# Minimizing Value-at-Risk in Single-Machine Scheduling

Semih Atakan, Kerem Bülbül, Nilay Noyan
Sabancı University, Manufacturing Systems and Industrial Engineering, Orhanlı-Tuzla, 34956 İstanbul, Turkey.
semihatakan@sabanciuniv.edu, bulbul@sabanciuniv.edu, nnoyan@sabanciuniv.edu

Abstract: The vast majority of the machine scheduling literature focuses on deterministic problems in which all data is known with certainty a priori. In practice, this assumption implies that the random parameters in the problem are represented by their point estimates in the scheduling model. The resulting schedules may perform well if the variability in the problem parameters is low. However, as variability increases accounting for this randomness explicitly in the model becomes crucial in order to counteract the ill effects of the variability on the system performance. In this paper, we consider single-machine scheduling problems in the presence of uncertain parameters. We impose a probabilistic constraint on the random performance measure of interest, such as the total weighted completion time or the total weighted tardiness, and introduce a generic risk-averse stochastic programming model. In particular, the objective of the proposed model is to find a non-preemptive static job processing sequence that minimizes the value-at-risk (VaR) of the random performance measure at a specified confidence level. We propose a Lagrangian relaxation-based scenario decomposition method to obtain lower bounds on the optimal VaR and provide a stabilized cut generation algorithm to solve the Lagrangian dual problem. Furthermore, we identify promising schedules for the original problem by a simple primal heuristic. An extensive computational study on two selected performance measures is presented to demonstrate the value of the proposed model and the effectiveness of our solution method.

*Keywords*: single-machine scheduling; stochastic scheduling; value-at-risk; probabilistic constraint; stochastic programming; scenario decomposition; cut generation; dual stabilization; $K-$assignment problem

---

**1. Introduction** In the traditional single-machine problems, all parameters including the processing times, release dates, due dates, and weights are assumed to be known with certainty at the time the dispatcher determines the job processing sequence or the full schedule. However, in many practical settings the exact values of one or several of these parameters may not be available in advance. For instance, possible machine breakdowns, variable setup times, inconsistency of the worker performance, or changes in tool quality may introduce uncertainty into the processing times. The uncertainty in the processing time of a job is then only resolved at the time of the job completion. Along these lines, we focus on the uncertainty in the parameters which leads to random scheduling performance measures such as the total weighted completion time ($TWCT$), the total tardiness ($TT$), and the total weighted tardiness ($TWT$). Ultimately, our objective in this work is to determine a risk-averse fixed job processing sequence that hedges against the inherent uncertainty. In the stochastic scheduling terminology (see, e.g., Pinedo, 2008), we construct a non-preemptive static list policy.

Traditional models for decision making under uncertainty define optimality criteria based on expected values and disregard the variability inherent in the system. Following this mainstream risk-neutral approach, most of the classical stochastic scheduling puts a lot of effort into analyzing the expected performance by assuming that the uncertain parameters such as the processing times follow specific distributions. See Pinedo (2008) for an excellent overview of conventional stochastic scheduling. However, variability typically implies a deterioration in performance, and risk-neutral models may provide solutions that perform poorly under certain realizations of the random data. Capturing the effect of variability can be accomplished by incorporating appropriate risk measures into the models that reflect the preferences of the decision makers. Several criteria to select risk measures have been discussed in the literature (see, e.g., Artzner et al., 1999; Ogryczak and Ruszczyński, 2002; Pflug and Römisch, 2007). Considering the wide range of criteria, there is no universally accepted single risk measure appropriate for all decision making contexts. In this study, we consider VaR which is a very popular and widely applied risk measure in the finance literature. In our context, we select a commonly considered scheduling performance measure, such as $TWCT$, $TT$, or $TWT$, as the random outcome of interest associated with a fixed job processing sequence. The goal is to specify the smallest possible upper bound on the random performance measure that will be exceeded with at most a prespecified small probability. Here, the

determined upper bound (threshold) is the VaR of the random performance measure at the desired probability level, and we minimize VaR. The concept of VaR is closely related to probabilistic (chance) constraints. Stochastic programming models with chance constraints have been employed successfully in a variety of fields; the interested reader is referred to Prékopa (1995) and Dentcheva (2006) for reviews and a comprehensive list of references. In the single-machine scheduling literature, Akker and Hoogeveen (2008) study a chance constrained problem to minimize the number of late jobs; a job is considered to be on time if the probability that it is completed by its due date is at least equal to a given probability. However, the authors assume that the processing times follow specific distributions (normal, gamma, or negative binomial distribution) and the probability calculations require restrictive assumptions such as the independence of the stochastic processing times. On the other hand, Daniels and Carrillo (1997) and Wu et al. (2009) focus on maximizing the probability that the total completion time ($TCT$) is smaller than a given threshold. Thus, they focus on improving the probability level for a given threshold instead of improving the threshold for a given probability level. However, these studies also make restrictive distributional assumptions; Daniels and Carrillo (1997) assume that the processing times are independent and use the normal approximation for the $TCT$, while in Wu et al. (2009) the processing times are independent and normally distributed. We also note that the latter three studies on probabilistic scheduling only consider the randomness in the processing times and their modeling approaches cannot be generalized to incorporate randomness into the other parameters without making further restrictive assumptions on the corresponding joint distributions. Our scenario-based VaR minimization approach is an intuitive and practical alternative way of modeling a service level requirement for a selected performance measure under the stochastic setup and leads to a novel risk-averse stochastic programming model. To the best of our knowledge, this is a first in the machine scheduling literature.

We characterize the randomness associated with the uncertain parameters by a finite set of scenarios, where a scenario represents a joint realization of all random parameters. It is important to point out that the scenario approach allows us to generate data from any distribution and model the correlation of the random parameters among different jobs by considering their joint realizations. In this sense, a scenario-based approach is more general than assuming specific distributions. On the down side, the computational complexity of solving the problem is closely affected by the number of scenarios. There are relatively few studies utilizing a scenario-based approach for machine scheduling problems. Among these, we briefly review the relevant papers that consider a single-machine scheduling environment. Gutjahr et al. (1999) minimize the expected $TWT$ with stochastic processing times and propose a stochastic branch-and-bound technique, where a sampling approach is embedded into their bounding schemes. Alternatively, the other existing scenario-based studies mainly develop robust optimization models in order to optimize the worst-case system performance over all scenarios. Such a worst-case analysis does not require the probabilities of the scenarios. The sum of completion times is employed in Daniels and Kouvelis (1995); Yang and Yu (2002); Lu et al. (2012), and the weighted sum of completion times is considered by de Farias et al. (2010), while Kasperski (2005) focuses on the maximum lateness as the random performance criterion. One or several of the robustness measures known as the maximum deviation from optimality, the maximum relative deviation from optimality, and the maximum value over all scenarios are incorporated in these papers. Except de Farias et al. (2010), all these studies design specialized algorithms for the robustness measure and random performance criterion of interest. de Farias et al. (2010) identify a family of valid inequalities to strengthen the mixed-integer formulation of their problem. Furthermore, Aloulou and Croce (2008) provide several complexity results in the domain of robust single-machine scheduling. There also exist a few recent papers that propose scenario-based models for scheduling problems with multiple machines (see, e.g., Alonso-Ayuso et al., 2007; Kasperski et al., 2012); however, these also rely on risk-neutral or robust approaches.

In contrast to the robust approaches which adopt a conservative worst-case view, we define our optimality criterion based on VaR which is a quantile of the random outcome at a specified probability level. That is, we utilize probabilistic information and develop a risk-averse stochastic programming model alternative to existing robust optimization models. Note that setting the required probability level to exactly one subsumes

the robust optimization problem of minimizing the maximum performance measure over all scenarios. However, when the required probability level is specified as $\alpha < 1$, we minimize the maximum performance measure over a subset of the scenarios with an aggregate probability of at least $\alpha$. Our risk-averse model identifies the optimal subset of scenarios with the specified minimum aggregate probability level and minimizes the maximum performance measure over this subset. Thus, it is less conservative than the robust point of view which considers all scenarios. In our computational study, we also analyze the behavior of the proposed model in comparison to that of the corresponding risk-neutral model and provide insights on the impact of incorporating the risk preference. Note that a preliminary version of this work was presented in a conference proceeding (Atakan et al., 2011), where we introduced the concept of VaR for machine scheduling. However, that study only focuses on the total weighted tardiness as the performance measure and implements a standard tabu search heuristic to solve the proposed model. In this paper, we introduce a general modeling framework that can be applied to alternate random performance measures and develop a new effective mathematical programming based solution method.

It is well known that problems incorporating VaR exhibit a non-convex structure even if the underlying deterministic problem is convex. The approaches to solve such models are mainly based on approximation (see, e.g., Gaivoronski and Pflug, 2005) or global optimization methods (see, e.g., Pang and Leyffer, 2004; Wozabal, 2012); we refer to Larsen et al. (2002) and Wozabal et al. (2010) for a review of the available algorithms. These studies are generally concerned with portfolio optimization problems and the existing solution methods primarily deal with VaR integrated into a linear program (LP). Thus, the decision variables are continuous, and VaR is introduced on a random outcome expressed as a linear function of the decision variables. For instance, Larsen et al. (2002) introduce two heuristic algorithms which solve a series of problems involving a related risk measure known as conditional value-at-risk (CVaR). In contrast to VaR, the problem of minimizing CVaR can be formulated as an LP if the uncertainty is represented by a set of scenarios, and the proposed heuristics use LP techniques iteratively. However, in our study the underlying problem involves sequencing decisions that can only be expressed by employing binary variables; and therefore, even minimizing CVaR is hard. Consequently, such LP-based solution methods do not apply in our case.

Not limited to VaR, stochastic programming models are generally known to be computationally challenging. This can partially be attributed to the potentially large number of scenario-dependent variables and constraints. Introducing integer variables into stochastic programs further complicates solving these models. In the stochastic programming literature, the studies that focus on developing solution methods for stochastic integer programs mainly consider the risk-neutral two-stage framework. In such models, the first-stage decisions are taken before the uncertainty is revealed, while the second-stage (recourse) decisions are optimized given the first-stage decisions and the realized uncertainty. Various decomposition based solution methods have been proposed to deal with such large scale programs. The integer L-shaped algorithm proposed by Laporte and Louveaux (1993) is the first decomposition method for stochastic programs with integer decisions in the second-stage. It utilizes a branch-and-cut scheme in the master problem and requires pure binary first-stage decision variables. Carøe and Tind (1998) use the general duality theory and extend the integer L-shaped algorithm for the models with mixed-integer variables in both stages. Alternatively, Carøe and Schultz (1999) use the scenario decomposition approach of Rockafellar and Wets (1991) for the same setting and develop a branch-and-bound algorithm based on the Lagrangian relaxation of non-anticipativity. Recently, this solution approach has been adapted to two-stage stochastic integer programs incorporating risk measures such as excess probabilities (Schultz and Tiedemann, 2003) and CVaR (Schultz and Tiedemann, 2006). For a detailed discussion on various algorithms for stochastic integer programming we refer the reader to the overviews by Birge and Louveaux (1997); Klein Haneveld and van der Vlerk (1999); Louveaux and Schultz (2003); Sen (2005) and a comprehensive bibliography (van der Vlerk, 2007).

Among the existing methods discussed above, the L-shaped method and its variants exploit the specific directly decomposable two-stage structure and approximate the recourse function implicitly through cut gen-

eration. In contrast, Carøe and Schultz (1999) explicitly model the contribution of the second-stage decisions to the objective function, and in this paper we adapt their Lagrangian relaxation-based decomposition approach to obtain a lower bound for the optimal objective value of our stochastic integer programming model. A significant advantage of this approach is that it allows us to specifically deal with the VaR related linking constraint that couples the individual scenarios in our formulation. In particular, we consider a split-variable formulation which is essentially based on the idea of creating copies of the sequencing variables and then relaxing the constraints that force all these variables to be equal. In studies that focus on two-stage models (Carøe and Schultz, 1999; Schultz and Tiedemann, 2003), these constraints enforce that the first-stage decisions do not depend on the realized scenario and are referred to as "non-anticipativity" conditions. In our setting, non-anticipativity guarantees that the static job sequence is identical for all scenarios. We solve the Lagrangian dual problem by cut generation over a dynamically updated shrinking feasible region in an effort to improve the quality of the final lower bound. This is one of the most interesting characteristics of this paper from a methodological point of view. The cut generation algorithm is enhanced by dual stabilization to achieve faster convergence. The Lagrangian subproblems are solved through a set of combinatorial techniques that exploit their structure, and we also utilize parallel programming in order to improve the computational performance of our algorithm. In addition, we design a primal heuristic based on a simple local search that employs the Lagrangian subproblem solutions as starting points to find promising schedules for the original problem. The computed lower and upper bounds provide a quality certificate for our algorithm and they may also benefit alternate solution approaches – including branch and bound – in the future. We note that our proposed solution method is not limited to machine scheduling but could also be applied to a wide variety of VaR minimization problems. To the best of our knowledge, using such a variable splitting-based Lagrangian relaxation algorithm for minimizing VaR is a first in the literature.

In the next section, we formally define a general risk-averse machine scheduling problem and present the corresponding mathematical programming formulations. In Section 3, we introduce our scenario decomposition based solution method and discuss the details of the proposed cut generation algorithm and the primal heuristic. Section 4 is dedicated to additional algorithmic features and computational enhancements, while Section 5 presents the computational results. We conclude in Section 6 with further research directions.

**2. Optimization Models** In this section, we first present the underlying deterministic model of the generic stochastic single-machine scheduling problem under consideration. Then, we discuss how to model the uncertainty inherent in the system and develop our generic risk-averse stochastic programming model.

**2.1 Underlying Deterministic Model** A machine scheduling problem can be considered as a two-phase optimization problem. In the first phase, a feasible job processing sequence is determined for each machine involved, and then in the second phase the optimal start and completion times are computed for fixed job processing sequences. The difficult combinatorial structure of machine scheduling problems stems from the first phase, while the second phase - also referred to as the *optimal timing problem* - is a simple optimization problem for many important machine scheduling problems. On a single machine, the optimal timing problem is trivial for regular performance measures, and it can often be solved by a low-order polynomial time algorithm or as a linear programming problem for non-regular performance measures (Kanet and Sridharan, 2000). Moreover, a feasible job processing sequence can be expressed as an assignment (Keha et al., 2009), and for any assignment and performance measure the optimal job completion times and the resulting value of the performance measure can be identified by solving an appropriate optimal timing problem. This enables us to formulate a general single-machine scheduling model independent of the performance measure. In Section 3, we leverage on the separation of the sequencing and timing aspects of the model to design a combinatorial algorithm that can handle the subproblems in the proposed Lagrangian relaxation-based scenario decomposition for several scheduling performance criteria.

We define the set of jobs to be processed as $N := \{1, \ldots, n\}$, where $n$ denotes the number of jobs. Associated

with each job $j \in N$ are several parameters: a processing time $p_j$, a due date $d_j$ if the performance criterion is due date related – such as the maximum lateness or the total tardiness –, and a weight $w_j$ if job-dependent priorities are taken into account by the performance criterion. The binary variable $x_{jk}$ takes the value 1, if job $j$ is located at position $k$ in the processing sequence, and is zero otherwise. Assuming zero release dates, a deterministic single-machine scheduling problem with a generic objective function, described as $1//f(\boldsymbol{x})$ following the common three field notation of Graham et al. (1979), is formulated below:

$$(\mathbf{G}) \qquad \text{minimize} \quad f(\boldsymbol{x}) \tag{1}$$

$$\text{subject to} \quad \sum_{k\in N} x_{jk} = 1, \qquad \forall j \in N, \tag{2}$$

$$\sum_{j\in N} x_{jk} = 1, \qquad \forall k \in N, \tag{3}$$

$$x_{jk} \in \{0,1\} \qquad \forall j,k \in N. \tag{4}$$

Constraints (2)-(3) ensure that each job $j$ is allocated to a single position and each position $k$ accommodates a single job, respectively. Constraints (4) are the binary variable restrictions required for the sequencing decisions.

The constraints (2)-(4) model the sequencing aspect of the scheduling problem, and to customize the formulation for a specific performance measure of interest the objective function $f(\boldsymbol{x})$ of $(\mathbf{G})$ must be set as appropriate. This process is equivalent to integrating the timing aspect, and the formulation may need to be augmented with new variables and constraints to this end. For instance, the objective function of $(\mathbf{G})$ takes the form $f(\boldsymbol{x}) = \sum_{j\in N} p_j \sum_{k=1}^{n}(n-k+1)x_{jk}$ for minimizing $TCT$. It is also relatively simple to formulate the common due-date related performance measure $TT$. We denote the tardiness of the job at position $k$ by $T_k$, set $f(\boldsymbol{x}) = \sum_{k\in N} T_k$, and enforce the following additional set of constraints (Baker and Keller, 2010):

$$T_k \geq \sum_{j\in N} p_j \sum_{i=1}^{k} x_{ji} - \sum_{j\in N} d_j x_{jk}, \qquad \forall k \in N, \tag{5}$$

$$T_k \geq 0, \qquad \forall k \in N. \tag{6}$$

Along with the objective, the constraints (5)-(6) collectively mandate that $T_k = \max(0, C_j - d_j)$ if job $j$ is in position $k$ in the sequence, where $C_j$ represents the completion time of job $j$.

Modeling the weighted versions $TWCT$ and $TWT$ of these performance measures is more complicated because in this case we need to know explicitly the job assigned to position $k$ so that we can match the job-dependent priorities to the correct completion times and tardiness values. To this end, we designate the completion time of the $k$th job in sequence by $\gamma_k$ and append the following set of constraints (Keha et al., 2009) to $(\mathbf{G})$ in order to compute the completion times $C_j$ for all $j \in N$:

$$\gamma_1 \geq \sum_{j\in N} p_j x_{j1}, \tag{7}$$

$$\gamma_k \geq \gamma_{k-1} + \sum_{j\in N} p_j x_{jk}, \qquad k = 2,...,n, \tag{8}$$

$$\gamma_k \geq 0, \qquad \forall k \in N, \tag{9}$$

$$C_j \geq \gamma_k - M(1 - x_{jk}) \qquad \forall j,k \in N, \tag{10}$$

$$C_j \geq 0 \qquad \forall j \in N. \tag{11}$$

For minimizing $TWCT$, it then suffices to define $f(\boldsymbol{x}) = \sum_{j\in N} w_j C_j$. However, for minimizing $TWT$ the following set of constraints are required in addition to (7)-(11) so that the tardiness variables assume their correct values, where the tardiness $T_j$ of job $j$ is expressed as $T_j = \max(0, C_j - d_j)$.

$$T_j \geq C_j - d_j, \qquad \forall j \in N, \tag{12}$$

$$T_j \geq 0, \qquad \forall j \in N. \tag{13}$$

The objective is then specified as $f(\boldsymbol{x}) = \sum_{j \in N} w_j T_j$.

Before we proceed to present our generic stochastic programming model to minimize VaR we elaborate on our choice of using the assignment based formulation (**G**). For deterministic single-machine scheduling problems, four frequently used alternate deterministic formulations appear in the literature (see Keha et al., 2009): disjunctive (DF), time-indexed (TIF), linear ordering (LOF), and the assignment and positional date (APDF) formulations. TIF has a tight LP relaxation and is the best contender among these four formulations in terms of solution speed and LP bound quality if the processing times are small. TIF, however, cannot be adapted to our stochastic setting directly, because it infers the sequence from the completion times represented by binary decision variables. Recall that our goal is to find a non-preemptive static job processing sequence at time zero. That is, the decisions are independent of the random realizations of data, and therefore, relying on completion time information that is contingent on the random processing times is not appropriate to construct a static job processing sequence. Our preliminary results indicate that DF is outperformed by LOF and APDF. This observation is also supported by the extensive computational study presented in Keha et al. (2009). Thus, among the common formulations only LOF and APDF are viable options for our proposed risk-averse model. In this study, we focus on APDF as it proves useful in developing effective solution algorithms; however, the proposed modeling framework would apply to LOF in a similar way.

**2.2 Stochastic programming model** In the remainder of this paper, we restrict our attention to scheduling problems with random processing times for ease of exposition and in order to keep the discussion focused. However, we emphasize that the models and solution methods laid out in this study can be extended in a straightforward manner to capture the uncertainty in several problem parameters simultaneously. Moreover, from a practical point of view it is reasonable to presume that a due date is quoted (deterministically) as a result of a mutual agreement with the customer, and the weight assigned to a job is a reflection of the internal priority of the associated customer or a contractual agreement and is known.

In our stochastic setup, the actual values of the processing times are not certain at the time we determine the job processing sequence, and the processing times can be represented by random variables. This implies that the completion times and the performance measure associated with a sequence are also random variables, because they are functions of the random processing times. In this setting, comparing alternate candidate sequences requires comparing their respective random $f(\boldsymbol{x})$ values, where we assume that a feasible sequence is represented as an assignment $\boldsymbol{x} \in \{0,1\}^{n \times n}$ satisfying the constraints (2)-(4). In this paper, we propose a risk-averse approach which evaluates a sequence $\boldsymbol{x}$ with respect to a certain quantile of the distribution of the associated random outcome $f(\boldsymbol{x})$. For instance, the random total tardiness is expressed by

$$f(\boldsymbol{x}) = \sum_{k=1}^{n} \max\left( \sum_{j=1}^{n} \xi_j \sum_{i=1}^{k} x_{ji} - \sum_{j=1}^{n} d_j x_{jk},\ 0 \right) \tag{14}$$

as a function of the decision vector $\boldsymbol{x}$, where $\xi_j$ denotes the random processing time of job $j \in N$. We intend to model the risk associated with the variability of the random outcome $f(\boldsymbol{x})$ by introducing the following probabilistic constraint:

$$P(f(\boldsymbol{x}) \leq \theta) \geq \alpha, \tag{15}$$

where $\alpha$ is a specified large probability such as 0.90 or 0.95. Here $\theta$ denotes an upper bound on $f(\boldsymbol{x})$ that is exceeded with at most a small probability of $1-\alpha$. If $\alpha = 1$, $f(\boldsymbol{x}) \leq \theta$ holds almost surely. As discussed in more depth in Section 1, such a probabilistic constraint is intuitive and allows us to model a service level requirement for the performance measure of interest under the stochastic setup. We refer to $\alpha$ as the risk parameter which reflects the level of risk-aversion of the decision maker. Clearly, increasing $\alpha$ results in allowing a higher value of the upper bound $\theta$. We propose not to specify the value of $\theta$ as an input, but consider it as a decision variable with the purpose of identifying the sequence with the smallest possible value of $\theta$ given the risk aversion of the decision maker. Thus, in our model we minimize $\theta$ for a specified parameter

$\alpha$, which is equivalent to minimizing the $\alpha$-quantile of the random $f(\boldsymbol{x})$. The $\alpha$-quantile has a special name in risk theory as presented in the next definition.

DEFINITION 2.1 *Let $X$ be a random variable with a cumulative distribution function denoted by $F_X$. The $\alpha$-quantile*

$$\inf\{\eta \in \mathbb{R} : \ F_X(\eta) \geq \alpha\}$$

*is called the Value-at-Risk (VaR) at the confidence level $\alpha$ and denoted by $\mathrm{VaR}_\alpha(X)$, $\alpha \in (0, 1]$.*

The probabilistic constraint (15) can equivalently be formulated as a constraint on the VaR of the random $f(\boldsymbol{x})$:

$$\mathrm{VaR}_\alpha(f(\boldsymbol{x})) \leq \theta. \tag{16}$$

In other words, by considering the proposed probabilistic constraint (15) we specify VaR as the risk measure on the random $f(\boldsymbol{x})$, and minimizing $\theta$ corresponds to seeking the sequence with the smallest possible VaR measure for a specified $\alpha$ value.

A model with a probabilistic constraint similar to that in (15) with randomness on the left hand side was first studied by van de Panne and Popp (1963) and Kataoka (1963). Kataoka introduces a transportation type model and van de Panne and Popp present a diet (cattle feed) optimization model with a single probabilistic constraint. In both studies, all the decision variables are continuous, the random outcome of interest is a linear function of the decision vector, and the solution methods are specific to random coefficients with a joint normal distribution. In contrast, our main decision variables are binary, the random outcome $f(\boldsymbol{x})$ in our work is not necessarily a linear function of the decision vector as evident from (14), and we do not assume that the random parameters have a special distribution.

We characterize the random processing times by a finite set of scenarios denoted by $S$, where a scenario represents a joint realization of the processing times of all jobs and $\pi^s$ denotes the probability of scenario $s \in S$. To develop our stochastic programming formulation, the previously introduced parameters and variables are augmented with scenario indices. More specifically, we use $p_j^s$, $C_j^s$, and $T_j^s$ to denote the processing time, the completion time, and the tardiness of job $j$ under scenario $s \in S$, respectively. For a feasible job processing sequence $\boldsymbol{x}$, the realization of the random performance criterion $f(\boldsymbol{x})$ under scenario $s$ is represented by $f^s(\boldsymbol{x})$. Then, we formulate the problem of minimizing the VaR of a generic random performance criterion in the single-machine environment as follows:

$$(\mathbf{G-VaR}) \qquad \text{minimize} \quad \theta \tag{17}$$

$$\text{subject to} \quad (2)-(4),$$

$$f^s(\boldsymbol{x}) - \theta \leq f^s_{\max}\beta^s, \qquad \forall s \in S, \tag{18}$$

$$\sum_{s\in S} \pi^s \beta^s \leq 1-\alpha, \tag{19}$$

$$\beta^s \in \{0,1\}, \qquad \forall s \in S, \tag{20}$$

$$\theta_{LB} \leq \theta \leq \theta_{UB}. \tag{21}$$

We emphasize that the constraints (2)-(4) in the generic deterministic formulation (**G**) are directly incorporated into the generic risk-averse stochastic programming model (**G − VaR**). That is, the sequencing decisions are independent of the uncertainty. The parameter $f^s_{\max}$ is no smaller than the maximum possible realization of the random performance measure $f(\boldsymbol{x})$ under scenario $s$ for any sequence $\boldsymbol{x}$. Thus, $\beta^s$ is set to 1 by the corresponding constraint (18) without violating feasibility if the realization of the random performance measure $f(\boldsymbol{x})$ under scenario $s$ exceeds the threshold value $\theta$. Constraint (19) prescribes that the probability of exceeding the threshold value $\theta$ is no more than $1-\alpha$ for the random performance measure $f(\boldsymbol{x})$. Finally, constraint (21) is enforced to tighten the formulation and plays an instrumental role in improving the quality of the lower bound on the optimal objective value of (**G − VaR**) attained by our solution method. We further

discuss this issue in Section 3.5. Clearly, $\theta_{LB}$ and $\theta_{UB}$ might trivially be set to zero and infinity, respectively, and observe that the VaR associated with any feasible sequence yields a valid upper bound on the optimal value of $\theta$.

In the formulation above, $f^s(\boldsymbol{x})$ represents the optimal objective value of the optimal timing problem solved over the data pertinent to scenario $s$ under a fixed job processing sequence $\boldsymbol{x}$. With this point of view, $(\mathbf{G-VaR})$ is a two-stage stochastic program, where the optimal timing problem corresponds to the second-stage problem. However, for regular scheduling performance measures the solution of the optimal timing problem is obtained trivially by scheduling jobs consecutively without idle time in between. Consequently, the completion times may be expressed in closed form for any feasible $\boldsymbol{x}$, and for this case we can also view the proposed model as a single-stage stochastic program.

Following a rationale similar to that in Section 2.1 for the deterministic case, we can explicitly reflect the timing aspect of the problem in the model in order to construct a monolithic model that minimizes VaR. For any specific performance measure, we would then need to incorporate constraints and variables in the formulation above to ensure that the value of $f^s(\boldsymbol{x})$ is computed correctly for any $s \in S$ and its relationship to $\theta$ and $\beta^s$ is properly established. For instance, the set of constraints below – derived from constraints (7)-(13) – accomplishes this task for a given scenario $s$ under the performance measure $TWT$, where the completion time of the $k$th job in the sequence under scenario $s$ is denoted by $\gamma_k^s$. These constraints are replicated for each scenario $s \in S$ and replace (18) for a valid formulation that minimizes VaR under the performance measure $TWT$.

$$\gamma_1^s \geq \sum_{j\in N} p_j^s x_{j1}, \tag{22}$$

$$\gamma_k^s \geq \gamma_{k-1}^s + \sum_{j\in N} p_j^s x_{jk}, \qquad k = 2, ..., n, \tag{23}$$

$$\gamma_k^s \geq 0, \qquad \forall k \in N, \tag{24}$$

$$C_j^s \geq \gamma_k^s - M(1 - x_{jk}) \qquad \forall j, k \in N, \tag{25}$$

$$C_j^s \geq 0 \qquad \forall j \in N, \tag{26}$$

$$T_j^s \geq C_j^s - d_j, \qquad \forall j \in N, \tag{27}$$

$$T_j^s \geq 0, \qquad \forall j \in N, \tag{28}$$

$$\sum_{j\in N} w_j T_j^s - \theta \leq f_{\max}^s \beta^s. \tag{29}$$

To complete the formulation for $TWT$, we must also determine the upper bounds $f_{\max}^s, \forall s \in S$. In order to compute a reasonably small value for $f_{\max}^s$ given a scenario $s$, we sort the processing times under scenario $s$ in non-increasing order and denote the $j$th largest processing time by $p_{[j]}^s$. Then, the maximum possible completion time of the $k$th job in the sequence, $k = 1, \ldots, n$, under scenario $s$ is computed as $C_{[k]}^s = \sum_{j=1}^k p_{[j]}^s$. Next, the due dates and the unit tardiness weights are assigned to the completion times in non-increasing and non-decreasing order, respectively. A standard pairwise interchange argument (not necessarily adjacent) demonstrates that the resulting $TWT$ is an upper bound on the $TWT$ of any job processing sequence under scenario $s$.

At the end of Section 2.1, we pointed out that the formulations APDF and LOF are the only reasonable choices in our stochastic programming setup. We focus on the former because representing a job processing sequence as an assignment leads to some powerful combinatorial solution algorithms in Section 3 for solving a relaxation of our risk-averse model. However, for benchmarking purposes we also solve the monolithic formulation $(\mathbf{G-VaR})$ customized for two performance measures by an off-the-shelf mixed-integer solver in Section 5, and representing a sequence as a linear order described by the constraints (30)-(32) instead of an assignment given by (2)-(4) turns out to be computationally more effective in certain cases (see Section 5.3).

$$\delta_{jj} = 1, \qquad \forall j \in N, \tag{30}$$

$$\delta_{jk} + \delta_{kj} = 1, \qquad \forall j,\ k \in N\ :\quad 1 \leq j < k \leq n, \tag{31}$$

$$\delta_{jk} + \delta_{kl} + \delta_{lj} \leq 2, \qquad \forall j,\ k,\ l \in N\ :\quad j \neq k,\ k \neq l,\ l \neq j. \tag{32}$$

The variable $\delta_{jk}$ takes the value 1 if job $j$ precedes job $k$ in the processing sequence. The constraints (31) ensure that job $j$ cannot precede and succeed job $k$ in a feasible sequence, while constraints (32) prevent cyclic subsequences of length three. Finally, by convention a job precedes itself, and $\delta_{jj}$ is set to 1 for all $j$. Note that $C_j = \sum_{k \in N} p_k \delta_{kj}$ yields the completion time of job $j$, and constraints similar to (22)-(29) can be expressed based on this relationship.

We conclude this section with a brief discussion on the general applicability of our stochastic modeling and solution framework. One of the main points that has to come across in this section is that the separation of the sequencing and timing aspects of scheduling enables us to provide a generic optimization framework to minimize VaR in the next section based on enumerating a potentially large set of job processing sequences in a certain order. From a theoretical point of view, the basic requirement specific to the performance measure of interest is that the associated optimal timing problem can be solved effectively for a given job processing sequence. That is, non-regular performance measures such as total earliness/tardiness and non-linear performance measures such as total squared tardiness are within the boundaries of our modeling and solution framework. From a computational point of of view, however, the total effort expended may wildly differ from one performance measure to another – see Section 5. Moreover, as part of our solution method we sometimes have to resort to solving linear mixed-integer programs in combination with our enumerative algorithms. Then, the ability to express $f(\boldsymbol{x})$ with a set of linear constraints is of course crucial for computational effectiveness. Ultimately, we employ $TT$ and $TWT$ as the performance measures of interest in our experimental study in Section 5. These are two of the well-studied classical regular performance measures in deterministic scheduling and are known to give rise to practically and theoretically challenging problems. Note that the deterministic problems of minimizing $TT$ and $TWT$ on a single machine are both $\mathcal{NP}-$hard (Du and Leung, 1990; Lenstra et al., 1977). These results establish that minimizing VaR under either performance measure is $\mathcal{NP}-$hard as well because the associated deterministic problem is a special case of our risk-averse problem with a single scenario and $\alpha = 1$.

**3. Solution Methods** As discussed in the introduction, several decomposition based solution methods have been offered to solve stochastic programming models with an emphasis on the two-stage stochastic integer programs. However, (**G − VaR**) does not exhibit the well-known decomposable structure of traditional risk-neutral two-stage stochastic programs due to the presence of the coupling constraint (19) that reflects the fundamental structure of VaR. Adapting the Lagrangian relaxation based scenario decomposition approach designed by Carøe and Schultz (1999) offers a way of tackling this challenging issue as detailed in the sequel.

In this section, we first reformulate (**G − VaR**) through variable splitting – initially due to Jörnsten et al. (1985) – so that it is amenable to scenario decomposition through Lagrangian relaxation. Then, we focus on the Lagrangian subproblems in Section 3.2 which turn out to be the computationally most demanding component of our solution framework. The job processing sequences obtained from the Lagrangian subproblems are excellent seeds for constructing good primal feasible solutions for the original problem, as explained briefly in Section 3.3. The cut generation algorithm to solve the Lagrangian dual problem is presented in Section 3.4; however, we relegate the dual stabilization to Section 4, where we discuss numerous computational features and enhancements, so as not to detract the attention from the fundamentals. The dynamic nature of our scenario decomposition algorithm is introduced in Section 3.5, where we explain how to exploit the structure of the problem to obtain and solve progressively tighter Lagrangian relaxations of (**G − VaR**). We collectively refer to this solution framework as *SD-SMS* which stands for *Scenario Decomposition for Single-Machine Scheduling*.

**3.1 Scenario Decomposition** We begin by reformulating (**G − VaR**) through variable splitting. Essentially, we first create copies of $\theta$ and $x_{jk}$, $\forall j,k \in N$, for each scenario by defining $\theta^s$, $\forall s \in S$, and $x^s_{jk}$, $\forall j,k \in N$, $\forall s \in S$, and then ensure that all copies of a variable assume the same value by augmenting the

formulation with new constraints. Accordingly, $\theta$ is replaced by $\theta^s$ in constraints (18) and (21), the constraints (2)-(4) and (21) are replicated for each scenario, and the following non-anticipativity constraints are enforced:

$$\theta^1 = \theta^s, \qquad \forall s \in S, s \neq 1, \tag{33}$$

$$(1-\pi^1)x^1_{jk} = \sum_{s=2}^{|S|} \pi^s x^s_{jk}, \qquad \forall j,k \in N. \tag{34}$$

VaR is set to the same value for all scenarios through the constraints (33). Non-anticipativity constraints may be expressed in different ways (Shapiro et al., 2009); we opt for using (33) for $\theta$ which is implemented as the default option in the NEOS solver DDSIP for solving two-stage stochastic linear programs with mixed-integer recourse. In general, $\mathcal{O}(|S|)$ non-anticipativity constraints are required for a single variable; however, a binary variable allows for a compact representation with a single constraint (Carøe and Schultz, 1999) as in (34). These constraints mandate that the static job processing sequence determined at the time zero is identical for all scenarios, and are only valid because all scenario probabilities are strictly positive and the variables $x^s_{jk}$, $\forall j,k \in N, \forall s \in S$, are binary. Moreover, in order to attain a formulation with a decomposable structure the objective term $\theta$ in (17) is replaced by the equivalent expression $\sum_{s\in S} \pi^s\theta^s$ based on (33) and $\sum_{s\in S} \pi^s = 1$. Similarly, the term $(1-\alpha)$ on the right hand side of (19) is substituted by $\sum_{s\in S} \pi^s(1-\alpha)$. The resulting model presented below is equivalent to (**G** − **VaR**).

$$\text{minimize} \quad \sum_{s\in S} \pi^s\theta^s \tag{35}$$

$$\text{subject to} \quad \sum_{k\in N} x^s_{jk} = 1, \qquad \forall j \in N, s \in S, \tag{36}$$

$$\sum_{j\in N} x^s_{jk} = 1, \qquad \forall k \in N, s \in S, \tag{37}$$

$$f^s(\boldsymbol{x}) - \theta^s \leq f^s_{\max}\beta^s, \qquad \forall s \in S, \tag{38}$$

$$\sum_{s\in S} \pi^s\beta^s \leq \sum_{s\in S} \pi^s(1-\alpha), \tag{39}$$

$$\beta^s \in \{0,1\}, \qquad \forall s \in S, \tag{40}$$

$$x^s_{jk} \in \{0,1\}, \qquad \forall j,k \in N, s \in S, \tag{41}$$

$$\theta_{LB} \leq \theta^s \leq \theta_{UB}, \qquad \forall s \in S, \tag{42}$$

$$(1-\pi^1)x^1_{jk} = \sum_{s=2}^{|S|} \pi^s x^s_{jk}, \qquad \forall j,k \in N, \tag{43}$$

$$\theta^1 = \theta^s, \qquad \forall s \in S, s \neq 1. \tag{44}$$

In this formulation, the scenarios are linked together by the non-anticipativity constraints (43)-(44), and the constraint (39). Consequently, we obtain a Lagrangian relaxation of (35)-(44) which decomposes by scenario if we dualize the constraint (39) by a non-negative dual multiplier $\lambda$, and the constraints (43) and (44) by unrestricted dual multipliers $u_{jk}$, $\forall j,k \in N$, and $\mu^s$, $s = 2,\ldots,|S|$, respectively. The *Lagrangian function* $L(\lambda, \boldsymbol{\mu}, \boldsymbol{u})$ is then stated as a sum of the Lagrangian functions for the individual scenarios:

$$L(\lambda, \boldsymbol{\mu}, \boldsymbol{u}) = \sum_{s\in S} L^s(\lambda, \mu^s, \boldsymbol{u}) \tag{45}$$

$$= \sum_{s\in S} \left( (\pi^s + \mu^s)\theta^s + \lambda\pi^s(\beta^s - 1 + \alpha) + \sum_{j\in N}\sum_{k\in N} u_{jk}\boldsymbol{H}^s x^s_{jk} \right), \tag{46}$$

where

$$\boldsymbol{\mu} = \begin{bmatrix} \mu^1 & \mu^2 & \mu^3 & \cdots & \mu^{|S|} \end{bmatrix}^\intercal,$$

$$\mu^1 = -\sum_{s=2}^{|S|} \mu^s, \tag{47}$$

$$\boldsymbol{u} = \begin{bmatrix} u_{11} & u_{12} & \cdots & u_{1n} & u_{21} & \cdots & u_{nn} \end{bmatrix}^{\intercal},$$
$$\boldsymbol{H} = \begin{bmatrix} (\pi^1 - 1) & \pi^2 & \pi^3 & \cdots & \pi^{|S|} \end{bmatrix}^{\intercal}, \text{ and}$$

$\boldsymbol{H}^s$ represents the $s$th component of $\boldsymbol{H}$. The analysis above provides us with the *Lagrangian dual* problem

$$(\mathbf{LD}) \qquad z_{LD} = \underset{\lambda \geq 0, \boldsymbol{\mu}, \boldsymbol{u}}{\text{maximize}} \quad D(\lambda, \boldsymbol{\mu}, \boldsymbol{u}) = \underset{\lambda \geq 0, \boldsymbol{\mu}, \boldsymbol{u}}{\text{maximize}} \quad \sum_{s \in S} D^s(\lambda, \mu^s, \boldsymbol{u}), \tag{48}$$

where $D(\lambda, \boldsymbol{\mu}, \boldsymbol{u})$ is the *dual function* and the *Lagrangian subproblem* $(\mathbf{LS}^s)$ for scenario $s$ is expressed as:

$$(\mathbf{LS}^s) \qquad D^s(\lambda, \mu^s, \boldsymbol{u}) = \text{minimize} \quad L^s(\lambda, \mu^s, \boldsymbol{u}) \tag{49}$$
$$\text{subject to} \quad \sum_{k \in N} x^s_{jk} = 1, \qquad \forall j \in N, \tag{50}$$
$$\sum_{j \in N} x^s_{jk} = 1, \qquad \forall k \in N, \tag{51}$$
$$f^s(\boldsymbol{x}) - \theta^s \leq f^s_{\max} \beta^s, \tag{52}$$
$$\beta^s \in \{0, 1\}, \tag{53}$$
$$x^s_{jk} \in \{0, 1\}, \qquad \forall j, k \in N, \tag{54}$$
$$\theta_{LB} \leq \theta^s \leq \theta_{UB}. \tag{55}$$

From the theory of Lagrangian relaxation, the value of the dual function for any $\lambda \geq 0$, $\boldsymbol{\mu}$, and $\boldsymbol{u}$ is a lower bound on the optimal objective value of $(\mathbf{G} - \mathbf{VaR})$. However, this lower bound is trivial if $\mu^s + \pi^s < 0$ for some scenario $s$. In this case, $L^s(\lambda, \mu^s, \boldsymbol{u})$ tends to negative infinity because the value of $\theta^s$ may be increased arbitrarily without violating feasibility, assuming that $\theta_{UB} = \infty$ for the sake of argument. Therefore, we assume that $\mu^s + \pi^s \geq 0$ holds $\forall s \in S$ from now on. This requirement is also incorporated into our search for the optimal set of dual multipliers in Sections 3.4 and 3.5, where we focus on maximizing the Lagrangian lower bound. In the sequel, we first devise an effective solution strategy for solving $(\mathbf{LS}^s)$.

**3.2 Solving the Lagrangian Subproblems** Solving the mixed-integer formulation (49)-(55) for each scenario at every iteration of the Lagrangian algorithm is clearly not a viable option from a computational point of view. Therefore, the sheer goal of the proposed solution framework for the Lagrangian subproblems is to obtain the optimal solution through fast combinatorial algorithms if possible and then, if necessary, turn to solving $(\mathbf{LS}^s)$ by an off-the-shelf solver as a last resort. We are driven by two fundamental observations in this endeavor. First, each subproblem features a single binary variable $\beta^s$, and we can attain the optimal solution to $(\mathbf{LS}^s)$ by analyzing the dichotomy that results from fixing $\beta^s = 0$ or $\beta^s = 1$. Essentially, the two resulting branches $\left(\mathbf{LS}^s_{\beta^s=0}\right)$ and $\left(\mathbf{LS}^s_{\beta^s=1}\right)$ can be solved separately, and the branch with the smaller objective value determines the optimal solution to $(\mathbf{LS}^s)$:

$$\begin{array}{llll} \left(\mathbf{LS}^s_{\beta^s=0}\right) & \text{minimize} & L^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 0) & \\ & & (50) - (51) & \\ & & (54) - (55) & \\ & & f^s(\boldsymbol{x}) - \theta^s \leq 0 & \end{array} \qquad \begin{array}{llll} \left(\mathbf{LS}^s_{\beta^s=1}\right) & \text{minimize} & L^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 1) & \\ & & (50) - (51) & \\ & & (54) & \end{array} \quad , \tag{56}$$

where

$$L^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 0) = \sum_{j \in N} \sum_{k \in N} u_{jk} \boldsymbol{H}^s x^s_{jk} + (\mu^s + \pi^s)\theta^s + \lambda(\alpha - 1)\pi^s \text{ and} \tag{57}$$
$$L^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 1) = \sum_{j \in N} \sum_{k \in N} u_{jk} \boldsymbol{H}^s x^s_{jk} + (\mu^s + \pi^s)\theta_{LB} + \lambda\alpha\pi^s \tag{58}$$

express the objective function of the Lagrangian subproblem under the restriction that $\beta^s$ is forced to take the value zero or one, respectively. Observe that the constraints (52) and (55) are redundant for $\beta^s = 1$, and the best strategy in this case is to set $\theta^s$ to $\theta_{LB}$. This is taken directly into account in the definition of

$\left(\mathbf{LS}^s_{\beta^s=1}\right)$. Consistent with the notation above, $D^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 0)$ and $D^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 1)$ represent the optimal objective function values of $\left(\mathbf{LS}^s_{\beta^s=0}\right)$ and $\left(\mathbf{LS}^s_{\beta^s=1}\right)$, respectively.

The final term in (57) and the last two terms in (58) are constant terms. Consequently, $\left(\mathbf{LS}^s_{\beta^s=1}\right)$ is a classical assignment problem, and $\left(\mathbf{LS}^s_{\beta^s=0}\right)$ is basically a bi-objective optimization problem. There is a trade-off between the cost of assigning the jobs to the positions in the sequence represented by the first term in (57) and the cost of the performance measure associated with the sequence under scenario $s$. The latter cost is expressed by the term $(\mu^s + \pi^s)\theta^s$ in (57) because the structure of $\left(\mathbf{LS}^s_{\beta^s=0}\right)$ imposes that $\theta^s$ is set to $f^s(\boldsymbol{x})$ at optimality as long as this is feasible with respect to (55). This trade-off between the direct cost of the job assignments and the cost of the resulting performance measure is the second fundamental observation that we exploit thoroughly in the design of an effective solution algorithm for $(\mathbf{LS}^s)$.

With the discussion above in mind, we solve $(\mathbf{LS}^s)$ by a three-phase approach. In the first phase, we investigate various special cases of $(\mathbf{LS}^s)$ that are relatively easy to tackle. Here, we also resolve how to detect whether $\left(\mathbf{LS}^s_{\beta^s=0}\right)$ is infeasible; the assignment problem $\left(\mathbf{LS}^s_{\beta^s=1}\right)$ is always feasible. If we fail to identify the optimal solution in the first phase, we proceed with a combinatorial algorithm based on enumerating and evaluating a subset of all job processing sequences. In theory, this algorithm is optimal; however, the number of sequences to be enumerated may be exponentially large. Therefore, from a practical point of view we only consider a limited number of sequences for computational performance. Finally, if $(\mathbf{LS}^s)$ is not solved to optimality after the initial two phases, we invoke an off-the-shelf solver on the formulation (49)-(55). We detail these steps in the remainder of this section.

The first task at hand in Phase I is to determine the feasibility of $\left(\mathbf{LS}^s_{\beta^s=0}\right)$ because there may be no job processing sequence $\boldsymbol{x}$ such that the constraint $f^s(\boldsymbol{x}) - \theta^s \leq 0$ is satisfied when $\theta_{LB} \leq \theta^s \leq \theta_{UB}$. This check is accomplished by identifying the minimum possible value of the performance measure under scenario $s$; that is, by solving a *deterministic single-machine scheduling problem* with the data pertinent to scenario $s$. We denote the resulting optimal solution and objective function value by $\boldsymbol{x}^s_{\min}$ and $f^s_{\min}$, respectively, and conclude that $\left(\mathbf{LS}^s_{\beta^s=0}\right)$ is feasible if and only if $f^s_{\min} \leq \theta_{UB}$. In the absence of an optimal algorithm for the performance measure of interest, a lower bound $f^s_{LB}$ from a relaxation of the deterministic problem associated with scenario $s$ may also be used to a less powerful effect. In this case, we arrive at the conclusion that $\left(\mathbf{LS}^s_{\beta^s=0}\right)$ is infeasible if $f^s_{LB} > \theta_{UB}$; however, $f^s_{LB} \leq \theta_{UB}$ does not necessarily imply the feasibility of $\left(\mathbf{LS}^s_{\beta^s=0}\right)$. If we determine that $\left(\mathbf{LS}^s_{\beta^s=0}\right)$ is infeasible, then the optimal solution to $(\mathbf{LS}^s)$ is identified as $\beta^s_* = 1$, $\theta^s_* = \theta_{LB}$, and $\boldsymbol{x}^s_* = \boldsymbol{x}^1_{AP}$ with an objective value of

$$D^s(\lambda, \mu^s, \boldsymbol{u}) = D^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 1) = z^1_{AP} + (\mu^s + \pi^s)\theta_{LB} + \lambda\alpha\pi^s, \tag{59}$$

where $\boldsymbol{x}^1_{AP}$ and $z^1_{AP}$ denote the optimal solution and the associated objective value of the assignment problem that minimizes the first term of (58) over the feasible region of $\left(\mathbf{LS}^s_{\beta^s=1}\right)$, respectively. This optimization may be carried out by any standard assignment algorithm, such as the famous Hungarian algorithm. Note that the motivation for employing a superscript of '1' in $\boldsymbol{x}^1_{AP}$ and $z^1_{AP}$ is clarified later in this section.

The feasibility check explained in the previous paragraph requires us to solve $|S|$ deterministic single-machine scheduling problems for the performance measure of interest. This immediately establishes that $(\mathbf{LS}^s)$ is $\mathcal{NP}-$hard for the performance measures $TT$ and $TWT$ considered in this paper because the deterministic single-machine $TT$ and $TWT$ problems are both $\mathcal{NP}-$hard as mentioned in Section 2.2. From an implementation point of view, the $|S|$ deterministic problems are clearly only solved once during the initialization step of the algorithm for solving the Lagrangian dual $(\mathbf{LD})$ (see Section 4.1). In subsequent iterations of the algorithm, we only compare $f^s_{\min}$ against the current best upper bound on $\theta$ which may decrease over the iterations. For ease of exposition, the remaining discussion in this section assumes that $\left(\mathbf{LS}^s_{\beta^s=0}\right)$ is feasible.

The solution of $(\mathbf{LS}^s)$ deserves special attention if we have $\boldsymbol{u} = \mathbf{0}$. Our preliminary computational experience with $(\mathbf{LD})$ revealed a critical observation. Many components of the optimal $\boldsymbol{u}$ are frequently identical to zero, but the variability in $\boldsymbol{u}$ from one iteration to the next is a major source of instability in the dual problem

and slows down the convergence. To alleviate this issue, we apply a dual stabilization scheme, which initially imposes the restriction $\boldsymbol{u} = \mathbf{0}$ in (**LD**) (see Section 4.3 for more details). The solution of ($\mathbf{LS}^s$) turns out to be particularly simple for this case, and the optimal solution of the deterministic problem associated with scenario $s$ plays a crucial role here as well. For $\boldsymbol{u} = \mathbf{0}$, the objective value of $\left(\mathbf{LS}^s_{\beta^s=1}\right)$ is a constant independent of the sequence. We have $D^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 1) = (\mu^s + \pi^s)\theta_{LB} + \lambda\alpha\pi^s$, $\theta^s_* = \theta_{LB}$, and set $\boldsymbol{x}^s_* = \boldsymbol{x}^s_{\min}$ by convention. On the other hand, the constraint $f^s(\boldsymbol{x}) - \theta^s \leq 0$ and the second objective term in (57) prescribe that we pick the sequence that minimizes the performance measure under scenario $s$ for $\left(\mathbf{LS}^s_{\beta^s=0}\right)$, also taking into account (55). Consequently, the optimal solution of $\left(\mathbf{LS}^s_{\beta^s=0}\right)$ is attained at $\theta^s_* = \max(\theta_{LB}, f^s_{\min})$, $\boldsymbol{x}^s_* = \boldsymbol{x}^s_{\min}$ and results in the objective value $D^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 0) = (\mu^s + \pi^s)\max(\theta_{LB}, f^s_{\min}) + \lambda(\alpha - 1)\pi^s$.

In the next special case considered in the first phase, we allow for $\boldsymbol{u} \neq \mathbf{0}$ and observe that if $f^s(\boldsymbol{x}^1_{AP}) \leq \theta_{UB}$ and $(\mu^s + \pi^s)(f^s(\boldsymbol{x}^1_{AP}) - \theta_{LB}) \leq 0$ then $\boldsymbol{x}^1_{AP}$ is optimal for $\left(\mathbf{LS}^s_{\beta^s=0}\right)$ and ($\mathbf{LS}^s$). The former condition ensures that (55) is not violated, and the latter prescribes that the cost of the performance measure associated with the job sequence with the minimum assignment cost is no more than $(\mu^s + \pi^s)\theta_{LB}$. Thus, $D^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 0) = z^1_{AP} + (\mu^s + \pi^s)\theta_{LB} + \lambda(\alpha - 1)\pi^s < z^1_{AP} + (\mu^s + \pi^s)\theta_{LB} + \lambda\alpha\pi^s = D^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 1)$ and the optimal solution of ($\mathbf{LS}^s$) is stated as $\beta^s_* = 0$, $\theta^s_* = \max(\theta_{LB}, f^s(\boldsymbol{x}^1_{AP}))$, and $\boldsymbol{x}^s_* = \boldsymbol{x}^1_{AP}$. This rule benefits from the progressively larger lower bounds on $\theta$ obtained during the course of the algorithm and turns more powerful over the iterations.

The final step of the first phase is to check whether the assignment cost incurred by $\boldsymbol{x}^s_{\min}$ – computed by the first term in (57) or (58) – is identical to $z^1_{AP}$. If this holds, then we set $\boldsymbol{x}^s_* = \boldsymbol{x}^s_{\min}$, compare $D^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 0) = z^1_{AP} + (\mu^s + \pi^s)\max(\theta_{LB}, f^s_{\min}) + \lambda(\alpha - 1)\pi^s$ to $D^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 1) = z^1_{AP} + (\mu^s + \pi^s)\theta_{LB} + \lambda\alpha\pi^s$, and then set $\beta^s_*$ and $\theta^s_*$ as appropriate. The motivation for this check is rooted in our observation that the problem of minimizing the first term of (57) or (58) subject to (50), (51), (54) does frequently feature multiple optima.

In Phase II of our solution approach for ($\mathbf{LS}^s$), we exploit two dominance relations to devise an optimal combinatorial algorithm. The key point to both of these conditions derives from the expression for the optimal objective value of $\left(\mathbf{LS}^s_{\beta^s=0}\right)$ under a feasible fixed job processing sequence $\boldsymbol{x}^s$:

$$D^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 0, \boldsymbol{x}^s) = z_{AP}(\boldsymbol{x}^s) + (\mu^s + \pi^s)\max(\theta_{LB}, f^s(\boldsymbol{x}^s)) + \lambda(\alpha - 1)\pi^s, \text{ where} \tag{60}$$

$$z_{AP}(\boldsymbol{x}^s) = \sum_{j \in N}\sum_{k \in N} u_{jk}\boldsymbol{H}^s x^s_{jk} \tag{61}$$

is the value of the first term in (57) corresponding to $\boldsymbol{x}^s$.

LEMMA 3.1 *If $\beta^s_* = 0$ in the optimal solution of ($\mathbf{LS}^s$), then*

$$z_{AP}(\boldsymbol{x}^s_*) \leq z^1_{AP} + \lambda\pi^s \tag{62}$$

*must hold for the corresponding optimal job processing sequence $\boldsymbol{x}^s_*$.*

PROOF. For a given fixed sequence $\boldsymbol{x}^s$ we have

$$\begin{aligned} & D^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 1) - D^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 0, \boldsymbol{x}^s) \geq 0 \\ \iff & (z^1_{AP} - z_{AP}(\boldsymbol{x}^s)) + (\mu^s + \pi^s)(\theta_{LB} - \max(\theta_{LB}, f^s(\boldsymbol{x}^s))) + \lambda\pi^s \geq 0 \\ \iff & z_{AP}(\boldsymbol{x}^s) - z^1_{AP} \leq (\mu^s + \pi^s)(\theta_{LB} - \max(\theta_{LB}, f^s(\boldsymbol{x}^s))) + \lambda\pi^s. \end{aligned} \tag{63}$$

In other words, if the direct cost of the job assignments associated with $\boldsymbol{x}^s$ exceeds the minimum possible assignment cost $z^1_{AP}$ by more than the right hand side of (63), then we set $\beta^s_* = 1$ instead of setting $\beta^s_* = 0$ and choosing the sequence $\boldsymbol{x}^s$. We can extend this argument by noting that $(\mu^s + \pi^s)(\theta_{LB} - \max(\theta_{LB}, f^s(\boldsymbol{x}^s))) \leq 0$ and arrive at the conclusion formalized in the statement of the lemma. □

Lemma 3.1 discards a set of inferior sequences from consideration in $\left(\mathbf{LS}^s_{\beta^s=0}\right)$ because one is better off by setting $\beta^s = 1$ instead. The next lemma is based on similar concepts and identifies a dominant set of

job processing sequences which must include the optimal solution for $(\mathbf{LS}^s_{\beta^s=0})$. In either case, the focus is on $(\mathbf{LS}^s_{\beta^s=0})$ because computing the optimal solution of $(\mathbf{LS}^s_{\beta^s=1})$ requires no more than solving a classical assignment problem. In the following, a set of feasible job processing sequences $\boldsymbol{x}^1_{AP}, \boldsymbol{x}^2_{AP}, \ldots, \boldsymbol{x}^k_{AP}, \ldots$ are ordered so that $z_{AP}(\boldsymbol{x}^1_{AP}) \leq z_{AP}(\boldsymbol{x}^2_{AP}) \leq \cdots \leq z_{AP}(\boldsymbol{x}^k_{AP}) \leq \cdots$. For brevity of notation, we use $z^k_{AP}$ for $z_{AP}(\boldsymbol{x}^k_{AP})$.

LEMMA 3.2 *The optimal job processing sequence for* $(\mathbf{LS}^s_{\beta^s=0})$ *must belong to the set of the first* $K^s_2$ *least cost assignments* $\boldsymbol{x}^1_{AP}, \ldots, \boldsymbol{x}^{K^s_2}_{AP}$, *where* $K^s_2 = \arg\min\{k \in \mathbb{Z}^+ \mid f^s(\boldsymbol{x}^k_{AP}) \leq \theta_{UB}$ *and* $(\mu^s+\pi^s)(f^s(\boldsymbol{x}^k_{AP})-\theta_{LB}) \leq 0\}$.

PROOF. The sequence $z^{K^s_2}_{AP}$ is feasible with respect to (55) and the cost of the performance measure associated with it does not exceed $(\mu^s + \pi^s)\theta_{LB}$. Therefore, for any job processing sequence $\boldsymbol{x}^s$ such that $z_{AP}(\boldsymbol{x}^s) > z^{K^s_2}_{AP}$, we have $D^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 0, \boldsymbol{x}^s) = z_{AP}(\boldsymbol{x}^s) + (\mu^s + \pi^s)\max(\theta_{LB}, f^s(\boldsymbol{x}^s)) + \lambda(\alpha - 1)\pi^s > z^{K^s_2}_{AP} + (\mu^s + \pi^s)\theta_{LB} + \lambda(\alpha - 1)\pi^s = D^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 0, \boldsymbol{x}^{K^s_2}_{AP})$. Thus, the sequence $\boldsymbol{x}^s$ can be excluded from consideration. □

Lemmas 3.1 and 3.2 motivate an algorithm for solving $(\mathbf{LS}^s)$ that hinges upon ranking the job processing sequences based on their assignment costs expressed in (61) for any given $\boldsymbol{x}^s$. We start enumerating assignments $\boldsymbol{x}^1_{AP}, \boldsymbol{x}^2_{AP}, \ldots$, in non-decreasing order of their costs. If $z^{k'}_{AP} \leq z^1_{AP} + \lambda\pi^s$ and $z^{k'+1}_{AP} > z^1_{AP} + \lambda\pi^s$ are satisfied for the $k'$th least cost assignment, then we set $K^s_1 = k'$ and terminate the enumeration by applying Lemma 3.1. We then calculate the minimum value of $D^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 0, \boldsymbol{x}^k_{AP})$ over $\boldsymbol{x}^k_{AP}, k = 1, \ldots, K^s_1$, and compare it to $D^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 1)$ given in (59). We thus solve $(\mathbf{LS}^s)$ optimally. Alternatively, we stop the enumeration at the $k''$th least cost assignment and set $K^s_2 = k''$ according to Lemma 3.2, if $f^s(\boldsymbol{x}^{k''}_{AP}) \leq \theta_{UB}$ and $(\mu^s + \pi^s)(f^s(\boldsymbol{x}^{k''}_{AP}) - \theta_{LB}) \leq 0$. Then, we proceed similarly to obtain the optimal solution of $(\mathbf{LS}^s)$. In practice, Lemma 3.1 is hardly employed, and the enumeration we implement – described below – is almost always brought to completion by Lemma 3.2. Note that this lemma can be regarded as a generalization of the special case considered in the first phase, where $z^1_{AP}$ turns out to be the optimal solution of both $(\mathbf{LS}^s_{\beta^s=0})$ and $(\mathbf{LS}^s)$. In addition, the condition of the lemma gets easier to satisfy as $\theta_{LB}$ is improved over the course of solving $(\mathbf{LD})$.

Several polynomial algorithms of complexity $\mathcal{O}(n^3K)$ are discussed in Section 5.4.1 of Burkard et al. (2009) for identifying the first $K$ least cost assignments for any fixed $K$. Thus, the ultimate computational effectiveness of our proposed strategy for a given scenario $s \in S$ depends on the value of $K^s = \min(K^s_1, K^s_2)$, which cannot be computed a priori but is only revealed whenever either the condition of Lemma 3.1 or that of Lemma 3.2 is satisfied while the assignments are evaluated. On the positive side, we only need to solve the $K-$assignment problem *twice* regardless of the number of scenarios $|S|$. The objective coefficients in (61) are a function of the vector of dual multipliers $\boldsymbol{u}$, which is independent of the scenario $s$, and $\boldsymbol{H}^s$. Note that $\boldsymbol{H}^1 = \pi^1 - 1 < 0$ and $\boldsymbol{H}^s = \pi^s > 0$ for $s \geq 2$. Consequently, for $s = 1$ the objective is to minimize $\sum_{j\in N}\sum_{k\in N} -u_{jk}x^1_{jk}$, and for any $s \geq 2$ the objective is to minimize $\sum_{j\in N}\sum_{k\in N} u_{jk}x_{jk}$. In the latter case, the superscript $s$ on the variables $x_{jk}$ is omitted deliberately because the least cost assignments are not scenario-dependent for $s \geq 2$.

The enumerative algorithm we have just presented provides the optimal solution to $(\mathbf{LS}^s)$ in theory. However, its complexity is not polynomial because the number of assignments (job processing sequences) to be enumerated is unknown at the start of the algorithm and may grow to $n!$ in the worst case. Therefore, for practical purposes we use a scenario-independent fixed upper bound $\kappa$ on the number of assignments to be enumerated. Then, we evaluate $\boldsymbol{x}^1_{AP}, \boldsymbol{x}^2_{AP}, \ldots, \boldsymbol{x}^\kappa_{AP}$ one by one and check whether the conditions of Lemmas 3.1 and 3.2 are satisfied. If it turns out that either $K^s_1 \leq \kappa$ or $K^s_2 \leq \kappa$, then we attain a provably optimal solution of $(\mathbf{LS}^s)$. Otherwise, we only have a suboptimal solution for $(\mathbf{LS}^s)$ at the conclusion of the second phase of our solution algorithm for the Lagrangian subproblems, and this subproblem is relegated to the final phase.

The optimal solutions of a large portion of the subproblems are available when the first two phases described above are completed. If we insist that all Lagrangian subproblems are solved to optimality, then the remaining subproblems are tackled by solving the formulation (49)-(55) through a mixed-integer programming solver. In Section 4, we discuss the circumstances under which we do not necessarily seek optimality for all subproblems and the final phase is skipped. Algorithm 1 summarizes the discussion in this section by presenting a pseudo-code for the three phases of solving ($\mathbf{LS}^s$) to optimality for all $s \in S$ given a set of dual multipliers $\lambda$, $\boldsymbol{\mu}$, and $\boldsymbol{u}$. For ease of presentation, both $K-$assignment problems are solved during the initialization stage in Algorithm 1. However, for an actual implementation it is more efficient to execute the $K-$assignment algorithm within the main `for`-loop that starts on line 2 and only if it is necessary.

We emphasize that Algorithm 1 is widely applicable to different performance measures because our mathematical model clearly differentiates between the sequencing and timing aspects of scheduling. Regardless of the specific performance measure of interest, the sequences are generated by the same $K-$assignment algorithm. Algorithm 1, however, requires that we have a fast optimal timing procedure at our disposal for computing the performance measure because this procedure is called for every sequence produced by the $K-$assignment algorithm. In addition, the existence of an optimal algorithm for the associated deterministic single-machine scheduling problem is essential.

Finally, we revisit and finalize the discussion on our preference of APDF over LOF for our modeling and solution framework. The structure of the Lagrangian subproblems would stay intact under LOF; however, this would rule out an effective method for the ranking of sequences in the second phase of Algorithm 1 because the linear ordering problem is $\mathcal{NP}-$hard (Karp, 1972).

**3.3 Primal Feasible Solutions** By definition, a good relaxation that captures the essence of the original problem provides a tight lower bound on the optimal objective value of the original problem. However, equally important is the information extracted from the solution of the relaxation that paves the way for easily constructing feasible solutions of good quality. To demonstrate that the latter characteristic is present in our scenario decomposition, we keep our primal algorithm very simple.

One of the desirable properties of ($\mathbf{G-VaR}$) is that any job processing sequence is a feasible solution, and the best solutions of the Lagrangian subproblems yield $|S|$ feasible job processing sequences at every iteration of *SD-SMS*. It turns out that good primal feasible solutions for the original problem may be obtained starting from one or several of these sequences. Our primal algorithm consists of computing the VaR associated with these sequences and updating our best solution available. We further extend this idea by evaluating the neighborhood of each subproblem sequence by means of a simple local search algorithm. The neighborhood structure is defined by a general pairwise interchange of the jobs, and we allow up to 5 consecutive non-improving moves to prevent getting stuck at a local optimum.

**3.4 Solving the Lagrangian Dual Problem** The set of candidate solutions in the feasible region $\Phi^s$ of the Lagrangian subproblem ($\mathbf{LS}^s$) consists of exponentially many but a finite set of points $(\boldsymbol{x}_p^s, \beta_p^s, \theta_p^s)$, $p = 1, \ldots, \phi^s$, because there is only a finite number of job processing sequences and $\theta^s$ may be set optimally for any given $\boldsymbol{x}^s$ and $\beta^s$ by following the simple observations in Section 3.2. Based on this representation, the Lagrangian dual problem ($\mathbf{LD}$) specified in (48) in Section 3.1 may be posed in a different form (Wolsey, 1998, Section 10.2):

$$z_{LD} = \underset{\lambda \geq 0, \boldsymbol{\mu}, \boldsymbol{u}}{\text{maximize}} \quad \sum_{s \in S} \left\{ \underset{(\boldsymbol{x}^s, \beta^s, \theta^s) \in \Phi^s}{\text{minimize}} \quad L^s(\lambda, \mu^s, \boldsymbol{u}) \right\} \tag{64}$$

$$= \underset{\lambda \geq 0, \boldsymbol{\mu}, \boldsymbol{u}}{\text{maximize}} \quad \sum_{s \in S} \left\{ \min_{p=1,\ldots,\phi^s} L^s(\lambda, \mu^s, \boldsymbol{u}/(\boldsymbol{x}_p^s, \beta_p^s, \theta_p^s)) \right\}, \text{ where} \tag{65}$$

$L^s(\lambda, \mu^s, \boldsymbol{u}/(\boldsymbol{x}_p^s, \beta_p^s, \theta_p^s))$ represents the value of $L^s(\lambda, \mu^s, \boldsymbol{u})$ evaluated at the point $(\boldsymbol{x}_p^s, \beta_p^s, \theta_p^s)$. This reformulation immediately leads to an LP formulation of ($\mathbf{LD}$), where $\eta^s$ denotes an upper bound on the dual function

**Algorithm 1:** Solving the Lagrangian subproblems.

**input** : Values of the dual variables $\lambda$, $\boldsymbol{\mu}$, and $\boldsymbol{u}$. $\boldsymbol{x}^s_{\min}$ and $f^s_{\min}$ for the deterministic single-machine problem associated with scenario $s$ for all $s \in S$. The bounds $\theta_{LB}$ and $\theta_{UB}$ on $\theta$.

**output**: The optimal solution $\boldsymbol{x}^s_*$, $\beta^s_*$, $\theta^s_*$ and the associated objective value $D^s(\lambda, \mu^s, \boldsymbol{u})$ for all $s \in S$.

Solve a $K-$assignment problem with $K = \kappa$ for $s = 1$ with the objective of minimizing (61) subject to (50), (51), (54). Repeat for $s = 2$. For ease of notation, in either case the assignments retrieved and their associated objective values are labeled as $\boldsymbol{x}^1_{AP}, \boldsymbol{x}^2_{AP}, \ldots, \boldsymbol{x}^\kappa_{AP}$ and $z^1_{AP}, z^2_{AP}, \ldots, z^\kappa_{AP}$, respectively;
**for** $s = 1$ **to** $|S|$ **do**
  **if** $f^s_{\min} \le \theta_{UB}$ **then** // $\left(\mathbf{LS}^s_{\beta^s=0}\right)$ is feasible.
    // Phase I. Consider the special cases.
    **if** $\boldsymbol{u} = \mathbf{0}$ **then**
      **if** $D^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 0) \le D^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 1)$ **then**
        $\beta^s_* = 0$, $\theta^s_* = \max(\theta_{LB}, f^s_{\min})$, $\boldsymbol{x}^s_* = \boldsymbol{x}^s_{\min}$,
        $D^s(\lambda, \mu^s, \boldsymbol{u}) = (\mu^s + \pi^s)\max(\theta_{LB}, f^s_{\min}) + \lambda(\alpha - 1)\pi^s$;
      **else** $\beta^s_* = 1$, $\theta^s_* = \theta_{LB}$, $\boldsymbol{x}^s_* = \boldsymbol{x}^s_{\min}$, $D^s(\lambda, \mu^s, \boldsymbol{u}) = (\mu^s + \pi^s)\theta_{LB} + \lambda\alpha\pi^s$ ;
      Continue with the next scenario;
    **end**
    **if** $f^s(\boldsymbol{x}^1_{AP}) \le \theta_{UB}$ & $(\mu^s + \pi^s)(f^s(\boldsymbol{x}^1_{AP}) - \theta_{LB}) \le 0$ **then** // $\boldsymbol{x}^1_{AP}$ is optimal for $\left(\mathbf{LS}^s_{\beta^s=0}\right)$.
      $\beta^s_* = 0$, $\theta^s_* = \max(\theta_{LB}, f^s(\boldsymbol{x}^1_{AP}))$, $\boldsymbol{x}^s_* = \boldsymbol{x}^1_{AP}$, $D^s(\lambda, \mu^s, \boldsymbol{u}) = z^1_{AP} + (\mu^s + \pi^s)\theta_{LB} + \lambda(\alpha - 1)\pi^s$;
      Continue with the next scenario;
    **end**
    **if** $\boldsymbol{u}^\intercal \boldsymbol{H}^s \boldsymbol{x}^1_{AP} = \boldsymbol{u}^\intercal \boldsymbol{H}^s \boldsymbol{x}^s_{\min}$ **then** // $\boldsymbol{x}^s_{\min}$ is a minimum cost assignment.
      **if** $D^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 0) \le D^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 1)$ **then**
        $\beta^s_* = 0$, $\theta^s_* = \max(\theta_{LB}, f^s(\boldsymbol{x}^s_{\min}))$, $\boldsymbol{x}^s_* = \boldsymbol{x}^s_{\min}$,
        $D^s(\lambda, \mu^s, \boldsymbol{u}) = z^1_{AP} + (\mu^s + \pi^s)\max(\theta_{LB}, f^s_{\min}) + \lambda(\alpha - 1)\pi^s$;
      **else** $\beta^s_* = 1$, $\theta^s_* = \theta_{LB}$, $\boldsymbol{x}^s_* = \boldsymbol{x}^s_{\min}$, $D^s(\lambda, \mu^s, \boldsymbol{u}) = z^1_{AP} + (\mu^s + \pi^s)\theta_{LB} + \lambda\alpha\pi^s$ ;
      Continue with the next scenario;
    **end**
    // Phase II. The optimal solution was not obtained in Phase I.
    $\beta^s_* = 1$, $\theta^s_* = \theta_{LB}$, $\boldsymbol{x}^s_* = \boldsymbol{x}^1_{AP}$, $D^s(\lambda, \mu^s, \boldsymbol{u}) = D^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 1) = z^1_{AP} + (\mu^s + \pi^s)\theta_{LB} + \lambda\alpha\pi^s$;
    **for** $k = 1$ **to** $\kappa$ **do**
      **if** $f^s(\boldsymbol{x}^k_{AP}) \le \theta_{UB}$ & $D^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 0, \boldsymbol{x}^k_{AP}) \le D^s(\lambda, \mu^s, \boldsymbol{u})$ **then** // New incumbent.
        $\beta^s_* = 0$, $\theta^s_* = \max(\theta_{LB}, f^s(\boldsymbol{x}^k_{AP}))$, $\boldsymbol{x}^s_* = \boldsymbol{x}^k_{AP}$,
        $D^s(\lambda, \mu^s, \boldsymbol{u}) = z^k_{AP} + (\mu^s + \pi^s)\max(\theta_{LB}, f^s(\boldsymbol{x}^k_{AP})) + \lambda(\alpha - 1)\pi^s$;
      **end**
      **if** $k > 1$ & $z^k_{AP} > z^1_{AP} + \lambda\pi^s$ **then** // Based on Lemma 3.1.
        $K^s_1 = k - 1$, continue with the next scenario;
      **end**
      **if** $f^s(\boldsymbol{x}^k_{AP}) \le \theta_{UB}$ & $(\mu^s + \pi^s)(f^s(\boldsymbol{x}^k_{AP}) - \theta_{LB}) \le 0$ **then** // Based on Lemma 3.2.
        $K^s_2 = k$, continue with the next scenario;
      **end**
    **end**
    // Phase III. The optimal solution was not obtained in Phases I-II.
    Solve the formulation (49)-(55) through a mixed-integer programming solver;
  **else** // Only $\left(\mathbf{LS}^s_{\beta^s=1}\right)$ is feasible.
    $\beta^s_* = 1$, $\theta^s_* = \theta_{LB}$, $\boldsymbol{x}^s_* = \boldsymbol{x}^1_{AP}$, $D^s(\lambda, \mu^s, \boldsymbol{u}) = D^s(\lambda, \mu^s, \boldsymbol{u}/\beta^s = 1) = z^1_{AP} + (\mu^s + \pi^s)\theta_{LB} + \lambda\alpha\pi^s$;
  **end**
**end**

$D^s(\lambda, \mu^s, \boldsymbol{u})$ associated with scenario $s$:

$$z_{LD} = \quad \text{maximize} \quad \sum_{s\in S} \eta^s \tag{66}$$

$$\text{subject to} \quad \eta^s \leq (\pi^s + \mu^s)\theta_p^s + \lambda\pi^s(\beta_p^s - 1 + \alpha) + \sum_{j\in N}\sum_{k\in N} u_{jk}\boldsymbol{H}^s x_{jkp}^s, \quad \forall s \in S, p = 1, \ldots, \phi^s, \tag{67}$$

$$\sum_{s\in S} \eta^s \leq \theta_{UB}, \tag{68}$$

$$\mu^s \geq -\pi^s, \qquad \forall s \in S, \tag{69}$$

$$\sum_{s\in S} \mu^s = 0, \tag{70}$$

$$\lambda \geq 0, \tag{71}$$

where the components of $\boldsymbol{x}_p^s$ are indicated by $x_{jkp}^s, \forall j, k \in N$. In the formulation above, the set of constraints (67) defines $D^s(\lambda, \mu^s, \boldsymbol{u})$ for any $s \in S$ as the minimum of a finite number of affine functions and establishes that it is piecewise affine and concave. The same property clearly holds for $D(\lambda, \mu, \boldsymbol{u}) = \sum_{s\in S} D^s(\lambda, \mu^s, \boldsymbol{u})$ as well. Constraint (68) imposes that (66)-(71) is always bounded even if a subset of the constraints (67) is omitted from the formulation as described in the sequel. During the initialization step of *SD-SMS* detailed in Section 4.1, the right hand side of (68) is set to the VaR associated with an initial primal feasible solution of the original problem. The constraints (69) are required for bounded Lagrangian subproblems as discussed in Section 3.1. Finally, constraint (70) reflects the condition (47).

The exponential size of the set of constraints (67) prompts a cut generation scheme for solving the *master problem* (66)-(71). Initially, a *restricted master problem* (*RMP*) is set up only with a small portion of the constraints (67) and new constraints are generated and added to the model on the fly as necessary. More specifically, the cut generation algorithm iteratively solves *RMP*, updates the values of the dual variables, solves the Lagrangian subproblems, and then employs the subproblem solutions to generate new cuts of the form (67) until the optimality gap between the optimal objective value of the current *RMP* and the best known lower bound obtained from the Lagrangian subproblems is reduced below a threshold. A further significant property of the LP formulation for solving (**LD**) is that it does not require that we solve the subproblems to optimality. Essentially, any of the feasible solutions $(\boldsymbol{x}_p^s, \beta_p^s, \theta_p^s)$, $p = 1, \ldots, \phi^s$, to ($\mathbf{LS}^s$) may be employed to generate a cut to be added to *RMP*. This is a fundamental advantage that we exploit in our algorithm for computational speed. We refer to such cuts as *suboptimal* and discuss the specifics of our implementation in Section 4.2.

In the *multi-cut* master problem formulation (66)-(71) above, we approximate the dual function $D(\lambda, \mu, \boldsymbol{u})$ by approximating each of its pieces $D^s(\lambda, \mu^s, \boldsymbol{u}), \forall s \in S$, separately as evident from the set of constraints (67). Alternatively, we could have employed a *single-cut* version of the master problem by aggregating all $|S|$ cuts generated after solving the Lagrangian subproblems via Algorithm 1 and replacing $\sum_{s\in S} \eta^s$ by a single variable $\eta$ in the formulation as appropriate. The single-cut version results in fast solution times for *RMP* at the expense of more iterations overall. From a computational point of view, however, the most demanding step of *SD-SMS* is solving the Lagrangian subproblems. Ultimately, the total solution time can only be reduced if we solve a smaller number of subproblems, and the computational time expended for solving the restricted master problems is a small fraction of the overall solution time as long as a proper cut management strategy is adopted as outlined in Section 4.4. In our preliminary runs, we observed that the multi-cut version outperforms its single-cut counterpart by significantly reducing the number of iterations until the master problem is solved to optimality. Therefore, we exclusively employ the multi-cut version (66)-(71) of the master problem throughout this study.

As a final note, we point out that the Lagrangian dual problem (**LD**) is a non-differentiable optimization problem. Among the several alternate methods available for such problems, we also experimented with the widely known subgradient optimization and a bundle method (see the `ConicBundle` library provided by

Helmberg, 2011). However, both approaches exhibited poor convergence in our preliminary analysis, and we ultimately opted for the LP formulation of (**LD**) solved by cut generation. The simplicity of updating the restricted master formulation as we modify the scenario decomposition as described next, and the ease of incorporating dual stabilization techniques into this approach – see Section 4.3 – further contributed to this decision.

**3.5 Progressive Tightening of the Scenario Decomposition** The master formulation (66)-(71) does clearly illustrate a well-known fact about Lagrangian relaxation. The value of the best lower bound $z_{LD}$ that can be attained by the relaxation depends on the sizes of the feasible regions of the subproblems. If we can shrink $\Phi^s$ then we may be able to eliminate some of the candidate solutions to (**LS**$^s$) from consideration and decrease the cardinality of the set of constraints (67). This, in turn, may potentially increase the value of $z_{LD}$. The proposed scenario decomposition features a special structure that helps us achieve these goals and obtain progressively tighter relaxations of (**G** − **VaR**) over the course of the algorithm *SD-SMS*. We reckon that this is one of the most interesting aspects of our work.

At each iteration of *SD-SMS*, we compute a lower bound $D(\lambda, \mu^s, \boldsymbol{u}) = \sum_{s\in S} D^s(\lambda, \mu^s, \boldsymbol{u})$ on the optimal value of $\theta$, and the primal heuristic described in Section 3.3 is executed with the hope of improving the best known upper bound on $\theta$. This all fits into a classical application of Lagrangian relaxation so far. However, observe that we also deliberately keep a set of lower and upper bounds on $\theta$ in the original problem (**G** − **VaR**), and these bounds automatically factor into the feasible regions of the subproblems (**LS**$^s$) for all $s \in S$ through the constraint (55). This setup presents us with an opportunity to dynamically update the constraint (55) as the best lower and upper bounds on the optimal value of $\theta$ are tightened over the course of *SD-SMS*. At first, $\theta_{LB}$ is trivially set to zero and $\theta_{UB}$ is set to the VaR associated with an initial primal feasible solution of (**G** − **VaR**) determined during the initialization stage of *SD-SMS* described in Section 4.1. Then, every time *SD-SMS* churns out a better lower or upper bound on the optimal value of $\theta$ over its course, this new best bound is plugged into constraint (55) in every subproblem (**LS**$^s$) and results in smaller subproblem feasible regions $\Phi^s, \forall s \in S$. On a related note, the value of the best upper bound on $\theta$ is also inserted into the right hand side of constraint (68) in *RMP*.

The discussion so far implies that $\Phi^s_{i+1} \subseteq \Phi^s_i$ holds for any subproblem (**LS**$^s$), where $\Phi^s_i$ is the feasible region of (**LS**$^s$) in the $i$th iteration of *SD-SMS*. Consequently, cuts are generated from progressively smaller subproblem feasible regions as we iterate, and a cut based on $(\boldsymbol{x}^s, \beta^s, \theta^s) \in \Phi^s_i$ that is currently present in *RMP* may no longer be valid for the modified scenario decomposition because we may have $(\boldsymbol{x}^s, \beta^s, \theta^s) \notin \Phi^s_{i+1}$. Such cuts must either be deleted from *RMP* or modified appropriately. We opt for the second option by creating a feasible solution $(\boldsymbol{x}^s, \beta^s_r, \theta^s_r) \in \Phi^s_{i+1}$ out of $(\boldsymbol{x}^s, \beta^s, \theta^s)$, and then replacing the invalid cut with a cut generated from $(\boldsymbol{x}^s, \beta^s_r, \theta^s_r)$. To this end, we keep the job processing sequence $\boldsymbol{x}^s$ fixed and update $\theta^s$ to $\theta^s_r$ and $\beta^s$ to $\beta^s_r$ by solving (**LS**$^s$) with the duals reset to their initial values and under the restriction that the job processing sequence $\boldsymbol{x}^s$ cannot be modified. This is practically equivalent to applying the logic in lines 5-7 of Algorithm 1 to $\boldsymbol{x}^s$ instead of $\boldsymbol{x}^s_{\min}$. This entire procedure may be regarded as a warm start for a new scenario decomposition with the updated lower and upper bounds on $\theta$ and relies on the concept of suboptimal cuts mentioned in the previous section and detailed in Section 4.2. Overall, solving progressively tighter relaxations of (**G** − **VaR**) as described enhances the solution quality of (**LD**) to a great extent as demonstrated in Section 5.4.

**4. Computational Features and Enhancements** Various computational aspects of the proposed solution approach *SD-SMS* that are not fundamental to the main discussion are discussed in this section. Among these, the use of suboptimal cuts and dual stabilization outlined in Sections 4.2 and 4.3, respectively, are crucial to speed up the convergence of *SD-SMS*. Results regarding their impact are reported in Section 5.4.

**4.1 Initialization** The main task performed during the initialization step of *SD-SMS* is to solve the deterministic single-machine problems associated with each scenario to optimality. To this end, we employ

the Single-Machine Scheduling Problem Solver (SiPS) provided by Tanaka et al. (2009), which is a powerful optimal algorithm for minimizing the total job completion cost in deterministic single-machine scheduling problems without machine idle time. The resulting optimal sequences $\boldsymbol{x}^s_{\min}, \forall s \in S$, are then evaluated for the original problem ($\mathbf{G-VaR}$), and we pick the best contender as the initial primal feasible solution. The associated VaR defines the right hand sides of constraint (55) in the Lagrangian subproblems and constraint (68) in the initial *RMP*.

All dual variables are initialized to zero, following the common practice. This turns out to be a particularly good choice for $\boldsymbol{u}$ as we elaborate on in Section 4.3.

**4.2 Suboptimal Cuts** In Section 3.4, we argued that reducing the number iterations in *SD-SMS* is our main strategy in an effort to decrease the number of subproblems solved and achieve faster overall solution times. Along with a multi-cut master problem formulation and dual stabilization, the use of suboptimal cuts is one of our main tools to this end. The fundamental idea was already introduced in Section 3.4. To obtain a cut from ($\mathbf{LS}^s$), we are not restricted to an optimal solution; any of the feasible solutions $(\boldsymbol{x}^s_p, \beta^s_p, \theta^s_p),\ p = 1, \ldots, \phi^s$, to ($\mathbf{LS}^s$) generates a valid cut. We refer to such cuts as *suboptimal* as introduced previously and append a number of them to *RMP*, even when ($\mathbf{LS}^s$) is solved to optimality, with the hope of approximating the associated dual function more quickly. Moreover, in the early stages of *SD-SMS* while the optimality gap between the upper and lower bounds on $z_{LD}$ is still relatively large, it is not crucial that we identify the deepest cut possible for a scenario subproblem. That is, we may even terminate Algorithm 1 prematurely and generate cuts based on the currently available solutions. One potential issue here is that if a particular subproblem is not solved to optimality, then the contribution of this subproblem to the Lagrangian lower bound must be determined based on a lower bound on its optimal objective value.

In our implementation, we maintain a pool of no more than 10 best solutions for each subproblem not solved to optimality in Phase I of Algorithm 1 and prescribe that the gap between the best and worst solutions in the pool does not exceed 40%. The pool may be populated by both the $K-$assignment algorithm in Phase II and the mixed-integer solver in Phase III. Furthermore, as long as the optimality gap in ($\mathbf{LD}$) is larger than 1% and the optimal objective value of *RMP* has decreased by at least 1% over the last three iterations we skip Phase III in Algorithm 1. That is, in this case we do not invoke the mixed-integer programming solver even if the $K-$assignment algorithm fails to identify the optimal solution for a subproblem. This strategy tends to reduce the computational effort significantly. However, if we detect that *SD-SMS* stalls with an improvement less than 1% in the optimal objective value of *RMP* over the most recent three iterations, then we re-start calling the mixed-integer solver for a subproblem when the $K-$assignment algorithm does not return an optimal solution. Note that if the optimal solution to ($\mathbf{LS}^s$) is not determined in Phase II and Phase III is not executed, then the contribution of ($\mathbf{LS}^s$) to the Lagrangian lower bound is computed by bounding the three components of the subproblem objective independently from below. The resulting expression is $(\pi^s + \mu^s)\theta_{LB} + \lambda\pi^s(\alpha - 1) + z^1_{AP}$.

We always impose a time limit on the solution time of the mixed-integer solver in Phase III of Algorithm 1. For the first iteration, this time limit is $\bar{t}^s_1 = 1$ second for ($\mathbf{LS}^s$) for all $s \in S$. We then follow an adaptive scheme to update these time limits independently depending on the relative difficulty of each subproblem. If Algorithm 1 only reports a suboptimal solution for ($\mathbf{LS}^s$) at the end of Phase III in iteration $i$ with a time limit of $\bar{t}^s_i$, then we set $\bar{t}^s_{i+1} = \max(1, \bar{t}^s_i \times 2^{\min(2, optgap^s)})$, where $optgap^s$ is the optimality gap of ($\mathbf{LS}^s$) retrieved from the solver at termination. Here, the rationale is that the subproblems that terminate with a significant optimality gap should be allowed more time in the next iteration. Otherwise, if the optimal solution is attained the same time limit is maintained. If the solver hits the time limit before proving optimality for ($\mathbf{LS}^s$), then the contribution of this subproblem to the Lagrangian lower bound in the current iteration is the currently best available lower bound from the solver.

**4.3 Dual Stabilization** The speed of convergence of the Lagrangian methods in integer programming – including column generation – generally suffers from a phenomenon known as *dual instability* (Frangioni,

2005; Ben Amor et al., 2009). Described in our context, *SD-SMS* would not terminate even if the optimal solution of (**LD**) would be known at the outset and the initial *RMP* is constructed with the cuts generated after solving the Lagrangian subproblems with this information. The algorithm converges only after collecting enough information "around" the optimal dual solution. However, a good estimate of the optimal dual solution is generally not available initially. Furthermore, we have no control over the dual solutions provided by *RMP* from iteration to iteration, and the successive optimal dual solutions of *RMP* may wildly differ from each other offering no option to investigate a particular part of the feasible region of (**LD**) in detail. Such considerations give rise to dual stabilization methods which offer mechanisms to explore the region around the current *stability center* – an estimate of the optimal dual solution – in depth. The collected information allows the stability center and the way its neighborhood is traversed to be updated in a dynamic fashion with the goal of eventually converging to the optimal dual solution. The interested reader may consult Frangioni (2005); Ben Amor et al. (2009) for further information and references.

Our preliminary studies indicate that the values of $\boldsymbol{u}$ demonstrate high variability between consecutive iterations of *SD-SMS*. This slows down the convergence of the algorithm considerably and prompts us to seek a dual stabilization method. Among the several strategies proposed in the literature, we take the trust-region approach of Kallehauge et al. (2006) employed to solve a Lagrangian dual problem in an application to the vehicle routing problem with time windows and enhance it by an explicit stabilizing penalty term (Ben Amor et al., 2009). The trust-region method of Kallehauge et al. (2006) itself is an extension of the *boxstep* approach of Marsten et al. (1975). Note that we adopt a partial dual stabilization strategy by not applying any stabilization to $\lambda$ and $\boldsymbol{\mu}$.

The original boxstep method constrains the dual variables to remain within a fixed radius $\Delta$ around a center point $\boldsymbol{u}^c$, where $\boldsymbol{u}^c$ is believed to be a good approximation of the optimal dual vector $\boldsymbol{u}$. Alternatively, the objective of *RMP* may be augmented with a stabilizing term that penalizes deviations of $\boldsymbol{u}$ from the stability center $\boldsymbol{u}^c$. Combining these two ideas and encouraged by the computational success of the piecewise linear stabilizing terms (Ben Amor et al., 2009), we append the following *4-piecewise linear penalty function* to the objective of *RMP*:

$$\Psi(\boldsymbol{u}) = \sum_{j \in N} \sum_{k \in N} \Psi_{jk}(u_{jk}),$$

where

$$\Psi_{jk}(u_{jk}) = \begin{cases} -\infty, & \text{if} \quad u_{jk} < u^c_{jk} - \Delta \\ -\zeta(u^c_{jk} - u_{jk}), & \text{if} \quad u^c_{jk} - \Delta \leq u_{jk} \leq u^c_{jk} \\ -\zeta(u_{jk} - u^c_{jk}), & \text{if} \quad u^c_{jk} < u_{jk} \leq u^c_{jk} + \Delta \\ -\infty & \text{if} \quad u^c_{jk} + \Delta < u_{jk} \end{cases} . \tag{72}$$

The stabilizing term $\Psi_{jk}(u_{jk})$ is symmetric; i.e., $\Psi_{jk}(u_{jk}) = -\zeta|u_{jk} - u^c_{jk}|$, where $\zeta$ is the penalty per unit deviation from the stability center. Furthermore, (72) defines the box of width $\Delta$ around $u^c_{jk}$ by setting the penalty coefficients to $-\infty$ outside the boundaries of the box. During the second stage of our dual stabilization for *SD-SMS* described below, the width of the box $\Delta$ and the stability center $\boldsymbol{u}^c$ are updated by adopting the scheme of Kallehauge et al. (2006), but not with all identical parameter values.

We implement a three-stage dual stabilization strategy. As we discussed in Section 3.2, preliminary computational testing uncovered that many components of the optimal $\boldsymbol{u}$ are frequently identical to zero. Therefore, we initially fix $\boldsymbol{u}^c = \boldsymbol{0}$ with $\Delta = 0$ and collect as many cuts as possible under this restriction. Recall that (**LS**$^s$) is solved in constant time for $\boldsymbol{u} = \boldsymbol{0}$ once the optimal solution of the associated deterministic single-machine scheduling problem is available after the initialization step of *SD-SMS*. Therefore, the main effort here is solving *RMP* successively until the optimal objective value of *RMP* is identical to the best Lagrangian lower bound which marks the end of the first stage. In the second stage, we allow nonzero values for the components of $\boldsymbol{u}$ by initially setting $\Delta = 1$. The value of $\zeta = 0.1$ is kept constant throughout. At every iteration $i$ of

*SD-SMS*, we compute a metric $\rho$ before solving *RMP*:

$$\rho = \frac{D(\lambda^i, \boldsymbol{\mu}^i, \boldsymbol{u}^i) - D(\lambda_*^{i-1}, \boldsymbol{\mu}_*^{i-1}, \boldsymbol{u}_*^{i-1})}{z_{RMP}^{i-1} - D(\lambda_*^{i-1}, \boldsymbol{\mu}_*^{i-1}, \boldsymbol{u}_*^{i-1})}.$$

In this expression, $z_{RMP}^{i-1}$ denotes the optimal objective value of *RMP* in iteration $i-1$, $(\lambda^i, \boldsymbol{\mu}^i, \boldsymbol{u}^i)$ and $D(\lambda^i, \boldsymbol{\mu}^i, \boldsymbol{u}^i)$ represent the dual solution and the value of the Lagrangian lower bound in the current iteration, respectively, and $D(\lambda_*^{i-1}, \boldsymbol{\mu}_*^{i-1}, \boldsymbol{u}_*^{i-1})$ is the value of the best Lagrangian lower bound obtained until iteration $i-1$. The metric $\rho$ measures the improvement in the Lagrangian lower bound from iteration $i-1$ to $i$ with respect to the estimated improvement at the end of iteration $i-1$ as expressed through the denominator of $\rho$. If $\rho = 1$, we just move along one of the pieces of the dual function and obtain no new information. Therefore, we expand the boundaries of the trust region by increasing $\Delta$ by 25% and allow $\boldsymbol{u}$ to assume values further away from the stability center. If $\rho < 0$, then we conclude that our approximation of the dual function around the current stability center is poor and shrink the size of the trust region by halving $\Delta$. Finally, if $\rho > 0.01$ we update the stability center $\boldsymbol{u}^c$ to $\boldsymbol{u}^i$ because the improvement in $D(\lambda, \boldsymbol{\mu}, \boldsymbol{u})$ is promising. This is known as a *serious step*. Otherwise, if $\rho \leq 0.01$ we perform a *null step* and keep the current stability center. The second stage of the stabilization scheme continues until the gap between the optimal objective value of *RMP* and the best Lagrangian lower bound drops below a threshold or the improvement in this bound tails off. In the final stage of dual stabilization, we gradually enlarge the trust region and eventually remove all stabilization from *RMP*. *SD-SMS* terminates once the gap between the best lower and upper bounds on $z_{LD}$ is no more than 0.01% or one of the alternate termination conditions specified in Section 5.3 is satisfied.

**4.4 Cut Management** Every iteration of *SD-SMS* introduces at least $|S|$ cuts into *RMP*. This number may be considerably larger if we also generate cuts based on the suboptimal subproblem solutions, and the portion of the solution time attributed to the solution of the master problem may grow fast with an increasing number of scenarios. To alleviate this performance issue, cuts that are inactive over a number of iterations are deleted from *RMP*. In our study, following each re-optimization of *RMP* we analyze the dual variables associated with the cuts. If we identify a dual variable that has not assumed a positive value over the most recent 5 iterations, we declare its associated cut as redundant and remove it from *RMP*. In addition to this strategy, we also invoke the presolve ability of the solver each time before solving *RMP*. Both of these strategies work well to reduce the size of *RMP* and prevent a certain deterioration in the computational performance.

**4.5 Parallel Processing** The advent of more accessible parallel computing architectures and user-friendly parallel computing libraries makes decomposition approaches even more attractive. In *SD-SMS*, the most demanding step is solving the Lagrangian subproblems and we heavily depend on parallelization to enhance the scalability of Algorithm 1 for an increasing number of scenarios. The two $K-$assignment problems to be solved for $s = 1$ and $s \geq 2$ are tackled simultaneously on two different threads, and in Phase II starting on line 20 of Algorithm 1 the generated job processing sequences are evaluated for the individual scenarios in parallel on a fixed number threads. However, we reap the most benefits from distributing the subproblems that are solved as mixed-integer programs in Phase III over a fixed number of available threads. Here, we also carefully balance the load of each thread by employing the well-known *longest processing time rule* in parallel machine scheduling (Pinedo, 2008, Section 5.1). To this end, we collect average solution time statistics for each subproblem over the iterations of *SD-SMS* and assign the subproblems in non-increasing order of these times to the threads as they become available.

**5. Computational Study** We designed our computational study with two main goals in mind. From a modeling point of view, we would like to demonstrate the value of the risk-averse stochastic programming model (**G − VaR**) for decision making purposes. To this end, we evaluate and contrast the solutions provided by *SD-SMS* and those produced by more traditional modeling approaches under uncertainty in Section 5.2. The computational effectiveness of the algorithm *SD-SMS* is investigated in the second part of the study. In Section 5.3, we report on the overall performance of *SD-SMS* for $TT$ and $TWT$. The individual contributions

of some of the fundamental aspects of our solution method are assessed in Section 5.4.

The algorithm *SD-SMS* was implemented in C++. The linear and mixed-integer linear programs are solved through the `Concert Technology` component library of `IBM ILOG CPLEX 12.4`, and the parallelizations are carried out by the `Boost 1.51.0` library. The $K-$assignment problems solved in the second phase of Algorithm 1 as part of the solution method for the Lagrangian subproblems are handled via the algorithm of Pascoal et al. (2003). The number of assignments to be enumerated is set to $\kappa = 10{,}000$ and $50{,}000$ for the performance measures $TT$ and $TWT$, respectively, as a result of a preliminary study with the objective of balancing solution time and quality. The source code of the $K-$assignment algorithm of Pascoal et al. (2003) was gracefully provided by the authors and deployed with no modifications. One disadvantage of this implementation for our purposes is that it runs for a fixed $\kappa$ and does not allow for the retrieval of the assignments one by one as intended in Algorithm 1. This drawback inflates the subproblem solution times to some extent and may be removed in the future for enhanced performance. All runs were executed on 6 threads of an HP workstation with two Intel® Xeon® W5580 3.20 GHz CPUs and 32 GB of memory running on Microsoft Windows Server 2003 R2 Enterprise x64 Edition.

**5.1 Generation of the Problem Instances** While our modeling framework allows for randomness in all problem parameters, we focus on the uncertainty in the processing times in our computational study as justified by the discussion in Section 1. For each instance, we generate a set of equally likely scenarios, where each scenario represents a joint realization of the processing times of all jobs. Each job $j \in N$ has an estimated processing time $\hat{p}_j$ drawn from an integer uniform distribution $U[10, 90]$ and perturbed randomly up or down for each scenario. In particular, if $\varepsilon$ represents a random relative perturbation, where $\varepsilon^s$ is the realization of $\varepsilon$ for scenario $s$, then the processing time of job $j$ under scenario $s$ is given by $p_j^s = \lceil \hat{p}_j(1+\varepsilon^s) \rceil$. We assume that observing a higher relative deviation is more likely for smaller values of the estimated processing times. Following this rationale, we partition the jobs into two groups, where the jobs with $10 \leq \hat{p}_j \leq 60$ belong to the first group and the remaining jobs with $60 < \hat{p}_j \leq 90$ are assigned to the second group. The probability distributions of the perturbations are specified in such a way that the random relative perturbation associated with the first group of jobs has a higher expectation and a higher variance. To this end, we consider the mixture of two group-specific uniform distributions to generate the perturbations in each group, where a smaller probability is associated with the uniform distribution that has higher possible values. The parameters of the mixture distributions and the expectation and the standard deviation of $\varepsilon$ are presented in Table 1 for three different data sets along with the coefficient of variation (CV) of the resulting random processing times. CV is a normalized measure of dispersion calculated by dividing the expectation of a random variable by its standard deviation. For the processing times, we obtain $CV(\hat{p}_j(1+\varepsilon)) = \sigma(\hat{p}_j(1+\varepsilon))/\mathbb{E}(\hat{p}_j(1+\varepsilon)) = \frac{\sigma(\varepsilon)}{1+\mathbb{E}(\varepsilon)}$. Furthermore, $\mathbb{E}(\hat{p}_j(1+\varepsilon)) = \hat{p}_j(1+\mathbb{E}(\varepsilon))$ yields the expectation of the processing times which is not reported separately in Table 1. The purpose of generating three sets of data with different CV values is to analyze the value of the risk-averse stochastic programming model ($\mathbf{G-VaR}$) under different settings in Section 5.2.

The setup above is consistent with the common observation that in many manufacturing environments the processing times tend to fluctuate around their estimated values and large deviations from these values occur with a small probability. These large deviations may be associated with major disruptions, e.g., machine breakdowns, which result in large delays but occur infrequently. In line with this discussion, the specified mixture distributions ensure that the processing times fluctuate around their $\hat{p}_j$ values with a high probability and take significantly larger values with a small probability.

In the literature, it is well established that the tightness and the range of the due dates are the primary determinants of the difficulty for due date related problems. Thus, by following the popular scheme of Potts and van Wassenhove (1982), we first generate the due dates from a discrete uniform distribution $[\lceil (1 - \mathrm{TF} - \mathrm{RDD}/2) \cdot \bar{P} \rceil, \lceil (1 - \mathrm{TF} + \mathrm{RDD}/2) \cdot \bar{P} \rceil]$, where $\bar{P}$ is the sum of the expected processing times, i.e., $\bar{P} = \sum_{j \in N} \mathbb{E}(\hat{p}_j(1+\varepsilon))$. The tardiness factor TF is a rough estimate of the proportion of jobs that might

| | Random Relative Perturbations | | | Random Processing Times |
|---|---|---|---|---|
| | Mixture Distribution | $\mathbb{E}(\varepsilon)$ | $\sigma(\varepsilon)$ | $CV(\hat{p}_j(1+\varepsilon))$ |
| **Group 1** | | | | |
| **Data Set 1** | $\varepsilon \sim \begin{cases} U(-0.10, 0.50) & \text{w/ prob } 0.90 \\ U(2.0, 3.0) & \text{w/ prob } 0.10 \end{cases}$ | 0.43 | 0.72 | 0.50 |
| **Data Set 2** | $\varepsilon \sim \begin{cases} U(-0.10, 0.20) & \text{w/ prob } 0.90 \\ U(3.0, 4.0) & \text{w/ prob } 0.10 \end{cases}$ | 0.40 | 1.04 | 0.75 |
| **Data Set 3** | $\varepsilon \sim \begin{cases} U(-0.10, 0.20) & \text{w/ prob } 0.90 \\ U(4.0, 5.0) & \text{w/ prob } 0.10 \end{cases}$ | 0.50 | 1.34 | 0.90 |
| **Group 2** | | | | |
| **Data Set 1** | $\varepsilon \sim \begin{cases} U(-0.10, 0.50) & \text{w/ prob } 0.99 \\ U(1.0, 1.5) & \text{w/ prob } 0.01 \end{cases}$ | 0.21 | 0.20 | 0.17 |
| **Data Set 2** | $\varepsilon \sim \begin{cases} U(-0.10, 0.25) & \text{w/ prob } 0.975 \\ U(1.5, 2.5) & \text{w/ prob } 0.025 \end{cases}$ | 0.12 | 0.32 | 0.28 |
| **Data Set 3** | $\varepsilon \sim \begin{cases} U(-0.10, 0.25) & \text{w/ prob } 0.95 \\ U(2.0, 3.0) & \text{w/ prob } 0.05 \end{cases}$ | 0.20 | 0.54 | 0.45 |

Table 1: Parameters and information related to the generation of the random processing times.

be expected to be tardy in an arbitrary sequence (Srinivasan, 1971) and is set to 0.4 or 0.6. Note that smaller TF values often lead to easy instances with an optimal VaR of zero. The due date range factor RDD is set to 0.2 to increase the contention around the average due date. In order to assign the weights for the $TWT$ instances, we follow a strategy similar to that in Pinedo and Singer (1999). The weights are set randomly to one of three possible values 3, 2, and 1, which represent high, medium, and low job priority levels, respectively. The associated respective probabilities are 0.2, 0.6, and 0.2.

**5.2 Value of the Risk-Averse Model** The value of a risk-averse model depends on its performance relative to traditional models for decision making under uncertainty which ignore the variability inherent in the system. Therefore, in this part, we benchmark the solutions produced by our risk-averse stochastic programming model against those provided by the corresponding deterministic and risk-neutral models as the risk parameter $\alpha$ is varied. The deterministic counterpart of our problem is a conventional single-machine scheduling problem, in which all processing times take on their expected values; we set $p_j$ to its estimated expected value of $\lceil \sum_{s\in S} \pi^s p_j^s \rceil$. In the risk-neutral version of our problem, we minimize the expected performance measure by solving the following formulation:

$$\begin{aligned} \text{minimize} \quad & \sum_{s\in S} \pi^s f^s(\boldsymbol{x}) \\ \text{subject to} \quad & (2) - (4). \end{aligned} \tag{73}$$

For the analyses in this section, the deterministic model is solved to optimality by the `SiPS` solver provided by Tanaka et al. (2009). *SD-SMS* is invoked with a time limit of 1800 seconds, and `CPLEX` is allotted the same period of time for solving the risk-neutral model.

In Figure 1, we zoom into an instance with $n = 15$ and $|S| = 500$ from Data Set 1 under the performance measure $TT$ in order to illustrate how VaR changes as $\alpha$ is varied. We observe in Figure 1(a) that the performance of the risk-neutral and deterministic models deteriorate with increasing levels of $\alpha$ in terms of the risk measure VaR. Perhaps more importantly, for this instance we obtain risk-averse solutions that do not sacrifice much from the expected $TT$ as $\alpha$ increases. In particular, according to Figure 1(b) the risk-neutral and the risk-averse models attain the same expected $TT$ for $\alpha \leq 0.85$, and for larger values of $\alpha$ the sacrifice from the expected $TT$ in the risk-averse solutions is relatively small compared to the gain in VaR.

To further shed light into the behaviors of the deterministic, risk-neutral, and risk-averse models, we analyze the same specific instance in more depth and calculate the empirical cumulative distribution function (CDF)

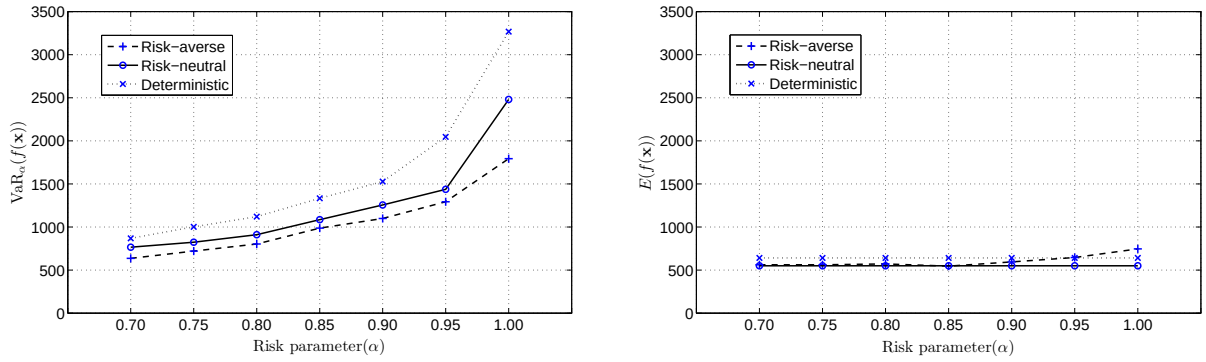



Figure 1: The effect of the risk parameter $\alpha$ on the VaR and the expectation of $f(\boldsymbol{x})$.

of the random $TT$ associated with the best available sequence of each model, where $\alpha = 0.9$ for *SD-SMS*. That is, for each of the three sequences we calculate the realization of the specified performance measure $TT$ under each scenario and derive the associated empirical CDFs depicted in Figure 2. The proposed risk-averse approach of minimizing VaR essentially shapes the CDF of the random performance measure according to the specified confidence level. In particular, it leads to a left shift in the associated right tail of the CDF; for the risk-averse sequence the portion of the CDF corresponding to $\alpha \geq 0.9$ appears to the left of the related portions of the CDFs of the sequences provided by the risk neutral and deterministic models. As a trade-off, the expectation increases and it implies a right shift in the left tail of the CDF associated with the risk-averse sequence.

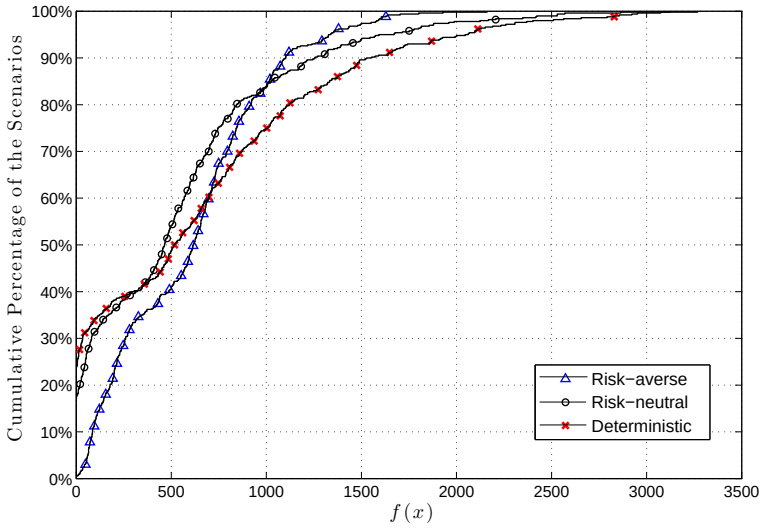



Figure 2: Empirical CDFs of the random $TT$.

We also present comparative results under the performance measure $TT$ for several problem instances from three data sets with different variability in the processing times. See Table 1 for details and note that the variability increases from Data Set 1 toward Data Set 3. A total of five instances for each combination of $n = 15, 20$, $|S| = 50, 100, 200, 300, 400, 500$, and TF $= 0.4, 0.6$, are solved by *SD-SMS* with $\alpha = 0.90$ and by the risk-neutral model. Note that these are a subset of the instances from the next section. For these instances, the entries in Table 2 indicate the relative percentage decrease in VaR and the relative percentage increase in the expected $TT$ for the solutions of the risk-averse model in comparison to those of the risk-neutral model averaged over 10 instances for each pair of $n$ and $|S|$ values. The risk-averse solutions exhibit significant improvements in VaR over their risk-neutral counterparts, albeit at times at the expense of the expected $TT$

in order to hedge against the uncertainty. Moreover, we observe that the trade-off between the expectation and the VaR criteria becomes more pronounced when the processing times have a larger CV.

| | | **Relative Gap (%)** | | | | | |
|---|---|---|---|---|---|---|---|
| | | **Data Set 1** | | **Data Set 2** | | **Data Set 3** | |
| **n** | **\|S\|** | $\mathbb{E}(f(\boldsymbol{x}))$ | $\text{VaR}_\alpha(f(\boldsymbol{x}))$ | $\mathbb{E}(f(\boldsymbol{x}))$ | $\text{VaR}_\alpha(f(\boldsymbol{x}))$ | $\mathbb{E}(f(\boldsymbol{x}))$ | $\text{VaR}_\alpha(f(\boldsymbol{x}))$ |
| 15 | 50 | 5.4 | -7.6 | 9.3 | -11.8 | 8.6 | -12.8 |
| | 100 | 4.2 | -6.0 | 9.1 | -8.8 | 7.3 | -13.2 |
| | 200 | 5.4 | -6.2 | 8.8 | -11.3 | 6.5 | -9.5 |
| | 300 | 4.7 | -6.4 | 7.9 | -9.1 | 6.0 | -8.0 |
| | 400 | 5.7 | -6.5 | 6.7 | -8.2 | 6.4 | -8.2 |
| | 500 | 7.0 | -6.1 | 9.0 | -8.0 | 6.8 | -9.6 |
| 20 | 50 | 6.4 | -7.2 | 8.1 | -13.4 | 6.2 | -13.7 |
| | 100 | 3.7 | -4.8 | 7.8 | -11.1 | 6.0 | -11.4 |
| | 200 | 2.5 | -3.6 | 5.6 | -6.6 | 4.1 | -6.4 |
| | 300 | 2.5 | -3.2 | 4.3 | -5.5 | 4.1 | -6.1 |
| | 400 | 2.8 | -3.6 | 4.1 | -5.3 | 3.5 | -5.0 |
| | 500 | 3.3 | -4.4 | 4.6 | -5.0 | 5.1 | -6.3 |
| **Average:** | | **4.5** | **-5.5** | **7.1** | **-8.7** | **5.9** | **-9.2** |

Table 2: Benchmarking risk-averse solutions against risk-neutral solutions under $TT$.

**5.3 Computational Performance of the Proposed Algorithm** The second part of our study demonstrates that our proposed algorithm *SD-SMS* provides good lower and upper bounds in short computation times for the problem of minimizing VaR under the performance measures $TT$ and $TWT$ on a single machine. Following the specifications for Data Set 1 in Table 1, we generate 5 instances for each combination of $n = 10, 15, 20, 25, 30$, $|S| = 50, 100, 200, 300, 400, 500$, and TF $= 0.4, 0.6$, as described in Section 5.1. To ensure a healthy interpretation of the results as the number of scenarios grows for a fixed number of jobs, instances with $|S| < 500$ are created from the respective instances with $|S| = 500$ and otherwise identical instance generation parameters by simply deleting the required number of scenarios. Furthermore, the $TT$ instances are identical to the $TWT$ instances except that all weights are set to one so that we are able to contrast the relative difficulty of minimizing VaR under $TT$ and $TWT$ in a meaningful way. The risk parameter $\alpha$ is set to 0.90, and the instances are solved under the specified performance measures $TT$ ($n = 10$ excluded) and $TWT$ ($n = 25, 30$ excluded) by both *SD-SMS* and the state-of-the-art solver `CPLEX`. For `CPLEX`, we employ the formulation LOF per our discussion at the end of Section 2.2 as it outperforms APDF for large instances. Both `CPLEX` and *SD-SMS* are allowed to use up to 6 parallel threads as mentioned at the start of Section 5. All reported times are elapsed times, and the time limit is set to 1800 seconds for both algorithms. `CPLEX` is invoked with its default set of options and parameters. *SD-SMS* terminates if the relative optimality gap between the best lower bound $D(\lambda_*, \boldsymbol{\mu}_*, \boldsymbol{u}_*)$ on $z_{LD}$ and the optimal objective value of *RMP* is reduced to below 0.01% after removing the stabilizing terms from *RMP* (see Section 4.3) or if *SD-SMS* verifies that a primal feasible solution is optimal by comparing its VaR to $D(\lambda_*, \boldsymbol{\mu}_*, \boldsymbol{u}_*)$. If optimality is not proven within the time allotted, we record both the best lower bound on the optimal VaR and the incumbent solution available for either method.

The results for the performance measures $TT$ and $TWT$ appear in Tables 3 and 4, respectively. For each combination of $n$ and $|S|$, we present the relative gaps of the upper and lower bounds on the optimal VaR yielded by *SD-SMS* with respect to their counterparts obtained by `CPLEX`, as well as the optimality gaps of the best primal feasible solutions from *SD-SMS*. The gaps in the bounds are calculated by subtracting the values provided by `CPLEX` from those reported by *SD-SMS* and taking the ratio of this difference with respect to the value from `CPLEX`. Thus, the negative relative gaps in the upper bounds and the positive relative gaps in the lower bounds indicate improvements over `CPLEX` by *SD-SMS*. The optimality gap of the best primal

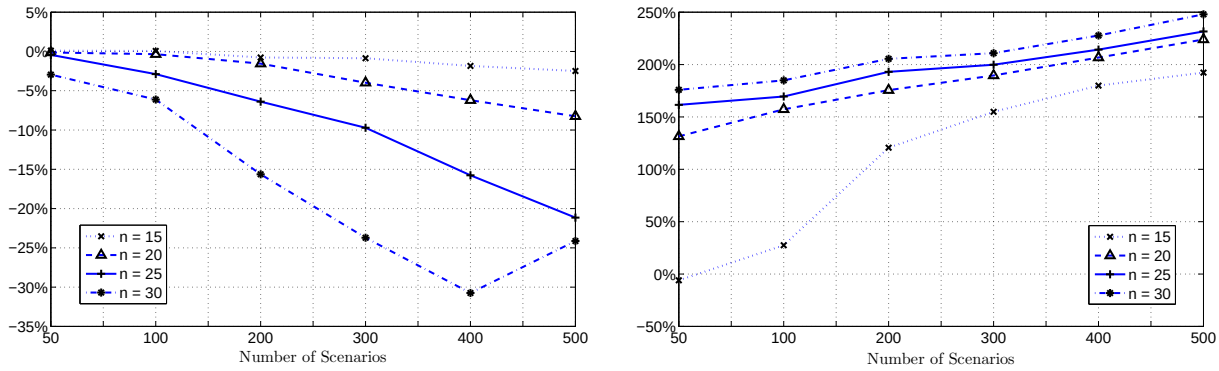
feasible solution identified by *SD-SMS* is computed by the expression $(\theta^*_{UB} - \theta^*_{LB})/\theta^*_{UB}$, where $\theta^*_{UB}$ and $\theta^*_{LB}$ denote the best upper bound obtained by *SD-SMS* and the best known lower bound, respectively. The best known lower bound is determined by taking the maximum of our lower bound $D(\lambda_*, \boldsymbol{\mu}_*, \boldsymbol{u}_*)$ and the best lower bound retrieved from CPLEX. The relative improvements and optimality gaps are given as percentages and all presented results are averaged over 5 instances.

| | | **TF=0.4** | | | | | **TF=0.6** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **Time (sec.)** | | **Rel. Gap (%)** | | ***SD-SMS*** | **Time (sec.)** | | **Rel. Gap (%)** | | ***SD-SMS*** |
| **n** | **\|S\|** | ***SD-SMS*** | CPLEX | **UB** | **LB*** | **Opt. Gap (%)** | ***SD-SMS*** | CPLEX | **UB** | **LB** | **Opt. Gap (%)** |
| 15 | 50 | 29.0 | 1684.8 | 0.0 | 0.3 | 9.2 | 59.8 | 1603.7 | 0.2 | -12.4 | 2.9 |
| | 100 | 19.1 | 1800.6 | 0.3 | 56.8 | 17.1 | 11.9 | 1800.4 | -0.1 | -1.8 | 14.8 |
| | 200 | 13.3 | 1800.4 | -0.5 | 394.5 | 20.7 | 13.9 | 1800.3 | -1.0 | 10.5 | 18.5 |
| | 300 | 25.6 | 1800.8 | -0.8 | 837.7 | 21.9 | 44.5 | 1800.2 | -0.9 | 32.6 | 18.7 |
| | 400 | 28.3 | 1800.8 | -1.1 | 7712.6 [1] | 20.6 | 65.8 | 1800.2 | -2.6 | 59.8 | 19.2 |
| | 500 | 28.8 | 1801.1 | -1.6 | - | 19.5 | 56.6 | 1800.2 | -3.3 | 84.7 | 18.2 |
| 20 | 50 | 36.5 | 1800.6 | -0.3 | 253.9 | 16.8 | 60.8 | 1800.4 | 0.0 | 9.0 | 12.3 |
| | 100 | 31.5 | 1800.5 | -0.6 | 9906.6 | 16.3 | 99.3 | 1800.2 | 0.0 | 14.5 | 15.9 |
| | 200 | 40.6 | 1801.0 | -1.0 | - | 18.9 | 168.2 | 1800.1 | -2.1 | 51.1 | 16.8 |
| | 300 | 122.0 | 1801.6 | -2.8 | - | 19.6 | 161.8 | 1800.1 | -5.2 | 79.1 | 17.2 |
| | 400 | 45.7 | 1801.9 | -2.8 | - | 19.6 | 171.9 | 1800.2 | -9.6 | 113.4 | 17.4 |
| | 500 | 72.8 | 1801.8 | -5.3 | - | 19.3 | 188.0 | 1800.2 | -11.2 | 147.8 | 17.0 |
| 25 | 50 | 152.6 | 1800.7 | -0.4 | 4831.5 [2] | 15.3 | 216.1 | 1800.2 | -0.4 | 23.0 | 14.5 |
| | 100 | 253.9 | 1801.3 | -3.6 | - | 18.5 | 486.0 | 1800.2 | -2.2 | 38.9 | 16.3 |
| | 200 | 216.2 | 1801.8 | -5.3 | - | 22.2 | 517.0 | 1800.1 | -7.5 | 86.0 | 18.3 |
| | 300 | 200.9 | 1800.6 | -9.8 | - | 22.6 | 388.3 | 1800.2 | -9.6 | 99.7 | 18.8 |
| | 400 | 229.4 | 1801.1 | -11.4 | - | 21.5 | 326.6 | 1800.2 | -20.1 | 128.5 | 18.8 |
| | 500 | 353.4 | 1800.6 | -15.8 | - | 21.0 | 600.2 | 1800.3 | -26.5 | 163.1 | 18.2 |
| 30 | 50 | 151.1 | 1801.7 | -1.4 | - | 17.1 | 233.8 | 1800.3 | -4.5 | 51.7 | 15.2 |
| | 100 | 254.8 | 1801.8 | -6.0 | - | 18.5 | 709.0 | 1800.2 | -6.2 | 69.8 | 18.0 |
| | 200 | 233.1 | 1800.8 | -11.0 | - | 21.4 | 951.2 | 1800.2 | -20.3 | 110.8 | 18.7 |
| | 300 | 328.0 | 1800.4 | -22.1 | - | 20.9 | 988.1 | 1800.3 | -25.3 | 121.8 | 19.0 |
| | 400 | 312.5 | 1800.6 | -29.5 | - | 20.4 | 982.9 | 1800.4 | -32.0 | 155.4 | 18.7 |
| | 500 | 341.4 | 1800.6 | -16.3 | - | 20.3 | 938.3 | 1800.5 | -32.0 | 195.9 | 18.5 |

* A dash (-) indicates that CPLEX yields a trivial lower bound of zero for all five instances.
Otherwise, the # of instances in which CPLEX attains a positive lower bound is presented in square brackets.
No bracket implies a positive lower bound for all five instances.

Table 3: Computational performance of *SD-SMS* with respect to CPLEX under $TT$.

We first discuss the results obtained under $TT$. According to Table 3, CPLEX cannot solve any of the instances to optimality within the time limit, with the exception of two instances with $n = 15$ and $|S| = 50$. In contrast, *SD-SMS* terminates in general with better feasible solutions in significantly smaller times. The improvement in the solution quality becomes increasingly more apparent with the increasing number of jobs and scenarios as illustrated in Figure 3(a); we observe that the average relative gap between the upper bounds of *SD-SMS* and CPLEX can grow up to 30%. The results in Table 3 also indicate that the instances with TF = 0.6 are more challenging for CPLEX to perform on a par with *SD-SMS* in constructing good feasible solutions. The average relative gap over all relevant instances is -9.27% for TF = 0.6 in contrast to -6.21% for TF = 0.4. An even more substantial conclusion from Table 3 is that the proposed scenario decomposition scheme is greatly superior to CPLEX in identifying tight lower bounds as illustrated in Figure 3(b). The lower bound performance of CPLEX degrades notably with an increasing number of jobs and scenarios, and CPLEX fails to deliver a non-trivial positive lower bound at termination for most of the instances with TF = 0.4. Clearly, the lower bounds resulting from the proposed scenario decomposition scheme may prove instrumental




(a) Relative upper bound gaps.

(b) Relative lower bound gaps.

Figure 3: Average relative performance of *SD-SMS* with respect to CPLEX for $TT$*.

* The averages are aggregated over TF = 0.4, 0.6. In Figure 3(b), the gaps of the trivial lower bounds of CPLEX and the gaps in excess of 300% are assumed to be 300%.

for future efforts to develop exact methods to solve the risk-averse scheduling problem under consideration. Furthermore, assessing the overall quality of the primal feasible solutions generated by *SD-SMS* through their optimality gaps we observe that the average gap over the whole data set is approximately 17.9%. The optimality gaps tend to increase slightly with the number of scenarios for a fixed number of jobs. However, we reckon that the performance of *SD-SMS* is quite robust as the optimality gap generally hovers in the range of 15-20%. Finally, we note that TF is an important parameter that affects the solution time performance of *SD-SMS*. In particular, Table 3 reveals that the elapsed times required to solve the instances with TF = 0.6 can grow up to 3 times of those required to solve their counterparts with TF = 0.4.

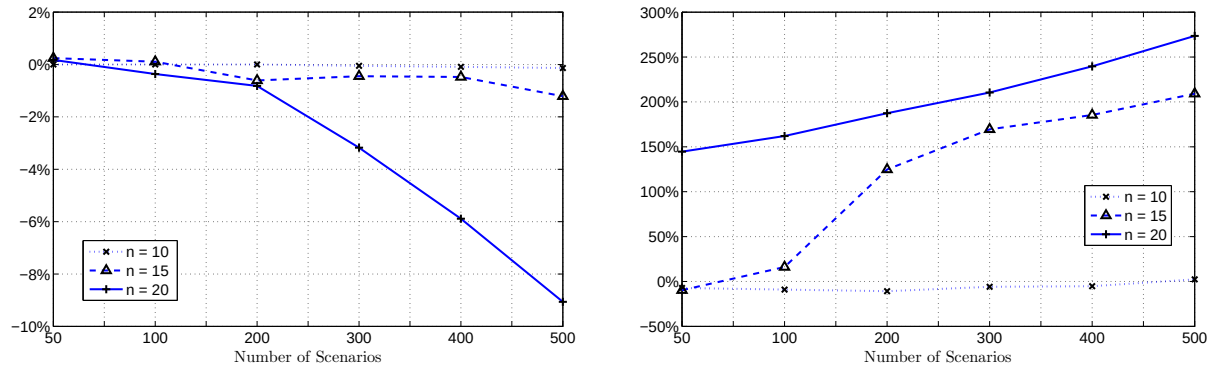
| | | **TF=0.4** | | | | | **TF=0.6** | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|
| | | **Time (sec.)** | | **Rel. Gap (%)** | | ***SD-SMS*** | **Time (sec.)** | | **Rel. Gap (%)** | | ***SD-SMS*** |
| **n** | **\|S\|** | ***SD-SMS*†** | **CPLEX** | **UB** | **LB*** | **Opt. Gap (%)** | ***SD-SMS*†** | **CPLEX** | **UB** | **LB** | **Opt. Gap (%)** |
| 10 | 50 | 37.8 | 8.6 | 0.0 | -5.4 | 0.0 | 117.5 | 4.5 | 0.0 | -9.0 | 0.0 |
| | 100 | 39.2 | 42.3 | 0.0 | -7.9 | 0.0 | 58.0 | 31.7 | 0.0 | -10.1 | 0.0 |
| | 200 | 33.2 | 274.6 | 0.0 | -9.5 | 0.0 | 354.3 | 456.1 | 0.0 | -12.2 | 0.0 |
| | 300 | 55.5 | 922.6 | 0.0 | -6.3 | 3.8 | 179.5 | 1263.5 | -0.1 | -5.6 | 7.4 |
| | 400 | 27.2 | 1148.9 | 0.0 | -11.5 | 0.0 | 220.4 | 1598.3 | -0.2 | 1.0 | 11.3 |
| | 500 | 54.8 | 1237.3 | 0.0 | -9.3 | 3.1 | 105.7 | 1800.3 | -0.3 | 14.0 | 13.5 |
| 15 | 50 | 841.7 | 1025.0 | 0.2 | -6.5 | 3.0 | 1855.6 | 439.5 | 0.3 | -12.9 | 0.3 |
| | 100 | 708.5 | 1800.4 | 0.1 | 37.4 | 16.1 | 1875.1 | 1696.6 | 0.1 | -5.2 | 9.4 |
| | 200 | 1829.1 | 1800.4 | -0.8 | 327.0 | 16.2 | 1614.5 | 1800.2 | -0.4 | 18.1 | 15.8 |
| | 300 | 905.7 | 1800.6 | -0.5 | 3949.4 [4] | 15.5 | 1958.3 | 1800.1 | -0.4 | 39.0 | 15.1 |
| | 400 | 960.5 | 1800.3 | -0.5 | - | 14.9 | 1605.0 | 1800.1 | -0.4 | 71.0 | 15.3 |
| | 500 | 533.3 | 1800.5 | 0.0 | - | 13.6 | 1784.2 | 1800.1 | -2.4 | 118.3 | 14.5 |
| 20 | 50 | 1023.9 | 1800.5 | 0.3 | 328.2 | 10.1 | 2009.7 | 1800.4 | 0.0 | 12.5 | 9.9 |
| | 100 | 1293.5 | 1800.5 | -0.8 | 1297.1 | 12.0 | 1771.3 | 1800.2 | 0.0 | 24.0 | 10.5 |
| | 200 | 1864.8 | 1800.9 | -0.2 | - | 13.4 | 1942.0 | 1800.1 | -1.4 | 74.9 | 12.7 |
| | 300 | 1099.5 | 1801.0 | -1.9 | - | 12.3 | 1915.4 | 1800.3 | -4.4 | 121.0 | 13.0 |
| | 400 | 1326.6 | 1801.3 | -2.6 | - | 13.3 | 1994.4 | 1800.2 | -9.2 | 179.2 | 13.3 |
| | 500 | 1541.8 | 1801.3 | -5.9 | - | 11.5 | 2175.5 | 1800.1 | -12.3 | 247.0 | 13.7 |

* A dash (-) indicates that CPLEX yields a trivial lower bound of zero for all five instances.
Otherwise, the # of instances in which CPLEX attains a positive lower bound is presented in square brackets.
No bracket implies a positive lower bound for all five instances.

† *SD-SMS* is only terminated after solving *RMP*. Consequently, the specified time limit may be exceeded during the last iteration if some Lagrangian subproblems prove to be time consuming to solve.

Table 4: Computational performance of *SD-SMS* with respect to CPLEX under $TWT$.

Minimizing VaR under $TWT$ turns out to be significantly more expensive compared to the same task under TT from a computational point of view, as we may also intuitively project from accumulated practical and theoretical experience with these performance measures in different problem settings in the literature. Taking this challenge into account, we experiment with instances of smaller size and only consider $n = 10, 15, 20$ for $TWT$. From Table 4, we can draw observations which are similar to those discussed above for $TT$. We mainly elaborate on the differing aspects here. On the one hand, CPLEX struggles even with instances with 10 jobs and $|S| \geq 300$ as evident from the rapidly growing corresponding elapsed times and hits the time limit quickly for instances with $n = 15, 20$ as it did in Table 3. On the other hand, $TWT$ poses significant difficulties for *SD-SMS* as well. The solution times for TF = 0.4 are at least an order of magnitude greater compared to the respective values for $TT$, and *SD-SMS* hits the time limit for 15- and 20-job instances with TF = 0.6 consistently. However, a very positive takeaway from Tables 3 and 4 is that the performance of *SD-SMS* is only affected slightly by an increasing number of scenarios in terms of both the solution time and quality. *SD-SMS* reaps the benefits of the subproblem solution algorithm designed in a way that some of its most time consuming routines are performed twice independent of the number of scenarios, which in turn allows us to exploit the benefits of scenario decomposition. The main challenge for *SD-SMS* is an increase in the number of jobs, rather than in the number of scenarios. The relative lower and upper bound gaps of *SD-SMS* with respect to CPLEX in Table 4 exhibit patterns that are similar to those of the same size instances in Table 3 and are also summarized graphically in Figure 4. In contrast to the poor lower bound performance of CPLEX for $n = 15, 20$, *SD-SMS* attains reasonably tight lower bounds – in particular for TF = 0.4 – even when convergence cannot be achieved within the time limit. In addition, somewhat surprisingly, we obtain primal feasible solutions of higher quality for $TWT$ than for $TT$. For small instances with 10 jobs, we almost always identify the optimal solution as verified by the results of CPLEX, and for the larger instances with $n = 15, 20$ the average optimality gap stands at 12.31% in contrast to the respective figure 17.02% for $TT$.




(a) Relative upper bound gaps.

(b) Relative lower bound gaps.

Figure 4: Average relative performance of *SD-SMS* with respect to CPLEX for $TWT$*.

* The averages are aggregated over TF = 0.4, 0.6. In Figure 4(b), the gaps of the trivial lower bounds of CPLEX and the gaps in excess of 300% are assumed to be 300%.

**5.4 Effectiveness of the Algorithmic Features** In this section, we analyze the impact and effectiveness of four essential algorithmic features embedded in the proposed algorithm *SD-SMS*: the progressive tightening of the scenario decomposition, dual stabilization, the use of suboptimal cuts, and the subproblem solution algorithms. We report illustrative results for a subset of the instances from Section 5.3.

We start with the progressive tightening of the scenario decomposition described in Section 3.5. Recall that during the course of *SD-SMS*, the feasible regions of the Lagrangian subproblems are shrunk in an effort to improve the final value of the Lagrangian lower bound as the bounds on the optimal value of $\theta$ are tightened. To evaluate the impact of this feature on the quality of the lower and upper bounds on the optimal VaR produced by *SD-SMS*, we run the algorithm on the instances with $n = 15, 20$ in Table 3 when this feature

is disabled. In particular, we set $\theta_{LB}$ to zero and $\theta_{UB}$ to the VaR associated with the initial primal feasible solution in constraint (55) in the Lagrangian subproblems and constraint (68) in *RMP* and then never update these bounds at an intermediate iteration. The average percentage changes in the upper and lower bounds on the optimal VaR with respect to the original results are reported in Table 5 under 'UB' and 'LB', respectively. As in Tables 3-4, positive relative changes in the lower bounds and negative relative changes in the upper bounds designate improvements over the original values. The loss of quality in the bounds across the board is evident, and the lower bounds are particularly affected. For $|S| = 400, 500$ and TF $= 0.4$, the lower bounds are reduced to about one tenth of their original values on average. Moreover, due to these weak lower bounds, the average optimality gaps of the best primal feasible solutions identified by *SD-SMS* – depicted in Columns 'Opt. Gap' in Table 5 – appear to be much poorer compared to those in Table 3. The average optimality gap of the primal feasible solutions over all instances in Table 5 jumps to 55.2% from 17.0% in Table 3. We can safely assert that both `CPLEX` and also *SD-SMS* without the progressive tightening of the scenario decomposition generate loose lower bounds, and this feature is fundamental to the promise of *SD-SMS* as a viable solution method to our risk-averse scheduling problem.

| | | **TF=0.4** | | | **TF=0.6** | | |
|---|---|---|---|---|---|---|---|
| | | **Rel. Gap (%)** | | ***SD-SMS*** | **Rel. Gap (%)** | | ***SD-SMS*** |
| **n** | **\|S\|** | **UB** | **LB** | **Opt. Gap** | **UB** | **LB** | **Opt. Gap** |
| 15 | 50 | 6.0 | -38.5 | 16.9 | 6.9 | -28.5 | 43.0 |
| | 100 | 4.6 | -37.9 | 48.1 | 7.2 | -28.7 | 44.6 |
| | 200 | 3.3 | -37.2 | 51.7 | 7.4 | -28.9 | 46.1 |
| | 300 | 3.0 | -37.8 | 52.8 | 6.5 | -28.9 | 45.7 |
| | 400 | 9.8 | -92.9 | 94.7 | 4.7 | -48.5 | 59.9 |
| | 500 | 1.4 | -95.8 | 96.5 | 4.3 | -57.2 | 66.0 |
| 20 | 50 | 8.3 | -31.4 | 47.2 | 8.9 | -25.8 | 40.2 |
| | 100 | 6.3 | -34.7 | 48.4 | 6.0 | -26.4 | 41.7 |
| | 200 | 4.3 | -34.5 | 49.0 | 4.2 | -26.9 | 41.6 |
| | 300 | 3.7 | -34.4 | 49.1 | 4.3 | -26.6 | 41.7 |
| | 400 | 3.2 | -90.7 | 92.6 | 3.3 | -42.3 | 53.5 |
| | 500 | 2.4 | -93.2 | 94.5 | 3.3 | -49.3 | 59.0 |
| **Average:** | | **4.7** | **-54.9** | **61.8** | **5.6** | **-34.8** | **48.6** |

Table 5: The impact of the progressive tightening of the scenario decomposition on the bounds produced by *SD-SMS*.

We proceed by demonstrating the impact of the dual stabilization on the behavior of *SD-SMS*. As discussed in Section 4.3, a rudimentary implementation of *SD-SMS* without dual stabilization suffers from a slow convergence due to the high variability exhibited by the values of $\boldsymbol{u}$ between the consecutive iterations of the algorithm. To investigate the role of dual stabilization in the performance of *SD-SMS*, we run the algorithm with and without dual stabilization on a particular instance with $n = 15$ and $|S| = 50$ from Data Set 1 under the performance measure $TT$. Both the stabilized and unstabilized versions attain the optimal solution of (**LD**) at the final iteration. To compare and contrast the progress of the dual solutions over the course of the execution of *SD-SMS* in either case, we plot in Figure 5 the Euclidean distance $\|(\lambda^i, \boldsymbol{\mu}^i, \boldsymbol{u}^i) - (\lambda^*, \boldsymbol{\mu}^*, \boldsymbol{u}^*)\|$, where $(\lambda^i, \boldsymbol{\mu}^i, \boldsymbol{u}^i)$ are the values of the dual variables at iteration $i$ and $(\lambda^*, \boldsymbol{\mu}^*, \boldsymbol{u}^*)$ denote the optimal dual variables. In these runs, the cut management routine of Section 4.4 is disabled in the unstabilized version of *SD-SMS* due its detrimental impact on performance. In this case, the fluctuations in $\boldsymbol{u}$ lead to cuts that approximate different portions of the dual function, generally not in the proximity of the optimal dual solution, in an ad hoc manner. Many of these cuts get removed – and then possibly regenerated – until we attain a close representation of the dual function late in the algorithm. An immediate observation from Figure 5 is that the unstabilized algorithm requires three times more iterations compared to its stabilized counterpart in order to converge to the same optimum. The dual variables of the unstabilized algorithm are far away from the optimum for the better part of the iterations. In fact, the algorithm spends 55 iterations before it

can identify a non-trivial positive Lagrangian lower bound in this case. In contrast, the third iteration of the stabilized algorithm yields a positive lower bound. This pattern is fairly typical. Our studies indicate that in many cases, the number of iterations spent by the unstabilized algorithm before attaining a positive lower bound is higher than the total number of iterations required for its stabilized counterpart to converge. As a final note on the subject, observe that *SD-SMS* with dual stabilization makes sustained progress toward the optimum aside from the fluctuations early in the algorithm. The jumps halfway through and toward the final iteration mark the transitions from one stage to the next in the dual stabilization scheme – see Section 4.3.

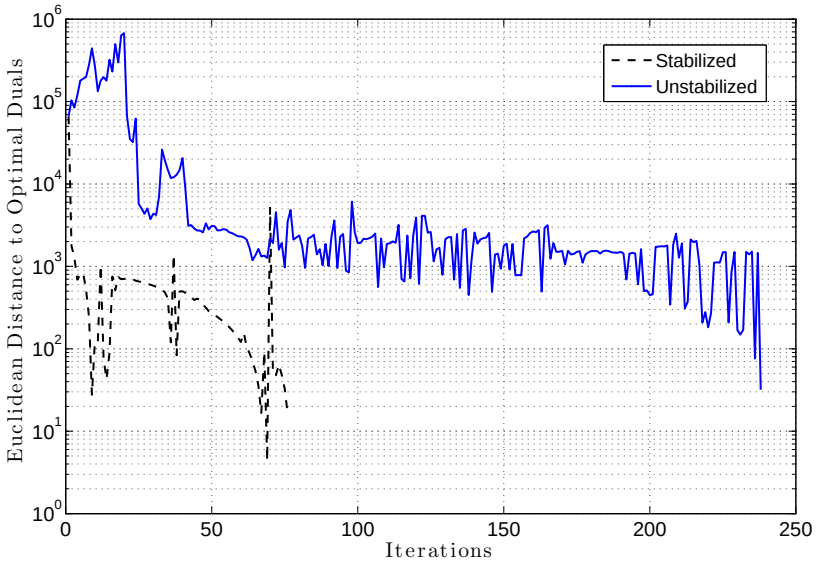



Figure 5: The progress of the dual variables over the course of *SD-SMS**.

* The $y-$axis is in log-scale, and the final iteration is omitted because a distance of zero cannot be represented in this scale.

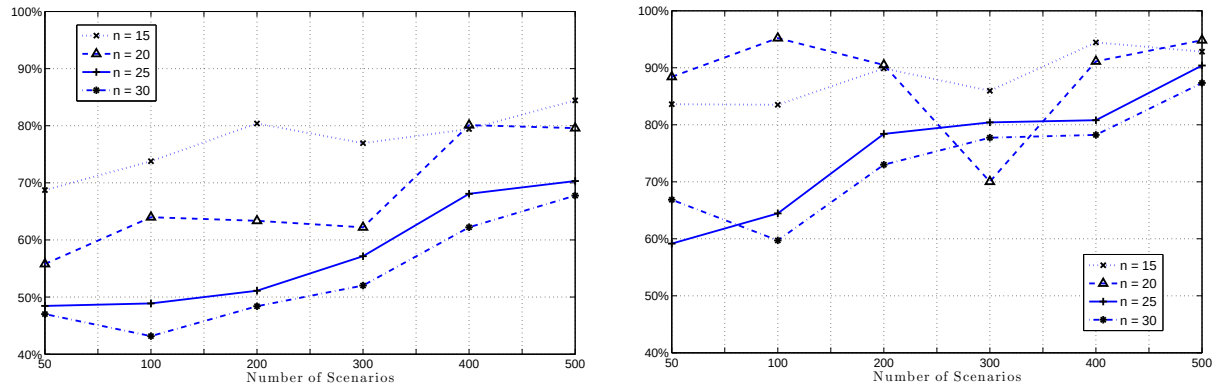
Without the two algorithmic features discussed so far in this section, *SD-SMS* demonstrates a lackluster performance both in terms of solution time and quality and does not establish itself as a viable solution method even for small instances. The next two algorithmic features, however, are geared toward enhancing the scalability of the algorithm to large instances. Table 6 presents the average relative increases in the elapsed times of *SD-SMS* for the same instances in Table 5 if we insist that the subproblems are always solved to optimality and no suboptimal cuts are added to the *RMP*. The results indicate that the suboptimal cuts contribute to the computational performance of the algorithm significantly, and the relative improvement becomes more evident as the number of jobs and the number of scenarios increase, reaching as high as 377%.

| | **\|S\|** | | | | | | |
|---|---|---|---|---|---|---|---|
| **n** | **50** | **100** | **200** | **300** | **400** | **500** | **Average** |
| **15** | 4.9 | 18.3 | 11.5 | 13.8 | 85.0 | 47.9 | **30.2** |
| **20** | 11.3 | 8.8 | 24.6 | 14.1 | 164.3 | 377.3 | **100.1** |
| **Average:** | **8.1** | **13.6** | **18.0** | **14.0** | **124.6** | **212.6** | **65.2** |

Table 6: Relative differences (%) in the solution times of *SD-SMS* when no suboptimal cuts are generated.

Next, we delve into the computational details of Algorithm 1 in Section 3.2 for solving the Lagrangian subproblems. First, we analyze the breakdown of the subproblems solved in each phase of Algorithm 1 and then assess the computational gain from using Algorithm 1 instead of attacking the subproblems directly by `CPLEX`. To complete the earlier task, we calculate the percentage of subproblems solved in each phase of Algorithm 1 for each *TT* instance in Section 5.3, except that we skip over the initial iterations with $\boldsymbol{u} = \mathbf{0}$ to avoid introducing a bias that favors Phase I. Recall that only the first phase of Algorithm 1 is executed if $\boldsymbol{u} = \mathbf{0}$. The average percentage of the instances solved in Phases I, II, and III turns out to be 46.2%, 17.6%,

and 36.1%, respectively. A further breakdown of these numbers is provided in Figure 6, where we report the average and maximum percentage of subproblems solved in the first two phases of Algorithm 1 grouped by $n$ and $|S|$. On the one hand, Figure 6 attests to a desired property of *SD-SMS* that it scales well with $|S|$ for a fixed $n$. That is, we generally observe that with increasing $|S|$ a higher percentage of the subproblems are solved in the computationally less expensive first two phases of Algorithm 1 without having to invoke `CPLEX`. On the other hand, the number of job processing sequences grows exponentially with the number of jobs and leads to a decline in the effectiveness of the rules in Phase I and the enumeration of the assignments in Phase II. This can partially be attributed to a drawback of our $K-$assignment implementation which evaluates a predetermined fixed number of assignments independent of $n$.




(a) Average percentage of subproblems solved in Phases I-II. (b) Maximum percentage of subproblems solved in Phases I-II.

Figure 6: Statistics on the percentage of subproblems solved without resorting to `CPLEX` in Phase III of Algorithm 1.

| | | **\|S\|** | | | | | | |
|---|---|---|---|---|---|---|---|---|
| | **n** | **50** | **100** | **200** | **300** | **400** | **500** | **Average** |
| ***TT*** | **15** | -13.3 | 14.4 | 75.5 | 81.8 | 67.5 | 84.3 | **51.7** |
| | **20** | -5.9 | 19.9 | 48.5 | 49.3 | 87.5 | 73.0 | **45.4** |
| | **25** | -11.1 | 5.0 | 23.4 | 21.0 | 41.5 | 52.9 | **22.1** |
| | **30** | -5.6 | -3.5 | 23.0 | 14.5 | 40.0 | 50.4 | **19.8** |
| **Average:** | | **-9.0** | **9.0** | **42.6** | **41.7** | **59.1** | **65.1** | **34.8** |
| ***TWT*** | **10** | 133.0 | 150.4 | 171.4 | 124.0 | 166.5 | 290.1 | **172.6** |
| | **15** | 960.6 | 645.2 | 289.1 | 239.7 | 123.7 | 191.7 | **408.3** |
| | **20** | 405.1 | 618.2 | 189.8 | 203.8 | 157.7 | 137.7 | **285.4** |
| **Average:** | | **499.6** | **471.3** | **216.8** | **189.2** | **149.3** | **206.5** | **288.8** |

Table 7: Relative differences (%) in the solution times of *SD-SMS* when the subproblems are solved directly by `CPLEX`.

Finally, we investigate the contribution of Algorithm 1 to reducing the solution times of *SD-SMS* under both performance measures $TT$ and $TWT$. To this end, we re-solve all instances in Tables 3 and 4 in Section 5.3 by *SD-SMS* and employ `CPLEX` to solve all subproblems directly. Clearly, *SD-SMS* does not necessarily take the same number of iterations to terminate when we change the subproblem solution algorithm. To account for this and ensure a meaningful comparison, we only compare the solution times over the first 40 iterations. For each instance, we compute the percentage change in the solution time with respect to the original value. The results are depicted in Table 7, where each figure denotes an average over ten instances for the corresponding $n$ and $|S|$ values. There are no obvious trends in the table; however, Algorithm 1 is clearly superior to employing `CPLEX` directly. Moreover, the computational savings for $TWT$ indicate that solving the mixed-integer formulation of the subproblems under this performance measure is significantly more challenging.

**6. Conclusions & Future Work** The key contribution of this work is a novel and generic modeling and solution framework for minimizing a widely popular risk measure known as value-at-risk in single-machine scheduling problems with uncertain parameters. Our solution methodology is based on a scenario decomposition obtained through Lagrangian relaxation and successfully provides high quality lower bounds and near-optimal feasible solutions in reasonable solution times for the very challenging set of problems under consideration. The strength of our solution algorithm is rooted in an efficient mix of state-of-the-art methodological and computational tools. Moreover, the solutions obtained from our risk-averse stochastic programming model are contrasted against those retrieved from the corresponding deterministic and risk-neutral models and the value of our approach for decision making purposes is illustrated.

As part of our ongoing work, we intend to extend the current framework to additional risk measures such as the conditional value-at-risk. A further avenue of research is to enhance and embed our scenario decomposition-based lower bounding scheme into an exact algorithm. In addition, recall that we restrict our attention to regular scheduling objectives total tardiness and total weighted tardiness in the computational part of the current study. An interesting question from a computational perspective is whether the performance of the current framework would carry over to non-regular scheduling objectives, such as the total weighted earliness and tardiness.

**Acknowledgment.** This work has been partially supported by The Scientific and Technological Research Council of Turkey (TUBITAK) under grant 112M864. We are sincerely grateful to Pascoal et al. and Tanaka et al. for providing us with the C source codes of the $K-$assignment algorithm and the `SiPS` C++ libraries, respectively. We would also like to thank Birce Tezel for her help with an earlier version of this work.

## References

Akker, M. and Hoogeveen, H. (2008). Minimizing the number of late jobs in a stochastic setting using a chance constraint. *Journal of Scheduling*, 11(1):59–69.

Alonso-Ayuso, A., Escudero, L., no, M. O., and Pizarro, C. (2007). On a stochastic sequencing and scheduling problem. *Computers and Operations Research*, 34(9):2604–2624.

Aloulou, M. A. and Croce, F. D. (2008). Complexity of single machine scheduling problems under scenario-based uncertainty. *Operations Research Letters*, 36(3):338–342.

Artzner, P., Delbaen, F., Eber, J. M., and Heath, D. (1999). Coherent measures of risk. *Mathematical Finance*, 9(3):203–227.

Atakan, S., Tezel, B., Bülbül, K., and Noyan, N. (2011). Minimizing value-at-risk in the single-machine total weighted tardiness problem. In Proceedings of the 5th Multidisciplinary International Scheduling Conference on Scheduling: Theory & Applications (MISTA 2011), 9-11 August 2011, Phoenix, Arizona, USA, pages 215-229.

Baker, K. R. and Keller, B. (2010). Solving the single-machine sequencing problem using integer programming. *Computers & Industrial Engineering*, 59(4):730–735.

Ben Amor, H. M., Desrosiers, J., and Frangioni, A. (2009). On the choice of explicit stabilizing terms in column generation. *Discrete Applied Mathematics*, 157(6):1167–1184.

Birge, J. and Louveaux, F. (1997). *Introduction to stochastic programming*. Springer, New York.

Burkard, R., Dell'Amico, M., and Martello, S. (2009). *Assignment Problems*. Society for Industrial and Applied Mathematics, Philadelphia.

Carøe, C. and Tind, J. (1998). L-shaped decomposition of two-stage stochastic programs with integer recourse. *Mathematical Programming*, 83(1):451–464.

Carøe, C. C. and Schultz, R. (1999). Dual decomposition in stochastic integer programming. *Operations Research Letters*, 24(1-2):37–45.

Daniels, R. and Kouvelis, P. (1995). Robust scheduling to hedge against processing time uncertainty in

single-stage production. *Management Science*, 41(2):363–376.

Daniels, R. L. and Carrillo, J. (1997). $\beta$-robust scheduling for single-machine systems with uncertain processing times. *IIE Transactions*, 29(11):977–985.

de Farias, JR, I. R., Zhao, H., and Zhao, M. (2010). A family of inequalities valid for the robust single machine scheduling polyhedron. *Computers and Operations Research*, 37(9):1610–1614.

Dentcheva, D. (2006). Optimization models with probabilistic constraints. In Calafiore, G. and Dabbene, F., editors, *Probabilistic and Randomized Methods for Design Under Uncertainty*, pages 49–97. Springer London.

Du, J. and Leung, J. Y.-T. (1990). Minimizing total tardiness on one machine is NP-hard. *Mathematics of Operations Research*, 15(3):483–495.

Frangioni, A. (2005). About Lagrangian methods in integer optimization. *Annals of Operations Research*, 139(1):163–193.

Gaivoronski, A. A. and Pflug, G. C. (2005). Value-at-risk in portfolio optimization: Properties and computational approach. *Journal of Risk*, 7(2):1–31.

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

Gutjahr, W. J., Hellmayr, A., and Pflug, G. C. (1999). Optimal stochastic single-machine-tardiness scheduling by stochastic branch-and-bound. *European Journal of Operational Research*, 117(2):396–413.

Helmberg, C. (2011). The ConicBundle library for convex optimization. Last viewed on August 16, 2013.

Jörnsten, K. O., Näsberg, M., and Smeds, P. A. (1985). *Variable splitting: A New Lagrangean Relaxation Approach to Some Mathematical Programming Models*. Linköping University, Department of Mathematics, Sweden.

Kallehauge, B., Larsen, J., and Madsen, O. B. G. (2006). Lagrangian duality applied to the vehicle routing problem with time windows. *Computers & Operations Research*, 33(5):1464–1487.

Kanet, J. and Sridharan, V. (2000). Scheduling with inserted idle time: problem taxonomy and literature review. *Operations Research*, 48(1):99–110.

Karp, R. M. (1972). Reducibility among combinatorial problems. In Miller, R. E., Thatcher, J. W., and Bohlinger, J. D., editors, *Complexity of Computer Computations*, The IBM Research Symposia Series, pages 85–103. Springer US.

Kasperski, A. (2005). Minimizing maximal regret in the single machine sequencing problem with maximum lateness criterion. *Operations Research Letters*, 33(4):431–436.

Kasperski, A., Kurpisz, A., and Zieliński, P. (2012). Approximating a two-machine flow shop scheduling under discrete scenario uncertainty. *European Journal of Operational Research*, 217(1):36–43.

Kataoka, S. (1963). A stochastic programming model. *Econometrica*, 31(1/2):181–196.

Keha, A. B., Khowala, K., and Fowler, J. W. (2009). Mixed integer programming formulations for single machine scheduling problems. *Computers and Industrial Engineering*, 56(1):357–367.

Klein Haneveld, W. K. and van der Vlerk, M. H. (1999). Stochastic integer programming: General models and algorithms. *Annals of Operations Research*, 85:39–57.

Laporte, G. and Louveaux, F. (1993). The integer L-shaped method for stochastic integer programs with complete recourse. *Operations Research Letters*, 13(3):133–142.

Larsen, N., Mausser, H., and Uryasev, S. (2002). Algorithms for optimization of value-at-risk. In Pardalos, P. M. and Tsitsiringos, V. K., editors, *Financial Engineering, E-commerce and Supply Chain*, volume 70 of *Applied Optimization*, pages 19–46. Springer US.

Lenstra, J. K., Rinnooy Kan, A. H. G., and Brucker, P. (1977). Complexity of machine scheduling problems. *Annals of Discrete Mathematics*, 1:343–362.

Louveaux, F. V. and Schultz, R. (2003). Stochastic integer programming. In Ruszczyński, A. and Shapiro, A., editors, *Stochastic Programming*, volume 10 of *Handbooks in Operations Research and Management Science*,

pages 213–266. Elsevier.

Lu, C.-C., Lin, S.-W., and Ying, K.-C. (2012). Robust scheduling on a single machine to minimize total flow time. *Computers & Operations Research*, 39(7):1682–1691.

Marsten, R. E., Hogan, W. W., and Blankenship, J. W. (1975). The BOXSTEP method for large-scale optimization. *Operations Research*, 23(3):389–405.

Ogryczak, W. and Ruszczyński, A. (2002). Dual stochastic dominance and related mean-risks models. *SIAM Journal on Optimization*, 13(2):60–78.

Pang, J. and Leyffer, S. (2004). On the global minimization of the value-at-risk. *Optimization Methods and Software*, 19(5):611–631.

Pascoal, M., Captivo, M. E., and Clímaco, J. (2003). A note on a new variant of Murty's ranking assignments algorithm. *4OR*, 255(1):243–255.

Pflug, G. C. and Römisch, W. (2007). *Modeling, managing and measuring risk*. World Scientific Publishing, Singapore.

Pinedo, M. (2008). *Scheduling: Theory, Algorithms, and Systems*. Springer, 3rd edition.

Pinedo, M. and Singer, M. (1999). A shifting bottleneck heuristic for minimizing the total weighted tardiness in a job shop. *Naval Research Logistics*, 46(1):1–17.

Potts, C. and van Wassenhove, L. (1982). A decomposition algorithm for the single machine total tardiness problem. *Operations Research Letters*, 1(5):177–181.

Prékopa, A. (1995). *Stochastic Programming*. Kluwer Academic, Dordrecht, Boston.

Rockafellar, R. T. and Wets, R. J.-B. (1991). Scenarios and policy aggregation in optimization under uncertainty. *Mathematics of Operations Research*, 16(1):119–147.

Schultz, R. and Tiedemann, S. (2003). Risk aversion via excess probabilities in stochastic programs with mixed-integer recourse. *SIAM Journal on Optimization*, 14(1):115–138.

Schultz, R. and Tiedemann, S. (2006). Conditional value-at-risk in stochastic programs with mixed-integer recourse. *Mathematical Programming*, 105(2):365–386.

Sen, S. (2005). Algorithms for stochastic mixed-integer programming models. In Aardal, K., Nemhauser, G. L., and Weismantel, R., editors, *Discrete Optimization*, volume 12 of *Handbooks in Operations Research and Management Science*, pages 515–558. Elsevier.

Shapiro, A., Dentcheva, D., and Ruszczyński, A. (2009). *Lectures on Stochastic Programming: Modeling and Theory*, volume 9. SIAM.

Srinivasan, V. (1971). A hybrid algorithm for the one machine sequencing problem to minimize total tardiness. *Naval Research Logistics Quarterly*, 18(3):317–327.

Tanaka, S., Fujikuma, S., and Araki, M. (2009). An exact algorithm for single-machine scheduling without machine idle time. *Journal of Scheduling*, 12(6):575–593.

van de Panne, C. and Popp, W. (1963). Minimum cost cattle feed under probabilistic protein constraints. *Management Science*, 9(3):405–430.

van der Vlerk, M. H. (1996–2007). Stochastic integer programming bibliography. World Wide Web, http://www.eco.rug.nl/mally/biblio/sip.html.

Wolsey, L. A. (1998). *Integer Programming*. John Wiley & Sons, New York.

Wozabal, D. (2012). Value-at-risk optimization using the difference of convex algorithm. *OR Spectrum*, 34(4):861–883.

Wozabal, D., Hochreiter, R., and Pflug, G. C. (2010). A difference of convex formulation of value-at-risk constrained optimization. *Optimization*, 59(3):377–400.

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

Yang, J. and Yu, G. (2002). On the robust single machine scheduling problem. *Journal of Combinatorial Optimization*, 6(1):17–33.