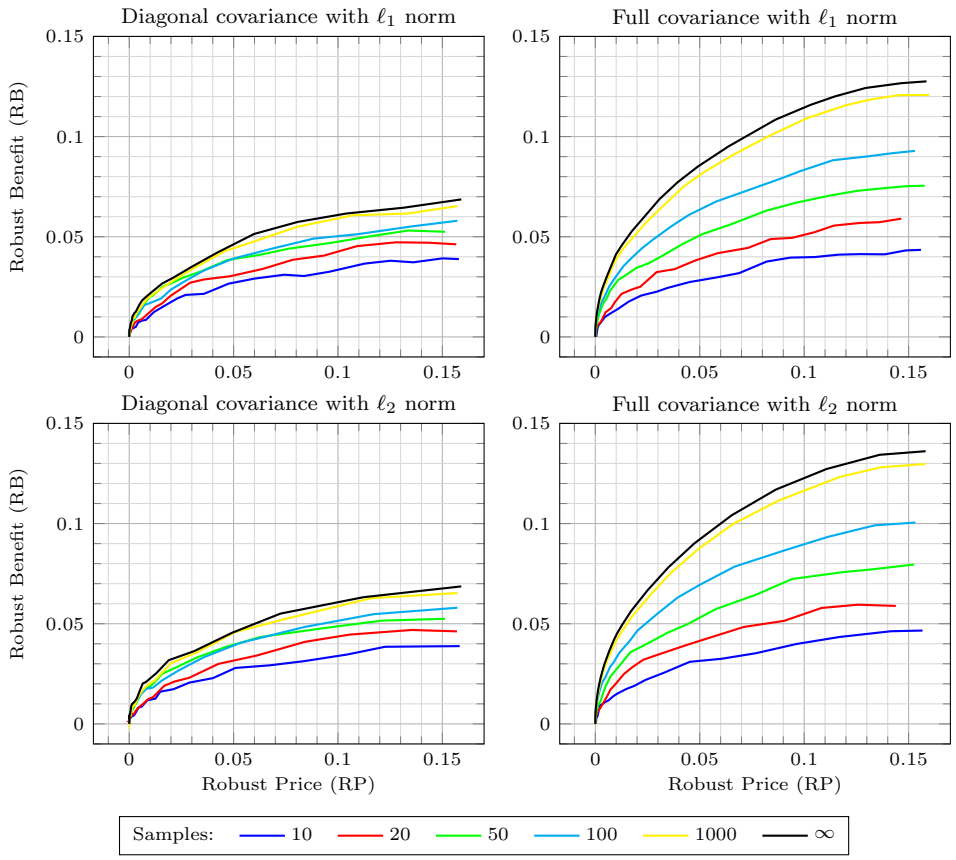

Figure 11: Effect of S-rate to RP/RB curve with copositive covariance matrices for instances with $n = 10$ and $m = 3$.

dominant, which is in line with the results observed in experiments with independent jobs. Furthermore, the results also show that using a full covariance matrix brings significantly more robustness than using just a diagonal part of the matrix. Thus, dealing with copositive covariance matrices is indeed meaningful.

Computational times. The last experiment measures the computational times of different methods. Interestingly, when we compare computational times of SOCP(ℓ_2) method for general covariance matrices in Table 2 and the computational times for copositive covariance matrices in Table 3, we see that the runtimes are significantly smaller for the later ones. This again points to a relation between copositive and diagonally dominant matrices, for which SOCP(ℓ_2) scales better than in the general case. On the other hand, for MILP(ℓ_1) method, the computational times are comparable regardless of the type of covariance matrix. Finally, it can be seen that SORT(ℓ_1) method is superior, being about 100 times faster than MILP(ℓ_1), which indicates a good scaling to even larger instances (given by polynomial computational complexity).

6. Conclusion

In this paper, we study a distributionally robust scheduling problem with the total flow time criterion. The distribution of uncertain processing times is subject

to ambiguity belonging to a set of distributions with constrained first two central moments. A prior work [8] has established that such a problem can be translated into a second-order conic programming problem. We have noticed that this optimization problem can be viewed as a minimization of a linear function plus a regularization term expressed in terms of ℓ_2 norm. A natural question immediately arises — is the use of a particular norm essential, or can it be replaced with some other ℓ_p norm, perhaps with more favorable computational properties while providing a similar level of robustness?

We answer this question affirmatively. We have provided a characterization of complexity for the problem with independent jobs in the sense of any ℓ_p norm. As a special case of our theorem, we have improved the upper bound on the complexity for the case of ℓ_2 formulation proposed in [8]. For the ℓ_1 norm, we obtained even stronger results, leading to a polynomial-time algorithm. For the case of dependent jobs, we identified a class of covariance matrices admitting an efficient, polynomial-time solution algorithm, when ℓ_1 regularization term is used. Interestingly, carefully conducted experiments have shown that solutions with ℓ_1 regularization term provide almost identical trade-offs between the quality and robustness to the more complex ℓ_2 regularization. This result comes as a surprise, considering that the best-known solution for dependent jobs in the sense of ℓ_2 regularization is able to solve only problems with 10 jobs and 3 machines within an hour, wheres our algorithm for ℓ_1 can successfully solve instances with hundreds of jobs, and for a class of generalized positive covariance matrices, is of polynomial time complexity.

The results also demonstrate the importance of utilizing the information about potential correlations between jobs, even when one does not have the perfect knowledge of covariance. This realistic case also shows that it is not crucial to use the formulation with ℓ_2 norm, but it can be replaced with ℓ_1 norm with essentially identical quality and stability — at a much reduced computational cost. This stimulates to study further the relation between tractable solutions for (conservative) ambiguity sets and approximate solutions for more expressive (but intractable) DRO formulations in environments with limited data availability. It is subject to a further study to which extent are the ideas developed in this paper applicable for more complex objective functions such as, e.g., total tardiness. Furthermore, the complexity of ℓ_1 formulation for general covariance matrices remains as an open question as well structural differences between the solutions obtained with different regularization norms.

Acknowledgement

We would like to thank our colleague Matej Novotny for numerous illuminating discussions. This work was supported by the European Regional Development Fund under the project AI&Reasoning (reg. no. CZ.02.1.01/0.0/0.0/15_003/0000466).

References

[1] S. Alimoradi, M. Hematian, G. Moslehi, Robust scheduling of parallel machines considering total flow time, Computers & Industrial Engineering 93 (2016) 152–161, ISSN 0360-8352.

[2] N. Balakrishnan, B. Scarpa, Multivariate measures of skewness for the skew-normal distribution, Journal of Multivariate Analysis 104 (1) (2012) 73–87, ISSN 0047-259X, URL https://www.sciencedirect.com/science/article/pii/S0047259X11001400.

[3] G. Bayraksan, D. K. Love, Data-driven stochastic programming using phi-divergences, in: The Operations Research Revolution, INFORMS, 1–19, 2015.

[4] A. Ben-Tal, E. Hochman, More bounds on the expectation of a convex function of a random variable, Journal of Applied Probability 9 (4) (1972) 803–812.

[5] Y. Berstein, S. Onn, Nonlinear bipartite matching, Discrete Optimization 5 (1) (2008) 53–65, ISSN 1572-5286.
[6] D. Bertsimas, V. Gupta, N. Kallus, Data-driven robust optimization, Mathematical Programming 167 (2) (2018) 235–292.
[7] P. Brucker, Scheduling Algorithms, Springer Berlin Heidelberg, ISBN 978-3-540-69516-5, 2007.
[8] Z. Chang, J.-Y. Ding, S. Song, Distributionally robust scheduling on parallel machines under moment uncertainty, European Journal of Operational Research 272 (3) (2019) 832–846, ISSN 03772217.
[9] M. Charikar, A. Sahai, Dimension reduction in the ℓ_1 norm, in: The 43rd Annual IEEE Symposium on Foundations of Computer Science, 2002. Proc., IEEE, 551–560, 2002.
[10] J. Cheng, E. Delage, A. Lisser, Distributionally robust stochastic knapsack problem, SIAM Journal on Optimization 24 (3) (2014) 1485–1506.
[11] S.-J. Chung, NP-completeness of the linear complementarity problem, Journal of optimization theory and applications 60 (3) (1989) 393–399.
[12] E. H. Delage, Distributionally robust optimization in context of data-driven problems, 2009.
[13] M. T. Emmerich, A. H. Deutz, A tutorial on multiobjective optimization: fundamentals and evolutionary methods, Natural computing 17 (3) (2018) 585–609.
[14] C. Gambella, B. Ghaddar, J. Naoum-Sawaya, Optimization Problems for Machine Learning: A Survey, European Journal of Operational Research ISSN 0377-2217.
[15] D. Ge, X. Jiang, Y. Ye, A note on the complexity of ℓ_p minimization, Mathematical programming 129 (2) (2011) 285–299.
[16] P. Indyk, Stable distributions, pseudorandom generators, embeddings and data stream computation, in: Proceedings 41st Annual Symposium on Foundations of Computer Science, IEEE, 189–197, 2000.
[17] R. Jonker, A. Volgenant, A shortest augmenting path algorithm for dense and sparse linear assignment problems, Computing 38 (4) (1987) 325–340.
[18] S. Kotz, T. J. Kozubowski, K. Podgórski, Asymmetric multivariate Laplace distribution, in: The Laplace distribution and generalizations, Springer, 239–272, 2001.
[19] A. Kramer, M. Dell'Amico, M. Iori, Enhanced arc-flow formulations to minimize weighted completion time on identical parallel machines, European Journal of Operational Research 275 (1) (2019) 67–79, ISSN 0377-2217.
[20] S. Leonardi, D. Raz, Approximating total flow time on parallel machines, Journal of Computer and System Sciences 73 (6) (2007) 875–891, ISSN 0022-0000.
[21] C.-C. Lu, S.-W. Lin, K.-C. Ying, Robust scheduling on a single machine to minimize total flow time, Computers & Operations Research 39 (7) (2012) 1682–1691, ISSN 0305-0548.
[22] S. N. Majumdar, A. Pal, Extreme value statistics of correlated random variables, 2014.
[23] O. Mangasarian, R. Meyer, Absolute value equations, Linear Algebra and Its Applications 419 (2-3) (2006) 359–367.
[24] O. Mangasarian, R. Meyer, Absolute value equations, Linear Algebra and its Applications 419 (2) (2006) 359–367, ISSN 0024-3795.
[25] K. P. Murphy, Machine learning: a probabilistic perspective, MIT press, 2012.
[26] S. Niu, S. Song, J.-Y. Ding, Y. Zhang, R. Chiong, Distributionally robust single machine scheduling with the total tardiness criterion, Computers & Operations Research 101 (2019) 13–28, ISSN 0305-0548.
[27] H. Rahimian, S. Mehrotra, Distributionally robust optimization: A review, arXiv preprint arXiv:1908.05659 .
[28] H. Scarf, A min-max solution of an inventory problem, Studies in the mathematical theory of inventory and production .
[29] C. Shang, F. You, Distributionally robust optimization for planning and scheduling under uncertainty, Computers & Chemical Engineering 110 (2018) 53–68, ISSN 0098-1354.
[30] A. Shapiro, On Duality Theory of Conic Linear Problems, Springer US, Boston, MA, ISBN 978-1-4757-3403-4, 135–165, 2001.
[31] A. Shapiro, A. Nemirovski, On complexity of stochastic programming problems, in: Continuous optimization, Springer, 111–146, 2005.
[32] K. S. Shehadeh, A. E. Cohn, R. Jiang, A distributionally robust optimization approach for outpatient colonoscopy scheduling, European Journal of Operational Research 283 (2) (2020) 549–561, ISSN 0377-2217.
[33] K. Sneha, G. M. Malle, Research on software testing techniques and software automation testing tools, in: 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), 77–81, 2017.
[34] C. Sohler, D. P. Woodruff, Subspace embeddings for the l1-norm with applications, in: Proceedings of the forty-third annual ACM symposium on Theory of computing, 755–764, 2011.
[35] R. Wang, D. P. Woodruff, Tight Bounds for ℓ_p Oblivious Subspace Embeddings, 2018.
[36] Y. Wang, Y. Zhang, J. Tang, A distributionally robust optimization approach for surgery block

allocation, European Journal of Operational Research 273 (2) (2019) 740–753.
[37] Z. Wang, P. W. Glynn, Y. Ye, Likelihood robust optimization for data-driven problems, Computational Management Science 13 (2) (2016) 241–261.
[38] G. Xue, Y. Ye, An Efficient Algorithm for Minimizing a Sum of p-Norms, SIAM Journal on Optimization 10 (2) (2000) 551–579.