Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence (AAAI-16)

A Proactive Sampling Approach to Project Scheduling under Uncertainty

Pradeep Varakantham, Na Fu and **Hoong Chuin Lau**
School of Information Systems, Singapore Management University
80 Stamford Road, Singapore 178902
{pradeepv, nafu, hclau}@smu.edu.sg

Abstract

Uncertainty in activity durations is a key characteristic of many real world scheduling problems in manufacturing, logistics and project management. RCPSP/max with durational uncertainty is a general model that can be used to represent durational uncertainty in a wide variety of scheduling problems where there exist resource constraints. However, computing schedules or execution strategies for RCPSP/max with durational uncertainty is NP-hard and hence we focus on providing approximation methods in this paper. We provide a principled approximation approach based on Sample Average Approximation (SAA) to compute proactive schedules for RCPSP/max with durational uncertainty. We further contribute an extension to SAA for improving scalability significantly without sacrificing on solution quality. Not only is our approach able to compute schedules at comparable runtimes as existing approaches, it also provides lower α-quantile makespan (also referred to as α-robust makespan) values than the best known approach on benchmark problems from the literature.

Introduction

In most manufacturing, logistics and project management problems, activity durations are uncertain because of commonly occurring events such as human error, manpower unavailability and weather changes. Hence, accounting for uncertainty in scheduling problems is practically important. There have been proactive (compute offline schedules or strategies), reactive (provide online decisions on next activity to be scheduled) and proactive-reactive combination approaches developed in the literature to solve scheduling problems with durational uncertainty. We refer the readers to (Beck and Wilson 2007; Fu et al. 2012; Davenport, Gefflot, and Beck 2001; Herroelen and Leus 2005; Bidot et al. 2009; Lombardi and Milano 2009; Cimatti, Micheli, and Roveri 2015) for a survey of existing work in scheduling under uncertainty.

In this paper, we are specifically interested in developing proactive approaches for resource constrained scheduling problems where there is durational uncertainty. Our research is complementary to other works that have focussed on reactive approaches for scheduling problems. We employ RCPSP/max with durational uncertainty model provided by (Fu et al. 2012). Jobshop Scheduling Problems (JSP) with uncertainty (Beck and Wilson 2007) and RCPSP with uncertainty (Lamas and Demeulemeester 2015) are subsumed by RCPSP/max with durational uncertainty. The presence of maximum time lags in RCPSP/max makes the problem significantly challenging, as demonstrated in our experimental results. Our approaches are similar to scenario-based optimization (Kouvelis, Daniels, and Vairaktarakis 2000; Mulvey, Vanderbei, and Zenios 1995) that have previously been applied to sub-problems of the RCPSP/max with uncertainty model. The current best approach for solving RCPSP/max is by (Fu et al. 2012), where local search methods are employed. We provide principled stochastic approximation approaches based on Sample Average Approximation (SAA).

Similar to a few existing proactive approaches for considering uncertainty in scheduling (Fu et al. 2012; Beck and Wilson 2007), we consider a risk aware objective in solving RCPSP/max with durational uncertainty. Specifically, given a risk parameter α, we minimize the α-quantile[1] on the makespan distributions[2] . We refer to this α-quantile also as the α-robust makespan.

Concretely, we make the following key contributions in this paper. First, we provide an exact MIP formulation for solving deterministic RCPSP/max. A key advantage of this deterministic formulation is that it helps provide a principled approximation approach for solving RCPSP/max with durational uncertainty. Second, we extend the MIP formulation with Sample Average Approximation to represent uncertainty in activity durations. Third, we provide a scalable extension to SAA in order to significantly improve the scalability of the MIP formulation with SAA. Because of this extension, existing algorithms (Schutt et al. 2013) for solving deterministic RCPSP/max become relevant and complementary. Finally, we provide an extensive experimental evaluation on benchmark problems from literature and compare against the best known approach to solve RCPSP/max with uncertainty.

[1] α-quantile for a probability distribution is the minimum value (outcome) amongst all those values (outcomes) whose cumulative probability exceeds α.

[2] Since the duration is uncertain, makespan is no longer a single value but a probability distribution.

RCPSP/max with Durational Uncertainty

A deterministic RCPSP/max consists of N activities $\{a_1, \cdots, a_N\}$ and K types of renewable resources limited by capacity C_k, where $k = 1, \cdots, K$. Each activity a_i should be executed for a time duration d_i without preemption. Furthermore, a_i requires r_{ik} units of type k resource during execution. In addition, two dummy activities a_0 and a_{N+1} with zero durations are introduced to represent the beginning and the completion of the project, respectively.

A schedule $\mathbf{s} = (s_1, \cdots, s_N)$ is an assignment of start times to all activities, where s_i represents the start time of activity a_i. The goal of the deterministic RCPSP/max is to determine a time (temporal constraints) and resource (resource capacity constraints) feasible schedule, such that the project makespan, which is defined as the start time of the final dummy activity a_{N+1}, is minimized. The two types of constraints present in an RCPSP/max instance are:

- Generalized Temporal Constraints ($\mathbf{s} \leq \mathbf{T}$): They specify the minimal or maximal time lags between pairs of activities. There exist four types of generalized temporal constraints: start-start, start-finish, finish-start and finish-finish. In the deterministic setting, the different types of constraints can be represented in standardized start-start form by using the transformation rules (Bartusch, Mohring, and Radermacher 1988).

$$T_{i,j}^{min} \leq s_j - s_i \leq T_{i,j}^{max}, \forall i, j$$

- Resource Capacity Constraints ($\mathbf{r} \leq \mathbf{C}$): These ensure that at any time during execution of schedule, the number of resources in use does not exceed the capacity.

$$\sum\nolimits_{\{i|s_i \leq t \leq s_i + d_i\}} r_{ik} \leq C_k, \forall t, k$$

In most real world problems, activity durations, d_i are uncertain. To represent this uncertainty, we assume that duration of an activity, a_i is a random variable, $\tilde{d}_i$. In the deterministic setting, makespan can be used to evaluate the performance of a schedule. However, when uncertainty is involved, the makespan itself becomes a random variable. Similar to existing work (Fu et al. 2012), we employ $\alpha-$robust makespan as the objective. We focus on computing a start time schedule, $\mathbf{s}$ instead of an execution strategy[3].

As indicated earlier, α-robust makespan for a schedule is the α-quantile value for the makespan distribution corresponding to the schedule. Formally, given α ($0 \leq \alpha \leq 1$), our goal is to find a start time schedule with the least α-robust makespan. Formally, ρ is an α-robust makespan for a schedule $\mathbf{s}$ if

$$Pr(\rho_{\mathbf{s}}(\tilde{\mathbf{d}}) \geq \rho \ \wedge \ \mathbf{s} \leq \mathbf{T} \ \wedge \ \mathbf{r} \leq \mathbf{C})) \leq \alpha$$

where $\rho_{\mathbf{s}}(\tilde{\mathbf{d}})$ is a random variable that denotes the makespan for $\mathbf{s}$ given uncertain durations of activities, $\tilde{\mathbf{d}}$. Formally, the least value of α-robust makespan is computed by solving the abstract optimization model in Table 1.

[3]More discussion on this choice in "Discussion" section

$$\begin{aligned} &\min_{\mathbf{s}} \ \ \rho \\ &\textbf{s.t} \ \ Pr(\rho_{\mathbf{s}}(\tilde{\mathbf{d}}) \geq \rho \ \wedge \ \mathbf{s} \leq \mathbf{T} \ \wedge \ \mathbf{r} \leq \mathbf{C})) \leq \alpha \end{aligned}$$

Table 1: Abstract Optimization Formulation

Solving the Deterministic RCPSP/max

In this section, we make the abstract optimization formulation presented in previous section more concrete. Specifically, we develop an MILP (Mixed Integer Linear Program) formulation for RCPSP/max that is amenable to sampling based extensions for addressing the stochasticity. In contrast to the formulations for modeling RCPSP using variables indexed by sequence and flow (Artigues, Michelon, and Reusser 2003), by starting and ending events (Koné et al. 2011), or using lazy clause generation to model cumulative constraints by decomposition (Schutt et al. 2009), we propose a formulation based on identifying resource usage overlaps. A key challenge in providing an optimization model is the efficient enforcement of the resource capacity constraint at all times. We employ the following observation to provide an efficient constraint that prevents resource capacity violation at all time points:

Observation 1. *Given a schedule $\mathbf{s}$, if we ensure no violation of resource capacity happens at the starting times of all activities, then no resource capacity constraint is violated at any time throughout the execution of the schedule.*

Observation 1 is justified as resource usage increases only at activity start times. We now develop the optimization formulation that enforces characteristics of a makespan minimizing schedule, $\mathcal{S}$.

Temporal Feasibility: We employ the following two constraints to enforce the minimum and maximum time lags:

$$s_j - s_i \geq T_{ij}^{min}, \qquad \forall (i,j) \in T^{min} \qquad (1)$$

$$s_j - s_i \leq T_{ij}^{max}, \qquad \forall (i,j) \in T^{max} \qquad (2)$$

where s_i and s_j correspond to the starting times of activity a_i and activity a_j respectively.

Note that this temporal feasibility is defined based on only the start-start temporal constraints. In the Discussion section, we will present how the other temporal constraints are handled in our stochastic setting.

Resource Feasibility: At any time during the execution of schedule $\mathbf{s}$, the consumption of any resource type k cannot exceed its capacity. Based on Observation 1, this can be enforced by:

(1) Identifying all activities a_j that are executing when activity a_i is started; and then

(2) Computing the overall resource consumption for every resource type k at the starting time of a_i and enforcing the capacity constraint.

First, we employ two sets of binary variables, δ_{ij}^1 and δ_{ij}^2 to determine if a_j is executing when a_i is started. If both δ_{ij}^1

and δ^2_{ij} are set to 1, then a_j is executing when a_i started. We add the following constraints to set δ^1_{ij} and δ^2_{ij}:

$$\delta^1_{ij} \geq \frac{s_i - s_j}{M} + \frac{1}{M} \; ; \; \delta^2_{ij} \geq \frac{s_j + d_j - s_i}{M}$$

where M in the above constraints is a large positive constant. One possible metric of M is the sum of all activity durations. We add the term $\frac{1}{M}$ in first constraint to account for the boundary case where $s_i = s_j$. To better understand how the above constraints work, we consider all three scenarios possible where ϵ represents a small positive value:

- Case 1: Activity a_i starts after activity a_j is completed.

$$\delta^1_{ij} \geq +\epsilon \implies \delta^1_{ij} = 1 \; ; \; \delta^2_{ij} \geq -\epsilon \implies \delta^2_{ij} = 0 \textbf{ or } 1$$

- Case 2: a_i starts during the execution of activity a_j.

$$\delta^1_{ij} \geq +\epsilon \implies \delta^1_{ij} = 1 \; ; \; \delta^2_{ij} \geq +\epsilon \implies \delta^2_{ij} = 1$$

- Case 3: a_i starts before activity a_j starts.

$$\delta^1_{ij} \geq -\epsilon \implies \delta^1_{ij} = 0 \textbf{ or } 1 \; ; \; \delta^2_{ij} \geq +\epsilon \implies \delta^2_{ij} = 1$$

Second, we use these sets of indicator variables δ^1 and δ^2 to compute resource consumption by an activity at the start of a specific activity. Formally, let res^i_{jk} denote the total resource of type k consumed by activity a_j at the start of activity a_i . If δ^1_{ij} and δ^2_{ij} are equal to 1, then res^i_{jk} is equal to r_{jk} (requirement of resource type k for a_j) and for any other case is equal to 0, as a_j is not being executed at the start of a_i. The following constraints ensure that the above logical condition is satisfied:

$$res^i_{jk} \leq \delta^1_{ij} r_{jk} \; ; \; res^i_{jk} \leq \delta^2_{ij} r_{jk}$$
$$res^i_{jk} \geq r_{jk} - (2 - \delta^1_{ij} - \delta^2_{ij}) \cdot \hat{M}$$

where $\hat{M}$ is a large number. Given values of res^i_{jk}, resource capacity can then be enforced using the following constraint:

$$r_{ik} + \sum_{j:j \neq i} res^i_{jk} \leq C_k$$

Makespan Minimization: Since the starting time of the sink node, s_{N+1} corresponds to the makespan of the given start time schedule, we use "$\min s_{N+1}$" as the objective. The overall optimization model for solving the RCPSP/max is shown in Table 2.

min	s_{N+1}	
$s.t.$	$s_j - s_i \geq T^{min}_{ij}, \quad \forall (i,j) \in T^{min}$	(3)
	$s_j - s_i \leq T^{max}_{ij}, \quad \forall (i,j) \in T^{max}$	(4)
	$s_0 = 0$	(5)
	$\delta^1_{ij} \geq \frac{s_i - s_j + 1}{M}, \quad \forall i,j$	(6)
	$\delta^2_{ij} \geq \frac{s_j + d_j - s_i}{M}, \quad \forall i,j$	(7)
	$res^i_{jk} \leq \delta^1_{ij} r_{jk}, \quad \forall i,j,k$	(8)
	$res^i_{jk} \leq \delta^2_{ij} r_{jk}, \quad \forall i,j,k$	(9)
	$res^i_{jk} \geq r_{jk} - (2 - \delta^1_{ij} - \delta^2_{ij}) \cdot \hat{M}, \quad \forall i,j,k$	(10)
	$r_{ik} + \sum_{j \neq i} res^i_{jk} \leq C_k, \quad \forall i,k$	(11)
	$res^i_{jk} \geq 0, \quad \forall i,j,k$	(12)

Table 2: SOLVERCPSPMAX($\{d_j\}_{j \leq N}$)

Solving RCPSP/max with Durational Uncertainty

In this section, we introduce a sampling based approach to solve RCPSP/max with durational uncertainty. More specifically, we employ duration samples to operationalise the abstract optimization model of Table 2. The approach presented in this section is not dependent on the specific distribution employed to represent durational uncertainty for activities. In fact, as long as there is a simulator that can generate duration samples for the activities, our methods are applicable.

The original problem is to compute the start time schedule that has the least α-robust makespan for RCPSP/max with uncertain activity durations. It can be approximated by computing a start time schedule that has the least γ-robust makespan for the discrete set of duration samples, ξ, of the durational uncertainty distribution. Since, the latter computation is focused on a limited set, in the worst case, the violations when the entire set is considered are higher. Hence, $\gamma \leq \alpha$. As we show in experimental results, we were able to identify that for our problems, $\alpha - \gamma = 0.1$ yields good results.

Formally, a duration sample q is defined as: $\xi^q = \{d^q_1, d^q_2, \cdots, d^q_n\}$, where d^q_i represents the duration of a_i in sample q and a set of duration samples is defined as $\xi = \{\xi^1, \xi^2, \cdots, \xi^{\mathcal{Q}}\}$ with $\mathcal{Q}$ samples[4].

In order to compute the start time schedule, **s** with the least α-robust makespan for a given sample set ξ with $\mathcal{Q}$ samples, we formulate an optimization model that extends on the optimization model provided in SOLVERCPSPMAX(). **s** with least α-robust makespan has to satisfy the following characteristics:

(1) Temporal constraints remain the same as for the deterministic case because of start-start time lags. Essentially, start-start time lags allow for the start time schedule to be independent of duration samples[5]. Also, as we explain in "Discussion" section, for other types of temporal constraints (ex: end-start), we have temporal constraints handled in a different manner and also contribute to temporary constraint violations.

(2) For a schedule **s**, resource constraint feasibility has to

[4]As we assume the existence of simulators, generating the set of duration samples is trivial.

[5]It should be noted that in our formulation, duration samples primarily affect resource capacity constraints.

$$\min \quad s_{N+1}$$
$$s.t. \quad Constraints\ (3)-(5)$$
$$\delta^1_{ij} \geq \frac{s_i - s_j + 1}{M}, \quad \forall i,j \tag{13}$$
$$\delta^{2,q}_{ij} \geq \frac{s_j + d^q_j - s_i}{M}, \quad \forall i,j,q \tag{14}$$
$$res^{i,q}_{jk} \leq \delta^1_{ij} r_{jk}, \quad \forall i,j,k,q \tag{15}$$
$$res^{i,q}_{jk} \leq \delta^{2,q}_{ij} r_{jk}, \quad \forall i,j,k,q \tag{16}$$
$$res^{i,q}_{jk} \geq r_{jk} - (2 - \delta^1_{ij} - \delta^{2,q}_{ij}) \cdot \hat{M}, \quad \forall i,j,k,q \tag{17}$$
$$r_{ik} + \sum_{j \neq i} res^{i,q}_{jk} \leq C_k + (1 - z^q) \cdot M, \quad \forall i,k,q \tag{18}$$
$$\sum_q z^q >= (1-\gamma) \cdot \mathcal{Q} \tag{19}$$

Table 3: SOLVERCPSPUNC-SAA($\{d^q_j\}^{q \leq \mathcal{Q}}_{j \leq N}$)

be examined for each sample of durations. We use the same resource constraints as those used in the deterministic case, with the exception that some of the variables are indexed using the sample number q. More specifically, the δ^1_{ij} variables are not indexed by the sample q as their assignment is only dependent on starting times of activities and not on durations of activities. On the other hand, δ^2 and res^i_{jk} variables are indexed by the sample q. Updated constraints are:

$$\delta^1_{ij} >= \frac{s_j - s_i}{M} \qquad \forall i,j,q$$
$$\delta^{2,q}_{ij} >= \frac{s_i + d^q_i - s_j}{M} \qquad \forall i,j,q$$
$$res^{i,q}_{jk} \leq \delta^1_{ij} r_{jk} \ \ ; \ \ res^{i,q}_{jk} \leq \delta^{2,q}_{ij} r_{jk} \qquad \forall i,j,k,q$$
$$res^{i,q}_{jk} \geq r_{jk} - (2 - \delta^1_{ij} - \delta^{2,q}_{ij}) \cdot \hat{M} \qquad \forall i,j,k,q$$

Overall, if both δ^1_{ij} and $\delta^{2,q}_{ij}$ are 1, then $res^{i,q}_{jk} = r_{jk}$.

(3) Schedule **s** should be such that the proportion of samples that violate resource capacity constraints at any time point is less than $\gamma * \mathcal{Q}$. We introduce a binary variable z^q associated with each sample q and z^q is set to 1 if q is resource feasible at all time points. Following constraints are used to set z^q:

$$\sum_{q \in \mathcal{Q}} z^q \geq (1-\gamma) \cdot \mathcal{Q}$$
$$r_{ik} + \sum_{j \neq i} res^{i,q}_{jk} \leq C_k + (1 - z^q) \cdot M \qquad \forall i,k,q$$

Intuitively, the first constraint enforces that at most $\gamma \cdot \mathcal{Q}$ of the samples can violate resource capacity constraints. The second constraint sets z^q to 0 only if the resource requirement for resource type k exceeds capacity C_k at the start of any activity, a_i with durations from sample ξ_q.
Combining the constraints, we formulate the optimization model SOLVERCPSPMAXUNC-SAA as shown in Tabl 3.

We refer to this approach of solving RCPSP/Max with durational uncertainty as SORU (SAA Optimization for solving RCPSP/Max under Uncertainty). The objective function of the SORU optimization model (i.e., s_{N+1}) is the γ-robust makespan.

Heuristic Approximation

The scalability of the SORU approach is determined by the problem type (tightness of scheduling constraints) and the number of samples employed. It turns out that the number of samples required for obtaining good schedules in complex and large scale problem instances increases quickly and can be a severe bottleneck in the scalability of SORU.

One key insight that we provide in this paper, which is a general purpose extension to SAA, is to summarise the samples and make use of only a few summary samples. More concretely, in this work, we summarise the duration sample set ξ by using one γ-percentile duration sample. We refer to this as SORU-H (SORU Heuristic) that trades off guarantees on solution quality for scalability and has the same complexity as solving the deterministic RCPSP/max.

Formally, given a duration sample set ξ and risk factor γ, we create a deterministic RCPSP/max $\hat{\mathcal{R}}$ with activity durations given by:

$$\hat{d}_i = PERCENTILE(\mathbf{d}_i, 1-\gamma), \quad \forall i \leq N \tag{20}$$

where $\mathbf{d}_i = \{d^1_i, \cdots, d^q_i, \cdots d^Q_i\}$ and d^q_i refers to the duration for activity a_i in duration sample ξ^q. The above expression computes a $(1-\gamma)$th percentile of activity duration from all durations in the set $\mathbf{d}_i$. Equation 20 entails that at least $(1-\gamma) \cdot Q$ samples in ξ will have lower durations than $\hat{d}_i$ for every activity i. The overall algorithm for SORU-H can therefore be summarized as SOLVERCPSPMAX($\{\hat{d}_i\}_{i \leq N}$).

Let the schedule obtained by solving the deterministic RCPSP/max instance be given by $\hat{\mathbf{s}} = (\hat{s}_1, \hat{s}_2, \cdots, \hat{s}_n)$. Let $\hat{\delta}^1_{ij}$ and $\hat{\delta}^2_{ij}$ be the set of variables used to check if a_j overlaps with a_i's starting time using the duration sample (Equation 20) selected by SORU-H.

Observation 2. *For a given schedule $\hat{\mathbf{s}}$, given two samples ξ^{q1} and ξ^{q2}: $d^{q1}_i \geq d^{q2}_i \implies \delta^{2,q1}_{ij} \geq \delta^{2,q2}_{ij}$*

Intuitively, for a given schedule, if the duration is higher for an activity a_i in a sample, then it only increases the chance that a_i would overlap in resource usage with a_j. Observation 2 provides an intuition for why SORU-H generates α-robust makespan for sample set ξ effectively.

Experimental Evaluation

To demonstrate the utility of our proposed approaches, we compare against the best known work in the literature on generating α-robust makespan for RCPSP/max with durational uncertainty, referred to as FLPV (Fu et al. 2012). FLPV employs Partial Ordered Schedule to represent solutions, and our approaches, SORU and SORU-H employ a start time schedule. Also, while FLPV uses local search, SORU and SORU-H use mixed integer optimization. Even with these differences in solution representation and approach, since both FLPV and our two approaches (SORU

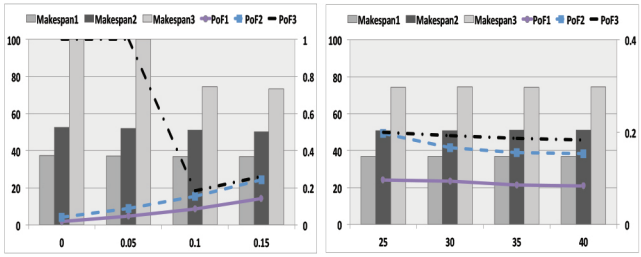

Figure 1: SORU performance: (a) Gamma; (b) Samples

and SORU-H) take as input an RCPSP/max with durational uncertainty instance and output α-robust makespan, they are directly comparable.

Experimental Setup

In FLPV, there are constraint violations with respect to maximum time lags due to durational uncertainty. In our case however, we need not deal with such maximum time lag violations since the start time schedule derived by our model already satisfy these constraints. On the other hand, our approaches need to handle resource capacity violations. In order to perform a fair comparison of these methods, we will count the number of violations (whether they are temporal or resource constraints). For both approaches, we define a common metric, namely the Probability of Failure (PoF), as the ratio of number of violations (obtained by evaluating the solution on duration instantiations) to the total number of duration instantiations (1000). In addition, to compare the quality of solution, we also measure the respecitve α-robust makespan (referred to also as α-RM in this section).

The problem instances considered in our experiments are obtained by extending the benchmark sets J10, J20 and J30 for RCPSP/max, as specified in PSPLib (Kolisch and Sprecher 1996). Instances in J10, J20 and J30 have 10, 20 and 30 activities respectively and each of the three sets have 270 instances with each instance considering 5 resources. Similar to (Fu et al. 2012), we also assume the duration of each activity is normally distributed with mean corresponding to the deterministic duration of instances given in the benchmark problems, and standard deviation, σ is taken from the set $\{0.1, 0.5, 1\}$. The reported α-RM for each instance is obtained by aggregating solutions from 10 runs.

SORU and SORU-H compute the α-robust makespan using a set of duration samples. Since the sample based techniques focus only on a limited set of samples, the probability of constraint violations when the entire uncertainty set is considered can only be higher in the worst case. Thus, to calculate α-robust makespan, we adopted a lower value of risk $\gamma \leq \alpha$ in the optimization models of SORU and SORU-H. This lower value of risk was computed through preliminary experiments like the one in Figure 1(a). We also did a similar experiment to identify the right number of samples while generating the schedule.

Result	FLPV vs SORU	SORU vs SORU-H
PoF1 $\leq \alpha \wedge$ PoF2 $\leq \alpha \wedge$ α-RM1 $\leq \alpha$-RM2	2.96%	11.48%
PoF1 $\leq \alpha \wedge$ PoF2 $\leq \alpha \wedge$ α-RM1 $> \alpha$-RM2	28.14%	37.04%
PoF1 $\leq \alpha \wedge$ PoF2 $> \alpha$	3.7%	5.93%
PoF1 $> \alpha \wedge$ PoF2 $\leq \alpha$	23.33%	0.37%
PoF1 $> \alpha \wedge$ PoF2 $> \alpha$	41.85%	45.19%

Table 4: Comparison of FLPV, SORU and SORU-H on J10 dataset for $\alpha = 0.2$. For results of second column, PoF1, α-RM1 correspond to FLPV and PoF2, α-RM2 correspond to SORU. For results of third column, PoF1, α-RM1 correspond to SORU and PoF2, α-RM2 correspond to SORU-H.

Results

In Figure 1 (a), we varied the input risk parameter, γ for a given actual risk parameter $\alpha = 0.2$, standard deviation, $\sigma = 0.5$ and tested the performance of SORU-H on three representative instances[6]. X-axis denotes the gamma, primary Y-axis denotes the α-robust makespan value and secondary Y-axis denotes PoF. When γ value is low (0 and 0.05), for one instance, optimization model becomes tightly constrained resulting in an infeasible solution. For other instances, PoF is much lower than desired (α = 0.2). When γ value is 0.15, for instance 2 and 3, PoF is higher than the α value. For γ = 0.1, for all 3 instances, the PoF was lower than the desired α value. We observed similar behavior for other parameter settings of σ and on other instances. Based on these initial tests and settings employed in prior work on SAA (Pagnoncelli, Ahmed, and Shapiro 2009; Varakantham and Kumar 2013) in other domains, we employed $\alpha - \gamma = 0.1$.

Unlike FLPV, SORU and SORU-H rely on samples to generate α-robust makespan and hence we consider number of samples, $\mathcal{Q}$ from the set: $\{25, 30, 35, 40\}$. It should be noted that $\mathcal{Q}$ is only to generate a schedule and not for evaluating the PoF. To determine the right number of samples, we ran preliminary experiments on multiple different problem instances. We show results on three representative instances in Figure 1(b). X-axis denotes the number of samples, primary Y-axis denotes the α-robust makespan value and secondary Y-axis denotes the PoF. For all three instances, the change in makespan is insignificant, however, PoF decreases as samples are increased until 35 and roughly stays the same for 40 samples and beyond (tested until 60 samples). This was observed for other settings of σ and other instances.

Unless otherwise specified, the default parameter values are $\sigma = 0.5$, $\alpha = 0.2$, $\gamma = 0.1$ and $\mathcal{Q} = 35$. We com-

[6]For purposes of exposition and to focus on the key aspects of the result, we consider three representative instances in this graph and in the graph considering variation in number of samples. However, we did test for both on 20 random instances.

pare our algorithms SORU, SORU-H against the best known solver for RCPSP/max with duration uncertainty in the literature, i.e., the FLPV approach[7] (Fu et al. 2012) on the three problems sets, J10, J20 and J30.

SORU was able to obtain solutions within 5 minutes for every one instance in J10 and 2 hrs for the J20 instances. However, for J30, we were unable to get optimal solutions for certain instances in the cut-off limit of 3 hrs. On the other hand, SORU-H was able to generate solutions for J10 instance within half of a second, J20 instances within 10 seconds and J30 instances within 10 minutes on average. Local search approaches of FLPV were able to finish in a maximum of a minute, so we ran multiple times (20) and took the best solution.

Since there is stochasticity in durations and there are no theoretical guarantees on violations for all the three approaches, we have to employ two key criterion for comparisons with respect to solution quality:

- α-RM (or α Robust Makespan); and
- PoF (or Probability of Failure);

For two given approaches, let α-RM1 and PoF1 be values for approach 1 and α-RM2 and PoF2 be values for approach 2. Robust makespan values can be compared only when both approaches find feasible solutions (i.e., PoF values are less than or equal to α). Therefore, for comparing two approaches, there are five different possibilities:

1. PoF1 $\leq \alpha \wedge$ PoF2 $\leq \alpha \wedge \alpha$-RM1 $\leq \alpha$-RM2: Both approaches provide feasible solutions and approach1 has lower makespan than the approach2.
2. PoF1 $\leq \alpha \wedge$ PoF2 $\leq \alpha \wedge \alpha$-RM1 $> \alpha$-RM2: Both approaches provide feasible solutions and approach2 has lower makespan than approach1.
3. PoF1 $\leq \alpha \wedge$ PoF2 $> \alpha \wedge$: Only the first approach provides a feasible solution. This result demonstrates the ability of approach1 to obtain feasible solutions.
4. PoF1 $> \alpha \wedge$ PoF2 $\leq \alpha$: Only the second approach provides a fesaible solution. This result demonstrates the ability of approach2 to obtain feasible solutions.
5. PoF1 $> \alpha \wedge$ PoF2 $> \alpha$: None of the two approaches provide a feasible solution.

We first compare FLPV and SORU in the second column of Table 4: (1) In around 31% of the cases, both approaches returned feasible solution and in 28% of those cases, SORU had a higher makespan than FLPV. (2) In more than 23% of the cases, SORU was able to find a solution, but FLPV was unable to find a solution[8]. We believe this is due to FLPVs inability to handle maximum time lags, an issue highlighted even in their paper (Fu et al. 2012). (3) In 4% percentage of the cases, SORU was unable to find a solution[9], where FLPV was able to find a solution. (4) In around 42% of the cases, neither approach returned a feasible solution. ***Overall, of the 58% feasible cases, SORU was able to outperform FLPV in more than 88%(=$\frac{51}{58}$ %) of the cases.***

[7]We use this to refer to the best results provided by the two approaches, SLA and GNLA.

[8]PoFs were higher than even $\alpha + 0.1$ for most of the cases

[9]PoF values were almost always between α and $\alpha + 0.1$. We believe these cases can be better solved by adopting higher number of samples or a slightly lower γ. However, identifying instances where such lower γ or higher samples are required is an important problem for future work.

Result	J10	J20	J30
PoF1 $\leq \alpha \wedge$ PoF2 $\leq \alpha \wedge$ α-RM1 $\leq \alpha$-RM2	29%	27%	25%
PoF1 $\leq \alpha \wedge$ PoF2 $\leq \alpha \wedge$ α-RM1 $> \alpha$-RM2	3%	1%	1%
PoF1 $\leq \alpha \wedge$ PoF2 $> \alpha$	18%	17%	23%
PoF1 $> \alpha \wedge$ PoF2 $\leq \alpha$	4%	4%	1%
PoF1 $> \alpha \wedge$ PoF2 $> \alpha$	47%	51%	50%

Table 5: Comparison of SORU-H and FLPV on J10, J20 and J30 datasets for $\alpha = 0.2$. PoF1, α-RM1 correspond to SORU-H and PoF2, α-RM2 correspond to FLPV.

We provide results of comparison between SORU and SORU-H in the third column of Table 4: (1) In around 48% of the cases, both approaches were able to obtain feasible solutions. Of the 48% of cases, SORU-H was able to obtain lower makespan in 37% of the cases and SORU was able to obtain lower makespan in 11% of the cases. (2) In around 6% of the cases, SORU was able to find a solution when SORU-H was unable to find a solution. In only 0.3% of the cases, SORU-H was able to find a solution when SORU was unable to find a solution. (3) In around 45% of the cases, neither approach returned a feasible solution. ***Overall, of the 55% of the feasible cases for one of the two algorithms, in 67% (=$\frac{37}{55}$ %) of the cases SORU-H was able to outperform SORU***. Given the better run-time and solution quality performance of SORU-H, we provide comparisons between SORU-H and FLPV on all the three problem sets.

	FLPV	**SORU-H**
J10	44.21	40.43
J20	71.62	62.34
J30	79.09	73.99

Table 6: Comparison of average α-robust makespan.

Table 5 provides the results for the comparison between SORU-H and FLPV on all three problem sets:
(1) In at least 25% of the instances where SORU-H provided lower robust makespan values. (2) In at least 17% of the instances, SORU-H provided a feasible solution but FLPV was unable to find a solution. (3) Over all the three problem sets, in at most 7% of the instances, FLPV found a solution that was either better or SORU-H was unable to find a solution. (4) For all the three problem sets, there were around 50% instances where both approaches had no feasible solution. ***In summary, in at least 44% of the cases, SORU-H outperformed FLPV. In only a maximum of 7% of cases,***

FLPV outperformed SORU-H. The comparison of average makespan over the feasible instances for both SORU-H and FLPV are provided in Table 6. SORU-H provides a clear improvement over FLPV on all the three data sets.

Discussion

Currently, we have start-start temporal constraints in our optimization models for RCPSP/max and RCPSP/max with durational uncertainty. Our optimization model for RCPSP/max in Table 2 can trivially be adapted to account for end-start temporal constraints. For RCPSP/max with durational uncertainty, accounting for end-start temporal constraints is not immediately obvious. We need to update the temporal lag constraints (constraints on the first line) in Table 3 to use the z^q variables introduced in SORU.

$$s_j - e_i \geq T_{ij}^{min} - (1 - z^q) \cdot M$$
$$s_j - e_i \leq T_{ij}^{max} + (1 - z^q) \cdot M \quad (21)$$

We now use the same variable, z^q to keep track of whether a sample ξ^q violates resource or temporal constraints. Similar changes can be made to account for other types of constraints (start-end and end-end). Thus, our approach is not limited by the way temporal constraints are specified.

For uncertain duration problems, solutions can either be start time schedules or execution strategies. Both these solution types have their own advantages and disadvantages. While start time schedules are rigid, they are easy to compute, understand and execute. On the other hand, while partially ordered schedules are flexible, they are harder to compute and have difficultly handling maximum time lags (as illustrated with (Fu et al. 2012)).

In summary, our approaches have many useful properties. First, they can work with any uncertainty distribution, as long as their is a simulator for durations. Second, all types of temporal constraints, start-start, end-start, start-end and end-end can be handled with our approaches. Furthermore, unlike previous work, our approaches are better equipped to handle maximum time lag constraints. Third, not only does SORU-H compute schedules at comparable runtimes as the best existing approach from literature, it also provides higher quality solutions. Finally, we can exploit commercial optimization software (e.g., CPLEX) to solve our MIPs.

Acknowledgements

This research is supported by Singapore National Research Foundation under its International Research Centre @ Singapore Funding Initiative and administered by the IDM Programme Office, Media Development Authority (MDA).

References

Artigues, C.; Michelon, P.; and Reusser, S. 2003. Insertion techniques for static and dynamic resource constrained project scheduling. *European Journal of Operational Research* 149:249–267.

Bartusch, M.; Mohring, R. H.; and Radermacher, F. J. 1988. Scheduling project networks with resource constraints and time windows. *Annals of Operations Research* 16(1-4):201–240.

Beck, J. C., and Wilson, N. 2007. Proactive algorithms for job shop scheduling with probabilistic durations. *Journal of Artificial Intelligence Research* 28(1):183–232.

Bidot, J.; Vidal, T.; Laborie, P.; and Beck, J. C. 2009. A theoretic and practical framework for scheduling in a stochastic environment. *Journal of Scheduling* 12:315–344.

Cimatti, A.; Micheli, A.; and Roveri, M. 2015. Strong temporal planning with uncontrollable durations: A state-space approach. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015*, 3254–3260.

Davenport, A. J.; Gefflot, C.; and Beck, J. C. 2001. Slack-based techniques for robust schedules. In *Proceedings of the 6th European Conferences on Planning (ECP)*.

Fu, N.; Lau, H. C.; Varakantham, P.; and Xiao, F. 2012. Robust local search for solving rcpsp/max with durational uncertainty. *J. Artif. Intell. Res. (JAIR)* 43:43–86.

Herroelen, W., and Leus, R. 2005. Project scheduling under uncertainty: Survey and research potentials. In *European Journal of Operational Research*, volume 165(2), 289–306.

Kolisch, R., and Sprecher, A. 1996. Psplib - a project scheduling library. *European Journal of Operational Research* 96:205–216.

Koné, O.; Artigues, C.; Lopez, P.; and Mongeau, M. 2011. Event-based milp models for resource-constrained project scheduling problems. *Comput. Oper. Res.* 38(1):3–13.

Kouvelis, P.; Daniels, R. L.; and Vairaktarakis, G. 2000. Robust scheduling of a two-machine flow shop with uncertain processing times. *IIE Transactions* 32:421–432.

Lamas, P., and Demeulemeester, E. 2015. A purely proactive scheduling procedure for the resource-constrained project scheduling problem with stochastic activity durations. *Journal of Scheduling* 1–20.

Lombardi, M., and Milano, M. 2009. A precedence constraint posting approach for the rcpsp with time lags and variable durations. In *Proceedings of the 15th international conference on Principles and practice of constraint programming*, CP'09, 569–583.

Mulvey, J. M.; Vanderbei, R. J.; and Zenios, S. J. 1995. Robust optimization of large-scale systems. *Operations Research* 43.

Pagnoncelli, B.; Ahmed, S.; and Shapiro, A. 2009. Sample average approximation method for chance constrained programming: theory and applications. *Journal of Optimization Theory and Applications* 142:399–416.

Schutt, A.; Feydy, T.; Stuckey, P. J.; and Wallace, M. 2009. Why cumulative decomposition is not as bad as it sounds. In *Principles and Practice of Constraint Programming - CP 2009*, 746–761.

Schutt, A.; Feydy, T.; Stuckey, P. J.; and Wallace, M. G. 2013. Solving rcpsp/max by lazy clause generation. *Journal of Scheduling* 16(3):273–289.

Varakantham, P., and Kumar, A. 2013. Optimization approaches for solving chance constrained stochastic orienteering problems. In *Proceedings of Algorithmic Decision Theory (ADT)*, 387–398.