Proceedings of the Twenty-Sixth International Conference on
Automated Planning and Scheduling (ICAPS 2016)

Robust Partial Order Schedules for RCPSP/max with Durational Uncertainty

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

Abstract

In this work, we consider RCPSP/max with durational uncertainty. We focus on computing robust Partial Order Schedules (or, in short POS) which can be executed with risk controlled feasibility and optimality, i.e., there is stochastic posteriori quality guarantee that the derived POS can be executed with all constraints honored and completion before robust makespan. To address this problem, we propose BACCHUS: a solution method on **B**enders **A**ccelerated **C**ut **C**reation for **H**andling **U**ncertainty in **S**cheduling. In our proposed approach, we first give an MILP formulation for the deterministic RCPSP/max and partition the model into POS generation process and start time schedule determination. Then we develop Benders algorithm and propose cut generation scheme designed for effective convergence to optimality for RCPSP/max. To account for durational uncertainty, we extend the deterministic model by additional consideration of duration scenarios. In the extended MILP, the risks of constraint violation and failure to meet robust makespan are counted during POS exploration. We then approximate the uncertainty problem with computing a risk value related percentile of activity durations from the uncertainty distributions. Finally, we apply Pareto cut generation scheme and propose heuristics for infeasibility cuts to accelerate the algorithm process. Experimental results demonstrate that BACCHUS efficiently and effectively generates robust solutions for scheduling under uncertainty.

Introduction

Most research on project scheduling focus on a perfectly pre-defined scheduling environment. However, durational uncertainty in real world projects may always happen and make deadline-driven project management a challenging and difficult task. Thus, effectively handling durational uncertainty in project scheduling is of realistic and practical importance.

Broadly, one may classify the decision-making approaches for tackling uncertainty in scheduling into two categories: Proactive scheduling computes an apriori buffered schedule or policy before uncertainty occur. Reactive scheduling provides online decisions on starting next activity when uncertainty occurs. For a survey of existing works on project scheduling uncertainty, one may refer to (Fu, Lau, and Varakantham 2015; Bidot et al. 2009; Lombardi and Milano 2009; Beck and Wilson 2007; Herroelen and Leus 2005). In this paper, we are concerned with proactive scheduling. Instead of a baseline schedule, we are interested in a Partial Order Schedule (POS) which is a set of partially ordered activities such that any embedded temporal feasible solution is also guaranteed resource feasible (Policella et al. 2009). Within a POS, each activity retains a set of feasible starting times which provides temporal flexibility against uncertainty. A simulation based approach to evaluate the expected makespan of POSs was considered in (Bonfietti, Lombardi, and Milano 2014).

The concrete problem addressed in this paper is Resource Constrained Project Scheduling Problem with minimum and maximum time lags (abbrev. RCPSP/max). Extended from RCPSP, the introduction of temporal separation constraints, particularly maximum time lags between activities offers a wide range of modelling capabilities like activity deadlines, setup times, etc. But it also exemplifies the problem setting at a much higher level of complexity where the feasibility problem is already NP-complete (Bartusch, Mohring, and Radermacher 1988).

Akin to a few existing proactive approaches for considering uncertainty in JSP (Beck and Wilson 2007) and RCPSP/max (Varakantham, Fu, and Lau 2016; Fu et al. 2012), we consider a risk management objective. Specifically, given a risk parameter α ($0 \leq \alpha \leq 1$) which can be prescribed by the planner, we are interested in computing the robust POS which minimizes the α-quantile makespan distribution. We refer to the least α-quantile also as the α-robust makespan.

In (Fu et al. 2012), heuristic techniques based on local search were provided for generating robust POS for RCPSP/max under durational uncertainty. However, the limitation of that work is that the derived POS may not meet maximum temporal constraints for certain uncertainty realisations during execution. This work is motivated by effectively managing risk on temporal constraint violation and failure to meet robust makespan during POS construction. Instead of a local search algorithm, we design a Mixed Integer Linear Programming (MILP) model for POS construction and develop a Benders decomposition algorithm and heuristic approximation for efficient robust makspan calculation. In (Varakantham, Fu, and Lau 2016), a start

time schedule with robust makespan was explored through a proactive sampling based technique. In that work, the probability of failure is counted when resource capacity constraints are violated. The characteristic feature of our proposed algorithm is the effort to effectively account for risk controlled feasibility and solution quality in terms of robust makespan. Thus, we manage the risks when temporal constraint is violated, and/or the actual execution fails to meet the robust makespan.

More specifically, we propose an approach, **B**enders **A**ccelerated **C**ut **C**reation for **H**andling **U**ncertainty (BACCHUS). BACCHUS consists of five components:

- An MILP formulation for the deterministic RCPSP/max and partition of POS generation process from scheduling decision making.
- An effective cut generation mechanism explored for resolving constraints conflicts used in Benders decomposition algorithm.
- An extension of the exact MILP model for solving the deterministic problem with sampling approximation for accommodating additional durational uncertainty.
- A scalable solution to robust optimization for solving RCPSP/max with durational uncertainty.
- Cut generation enhancements during iteration process for accelerating Benders algorithm.

The Deterministic RCPSP/max

The 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 requires r_{ik} units of resources of type k to be executed for a duration of p_i time units without preemption.

Generalized temporal constraints for RCPSP/max can specify a minimal or maximal time lag between any pair of activities. A minimal time lag $T_{i,j}^{min}$ specifies that activity j can only be started (or finished) when activity i has already started (or finished) for a certain time period of $T_{i,j}^{min}$. A maximal time lag $T_{i,j}^{max}$ specifies that activity j should be started (or finished) at the latest a certain number of $T_{i,j}^{max}$ time units beyond the start (or finish) of activity i. Thus, 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).

The two types of constraints involved in the RCPSP/max can be summarized as follows:

- Generalized Temporal Constraints ($\mathbf{s} \leq \mathbf{T}$):

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

- Resource Capacity Constraints:

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

A schedule $\mathcal{S} = (s_1, \cdots, s_N)$ 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 feasible schedule, such that the project makespan, which is defined as the start time of the final dummy activity a_{N+1}, is minimized.

$$\begin{aligned} \min_{\mathbf{y}} \quad & v_{\mathbf{y}} \\ s.t. \quad & Pr((v_{\mathbf{y}}(\tilde{\mathbf{p}}) \geq v_{\mathbf{y}}) \vee (\mathbf{s} > \mathbf{T})) \leq \alpha \end{aligned}$$

Table 1: BACCHUS: Optimization Model

RCPSP/max with Durational Uncertainty

In this work, we consider project scheduling under uncertainty, where the stochastic characteristics apply to the activity durations. In the deterministic setting, makespan can be used to evaluate the performance of a project. However, when uncertainty is involved, the makespan itself becomes a random variable. Similar to existing work (Fu et al. 2012), we employ the metric of α-robust makespan as the project objective. As indicated earlier, α-robust makespan for a scheduling project can be defined as the minimum α-quantile value over all possible makespan distributions of POSs.

Formally, given a value of risk α ($0 \leq \alpha \leq 1$), our goal is to find a POS $\mathbf{y}$ with the makespan achieving the value of α-robust makespan. Let $v_{\mathbf{y}}$ represent the α-quantile of the makespan distribution represented by POS $\mathbf{y}$, i.e.,

$$Pr((v_{\mathbf{y}}(\tilde{\mathbf{p}}) \geq v_{\mathbf{y}}) \vee (\mathbf{s} > \mathbf{T})) \leq \alpha$$

where $v_{\mathbf{y}}(\tilde{\mathbf{p}})$ is a random variable that denotes the makespan distribution represented by POS $\mathbf{y}$ and $\tilde{\mathbf{p}}$ denotes the uncertain activity durations. Therefore, the α-robust makespan v^*, which is also the least value of $v_{\mathbf{y}}$ over all possible makespan distributions, can be computed by solving the robust optimization problem in Table 1.

Solving the Deterministic RCPSP/max

Inspired by the work in (Artigues, Michelon, and Reusser 2003) on a flow-based continuous time formulation for RCPSP, we give an MILP formulation for RCPSP/max for POS construction and schedule determination. This model would be extended to handle durational uncertainty in the stochastic scheduling problem presented in the following section.

The Model

We define the following decision variables:

- x_{ij}^k: resource flow variables representing the number of resource units of type k transferred directly from activity a_i to activity a_j.
- y_{ij}: sequencing variables used for POS construction, $y_{ij} = 1$ if and only if activity a_i precedes activity a_j.
- s_i: scheduling variables for determining the start time of activity a_i.

$$
\begin{aligned}
v = \min \quad & s_{n+1} \\
s.t. \quad & s_j - s_i \geq T_{ij}^{min} \quad \forall (i,j) \in T^{min} && (1)\\
& s_j - s_i \leq T_{ij}^{max}, \quad \forall (i,j) \in T^{max} && (2)\\
& s_j \geq s_i + p_i^0 - M(1 - y_{ij}) \quad \forall i,j && (3)\\
& s_i \geq 0 \quad \forall i && (4)\\
& x_{ij}^k \leq min\{r_{ik}, r_{jk}\} y_{ij} \quad \forall i \neq 0, j \neq n+1, k && (5)\\
& x_{0j}^k \leq r_{jk} \quad \forall j,k && (6)\\
& x_{in+1}^k \leq r_{ik} \quad \forall i,k && (7)\\
& \sum_j x_{ij}^k = \sum_j x_{ji}^k = r_{ik} \quad \forall i \neq 0, n+1, k && (8)\\
& \sum_j x_{0j}^k = \sum_j x_{jn+1}^k \leq C_k \quad \forall k && (9)\\
& x_{ij}^k \geq 0, \quad \forall i,j,k && (10)\\
& y_{ij} \in \{0,1\} \quad \forall i,j && (11)\\
& y_{ij} + y_{ji} \leq 1 \quad \forall i,j && (12)
\end{aligned}
$$

Table 2: RCPSPMax($\mathbf{p^0}$)

$$
\begin{aligned}
& Master - MILP(\rho)\{ \\
& \qquad \min \quad z \\
& \qquad s.t. \quad Constraints \quad 5-12 \\
& \qquad z \geq \alpha_\tau(\mathbf{y}) \quad \forall \tau = 1, ...\rho && (13)\\
& \qquad \beta_\tau(\mathbf{y}) \geq 0 \quad \forall \tau = 1, ...\rho && (14)\\
& \qquad \}
\end{aligned}
$$

Table 3: The Master Problem

$$
\begin{aligned}
v(\bar{\mathbf{y}}^\rho) = \min \quad & s_{n+1} \\
s.t. \quad & s_j - s_i \geq T_{ij}^{min} \quad \forall (i,j) \in T^{min} && (15)\\
& s_j - s_i \leq T_{ij}^{max}, \quad \forall (i,j) \in T^{max} && (16)\\
& s_j \geq s_i + p_i^0 - M(1 - \bar{y}_{ij}^\rho) \quad \forall i,j && (17)\\
& s_i \geq 0 \quad \forall i && (18)
\end{aligned}
$$

Table 4: The Slave Problem

Table 2 presents the MILP formulation. Constraints 1 and 2 express the generalized temporal constraints with minimum and maximum time lags between the starting times of two activities. Constraint 3 links the starting times of activities a_i and a_j with sequencing variables y_{ij}. In other words, the constraint constructs POS in terms of y_{ij}. It is active when $y_{ij} = 1$ which enforces the precedence relationship $s_j \geq s_i + p_i^0$. Otherwise, the constraint is always satisfied if there is no precedence relationship between a_i and a_j, i.e., $y_{ij} = 0$. In that case, no resource flows are carried from a_i to a_j which sets $x_{ij}^k = 0$. But if a_i precedes a_j, the maximum resource flow sent a_i to a_j is forced to be $min\{r_{ik}, r_{jk}\}$, as shown in Constraint 5. The constraint also expresses that if there is a positive resource flow transfer from a_i to a_j, i.e., $x_{ij}^k > 0$, then the precedence relation is enforced with $y_{ij} = 1$. Constraints 6 and 7 handle boundary conditions.

Constraints 8-10 are flow conservation constraints where Constraint 8 states that the total flows sent to and from non-dummy activity a_i equal to its resource requirement of the corresponding resource type, r_{ik}. The total resource of type k for dispatching into the project network from the starting dummy node a_0 and collected at the sink dummy node a_{n+1} is upper bounded by its capacity, C_k as represented in Constraint 9. Constraints 11 and 12 are constituted for POS construction. Constraint 12 covers the total three relationships between two activities, either a_i precedes a_j, or a_j precedes a_i, or a_i and a_j are executed in parallel.

Benders Decomposition Algorithm

Benders Decomposition (Benders 1962) is a solution approach for large scale combinatorial optimization problem, based on the idea of partition and cut generation. One may refer to (Li and Womer 2009) and (Costa 2005) for its successful application in solving multi-skilled personnel and network design problem. In this work, we developed this decomposition algorithm for solving RCPSP/max.

In the MILP formulation of Table 2, there are three types of decision variables. The flow variable $\mathbf{x}$ and starting time variables $\mathbf{s}$ are continuous, but the sequencing variables $\mathbf{y}$ are complicating in taking binary integers. To provide an efficient solution method for solving large scale RCPSP/max, we partition those decision variables into two sets, $(\mathbf{x}\&\mathbf{y}, \mathbf{s})$. $\mathbf{x}\&\mathbf{y}$ are involved in the master problem where a resource feasible precedence decision rule is generated. For a fixed decision rule, the slave focuses on generating the optimal schedule by deciding the starting times $\mathbf{s}$.

At each iteration, we solve a relaxed master problem and generate a resource feasible POS. For a fixed POS $\mathbf{y}$, the slave problem decides starting time schedule with the goal of minimizing the project makespan. The Master problem and the Slave problem at the ρ^{th} iteration are given in Table 3 and Table 4, respectively. An each iteration, new cuts (to be explained in next section) as in Constraint 13 (optimality cuts) and Constraint 14 (global cuts and feasibility cuts) are added to the master problem and then make it progress towards an optimal solution.

Cut Generation Scheme

The way of generating effective cuts during iteration process is always the key to the success of a Benders decomposition algorithm. In this work, we developed three types of cuts for guiding master to generate good candidate POSs for converging to optimality: Global Cuts, Optimality Cuts and Feasibility Cuts.

Global Cuts

We first explore the static global cuts before the main algorithm iterations start. The main feature of our proposed cuts generation scheme is to explore problem structures and build constraints to avoid conflicting assignments to the sequenc-

ing variables $\mathbf{y}$. The role of solving the slave problem is to trigger the violated constraints and infer the cuts to the master problem. In the following, we infer cuts from the causes of infeasibility by temporal analysis and precedence transitivity analysis.

Temporal Analysis For activities a_i, a_j and a_m, we first update the existing temporal constraints between activities by using the following formulas:

- $T_{ij}^{max} \geq T_{im}^{max} + T_{mj}^{max} \implies T_{ij}^{max} = T_{im}^{max} + T_{mj}^{max} \ \forall i, j, m$
- $T_{ij}^{min} \leq T_{im}^{min} + T_{mj}^{min} \implies T_{ij}^{min} = T_{im}^{min} + T_{mj}^{min} \ \forall i, j, m$
- $T_{ij}^{max} \geq T_{mj}^{max} - T_{mi}^{min} \implies T_{ij}^{max} = T_{mj}^{max} - T_{mi}^{min} \ \forall i, j, m$
- $T_{ij}^{min} \leq T_{mj}^{min} - T_{mi}^{max} \implies T_{ij}^{min} = T_{mj}^{min} - T_{mi}^{max} \ \forall i, j, m$

The idea is, to reconstruct the temporal time lags between activities if tighter lags can be inferred. Based on the temporal analysis designed for resolving resource conflicts, we have the following observations and such constraints must be satisfied for all feasible solutions.

- $T_{ij}^{min} \geq p_i^0 \implies y_{ij} = 1 \quad \forall i, j$

For activities a_i and a_j, if the minimum time lags T_{ij}^{min} is no less than the duration of activity i, a_i must be executed before a_j to guarantee the solution feasibility. For this case, we add the cut $y_{ij} = 1$.

- $T_{ij}^{max} < p_i^0 \implies y_{ij} = 0 \quad \forall i, j$

If the maximum time lag T_{ij}^{max} is less than the duration of activity a_i, then a_j cannot be executed after a_i and we set $y_{ij} = 0$.

- $\exists$ resource $k, r_{ik} + r_{jk} > C_k \implies y_{ij} + y_{ji} = 1 \quad \forall i, j, k$

If there exists a resource type k, of which the overall consumption of activities a_i and a_j exceeds the capacity C_k, then both activities cannot be executed in parallel in order to respect resource constraints. In such a case, either a_i precedes a_j or a_j precedes a_i. Thus, we add the cut $y_{ij} + y_{ji} = 1$ to remove the parallel case.

- $p_i^0 + p_m^0 > T_{in}^{max} + T_{mj}^{max} \implies y_{ij} + y_{mn} \leq 1 \ \forall i, j, m, n$

Proof. Suppose $y_{ij} + y_{mn} > 1$, i.e., $y_{ij} = y_{mn} = 1$, we then have $s_j \geq s_i + p_i^0$ and $s_n \geq s_m + p_m^0$. The summation of these two inequations imply that $(s_n - s_i) + (s_j - s_m) \geq p_i^0 + p_m^0$. Given the definition of temporal constraints, we then have $T_{in}^{max} + T_{mj}^{max} \geq p_i^0 + p_m^0$. Hence proved.
This type of cut resolves the problem that pairs of activities cannot be sequentially connected simultaneously due to relationship between activity durations and time lags.

- $p_i^0 + p_m^0 > T_{in}^{max} - T_{jm}^{min} \implies y_{ij} + y_{mn} \leq 1 \ \forall i, j, m, n$

As above.

- $T_{ij}^{max} < p_i^0 + p_m^0, \implies y_{im} + y_{mj} \leq 1 \quad \forall i, j, m$

Proof. Suppose $y_{im} + y_{mj} > 1$, i.e., $y_{im} = y_{mj} = 1$, we then have $s_m \geq s_i + p_i^0$ and $s_j \geq s_m + p_m^0$. The summation of these two inequations imply that $s_j - s_i \geq p_i^0 + p_m^0$. Given $s_j - s_i \leq T_{ij}^{max}$, we then obtain $T_{ij}^{max} \geq p_i^0 + p_m^0$. Hence proved.

$$\max \sum_{i,j!=i} (T_{ij}^{min}\lambda_{ij} - T_{ij}^{max}\mu_{ij} + (p_i^0 - M(1-\bar{y}_{ij}))\gamma_{ij})$$

$s.t.$

$$\sum_{j|j\neq i, i\neq n+1} (\lambda_{ij} - \lambda_{ji} - \mu_{ij} + \mu_{ji} + \gamma_{ij} - \gamma_{ji}) \geq 0 \ \forall i \quad (19)$$

$$\sum_{j|j\neq i, i=n+1} (\lambda_{ij} - \lambda_{ji} - \mu_{ij} + \mu_{ji} + \gamma_{ij} - \gamma_{ji}) \geq -1 \quad (20)$$

$$\lambda_{ij}, \mu_{ij}, \gamma_{ij} \geq 0, \ \forall i, j, i \neq j \quad (21)$$

Table 5: The Slave Dual

Precedence Transitivity Analysis

- $y_{im} \geq y_{ij} + y_{jm} - 1 \quad \forall i, j, m$

The set of cuts are used to capture the transitivity of the ordering relations: if a_i precedes a_j and a_j precedes a_m, then we have a_i precedes a_m.

Optimality Cuts

For a fixed policy $\mathbf{y}$, if the slave is feasible, the minimization objective achieved at that iteration would be an upper bound to the optimization problem. Optimality cuts can then be derived and added to the master in next iteration to improve the approximation of the master objective from the optimal solution.

To examine the slave MILP model, we introduce non-negative dual variables λ_{ij}, μ_{ij} and γ_{ij} for Constraints 15, 16, and 17, respectively. Table 5 shows the dual LP of the slave problem. Suppose that the primal LP in Table 4 is always feasible for every policy $\mathbf{y}$ chosen, then the dual LP has a bounded feasible region. Let set U contain the extreme points of the polyhedron D defined by the constraints of Table 5 then we can instantiate Constraint 13 in the master by updating optimal cuts as provided in the following constraint:

$$\alpha_\rho(\mathbf{y}) = \sum_{i,j\neq i} (T_{ij}^{min}\lambda_{ij}^\rho - T_{ij}^{max}\mu_{ij}^\rho + (p_i^0 - M(1-y_{ij}))\gamma_{ij}^\rho) \quad (22)$$

where $(\lambda, \mu, \gamma) \in U$.

Every time a slave generates an optimal solution, an optimal cut can be constructed in terms of the dual values of the slave. Such constraints are then added to the master problem to improve the lower bound of the optimization problem.

Feasibility Cuts

However, it is possible that during iterating process, the generated POS from a relaxed master cannot produce a feasible schedule in the slave model. The feasibility cuts need to be inferred to deal with such cases. The feasibility cuts for iteration ρ have the general form in the following constraint:

$$\sum_{i,j|\bar{y}_{ij}^\rho=1} (1 - y_{ij}) + \sum_{i,j|\bar{y}_{ij}^\rho=0} y_{ij} \geq 1. \quad (23)$$

The idea is that, once infeasibility occurs at POS $\bar{y}^\rho$, the total number of binary variables flipping their value with respect to $\bar{y}^\rho$ either from 1 to 0 or from 0 to 1 is at least 1.

Let UB and LB represent the upper bound and the lower bound of the optimal solution, respectively. Algorithm 1 provides pseudocode of the extended benders decomposition algorithm for RCPSP/max.

Algorithm 1: Extended Benders Decomposition Algorithm (RCPSP/max Instance)

```
LB ← 0, UB ← ∞, ρ ← 0
Temporal analysis to obtain global cuts in Constraint 14;
terminate ← false
while terminate = false do
    Solve master problem and obtain ȳ^ρ;
    if z(ρ)= UB then
        terminate ← true;
    else
        LB ← z(ρ);
        Update constraints in the Slave problem;
        Solve the slave problem;
        if The slave problem is feasible then
            Update UB;
            Generate optimality cuts in Constraint 13;
        else
            Generate feasibility cuts in Constraint 14;
        end if
        Add cuts to master problem;
    end if
    ρ ← ρ + 1
end while
```

Solving RCPSP/max with Durational Uncertainty

Let s_i^q represent the scheduling variable for determining the start time of activity a_i on sample q, where $q = 1, ...Q$ and Q is the total number of samples. The indicator variable z_q equal to 1 if and only if the constructed POS cannot be executed feasibly or efficiently on sample q. As POS is generated from the master problem where resource feasibility were accommodated during construction, the temporal aspect is the only concern for feasibility. With the objective of robust optimization problem in Table 1 being generating a POS with the best robust makespan value, in the stochastic model of Table 6, we consider both feasibility and efficiency properties during POS exploration.

Specifically, Constraints 27-29 control the potential probability of failure during the POS construction. As the introduction of resource flow variables would already guarantee resource feasibility during schedule generation, $z_q = 0$ enforces the temporal constraints of maximum time lags, thus, it guarantees the schedule feasibility, as shown in Constraint 27. Let α represent the risk threshold that can be prescribed by the planner. Constraint 28 measures solution quality in terms of successful execution on Q different scenarios of realized uncertainty, and Constraint 29 considers the makespan for performance evaluation. $z_q = 1$ if the realized makespan of sample q termed as s_{n+1}^q reaches beyond

$$
\begin{aligned}
& \min \ RM \\
s.t. \ & Constraints\ 5-12 \\
& s_j^q \geq s_i^q + p_i^q - M(1-y_{ij}) \quad \forall i,j,q && (24)\\
& s_j^q - s_i^q \geq T_{ij}^{min} \quad \forall (i,j) \in T^{min}, q && (25)\\
& s_i^q \geq 0 \quad \forall i,q && (26)\\
& s_j^q - s_i^q \leq T_{ij}^{max} + z^q M \quad \forall i,j,q && (27)\\
& \sum_q z^q \leq \alpha Q \quad \forall q && (28)\\
& Mz^q > s_{n+1}^q - RM \quad \forall q && (29)
\end{aligned}
$$

Table 6: RCPSPMaxUnc ($\{\mathbf{p^q}\}_{q=1,...Q}$)

the robust makespan RM.

Though a very useful model for representing uncertainty in durations, as the problem scale increases, the dependency on number of samples employed gets the scalability issue of the multi-sample model worsened and it can hardly find solutions with only 20 samples adopted for J20 and J30 instances within a 10 minute time limit. To overcome the scalability problem, we adopt the idea of percentile sample approximation as in (Varakantham, Fu, and Lau 2016) and summarize the scenario sample set by using one $(1-\alpha)$-percentile duration sample. The duration of activity a_i in the percentile sample can be presented as $\hat{p}_i = PERCENTILE_{1-\alpha}(\mathbf{p}_i)$, where $\mathbf{p}_i = \{p_i^1, \cdots, p_i^q, \cdots p_i^Q\}$ and p_i^q is the duration of activity a_i in sample q. The idea is to entail that at least in $(1-\alpha)\cdot Q$ number of samples, duration of activity a_i is lower than $\hat{p}_i$. Therefore, the stochastic model in Table 6 can be summarized in RCPSPMaxUnc ($\hat{\mathbf{p}}$) with the implementation to be shown in next section.

Experimental Results

In this section, we conducted computational experiments to test the performance of BACCHUS and compare with the existing best known approaches SORU-H (Varakantham, Fu, and Lau 2016) and FPVL (Fu et al. 2012) for generating α-robust makepan for RCPSP/max under durational uncertainty.

The problem instances we run BACCHUS on are extended from benchmark sets J10 J20 and J30 for RCPSP/max from PSPlib (Kolisch, Schwindt, and Sprecher 1998). Each data set contains 270 instances and each instance has 5 types of resources available. The number of activities for instances in J10, J20 and J30 are 10, 20 and 30, respectively. We assume the processing time p_i of activity a_i is normally distributed with mean value corresponding to the deterministic duration p_i^0 given by benchmarks and standard deviation denoted as σ. The duration scenarios are generated by sampling from normal distributions with three different duration variabilities $\sigma = \{0.1, 0.5, 1\}$. We implemented BACCHUS in Java and Cplex studio on a Core i7-4790 CPU 3.60GHz processor with 32.0 GB RAM and

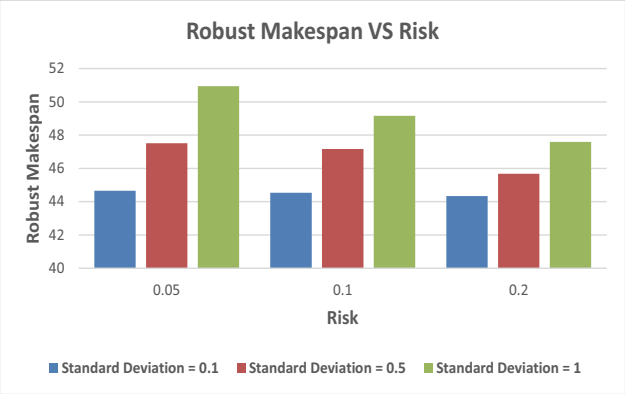

Figure 1: Robust Makespan VS Risk α and Standard Deviation σ

$$\max \sum_{i,j!=i} (T_{ij}^{min}\lambda_{ij} - T_{ij}^{max}\mu_{ij} + (p_i^0 - M(1-\bar{y}_{ij}^0))\gamma_{ij})$$
$$s.t.$$
$$Constraints\ 19-21$$
$$v(\bar{\mathbf{y}}) = \sum_{i,j!=i} (T_{ij}^{min}\lambda_{ij} - T_{ij}^{max}\mu_{ij} + (p_i^0 - M(1-\bar{y}_{ij}))\gamma_{ij})$$

Table 7: Pareto_optimal Cuts Generation

average the results over 10 runs for each problem instance.

We first run experiments on three data sets across increasing levels of risk $\varepsilon = \{0.05, 0.1, 0.2\}$ and varying standard deviations. For open problem instances that cannot be solved to optimality, we report the best results obtained within a 10-minute time limit. Figure 1 summarizes the average values of robust makespans over 270 instances in J10 with varying α and ε. We observe that, the value of robust makespan increases as the level of risk decreases. In other words, the lower risk value the decision maker is willing to take, the higher robust makespan for the POS generated. The lower degree of durational perturbation, the better value of robust makespan obtained.

Then, we explore heuristic enhancements for generating effective cuts and improving the algorithm efficiency. During the procedure of Benders iterations, it may happen that the slave solution is not unique, especially when the slave optimization suffers from degeneracy. To reduce the number of iterations towards convergence during Benders processing, the impact of stronger cuts were examined in (T. L. Magnanti 1981). In our experiments, we adopt the same idea and implement Pareto optimal cut generation for exploring stronger cuts. At iteration ρ, a cut generated from extreme point $(\lambda^{\rho 1}, \mu^{\rho 1}, \gamma^{\rho 1})$ is said to dominate a cut generated from extreme point $(\lambda^{\rho 2}, \mu^{\rho 2}, \gamma^{\rho 2})$, if $\alpha_\rho(\bar{\mathbf{y}}, \lambda^{\rho 1}, \mu^{\rho 1}, \gamma^{\rho 1}) \geq \alpha_\rho(\bar{\mathbf{y}}, \lambda^{\rho 2}, \mu^{\rho 2}, \gamma^{\rho 2})$ defined in Equation 22 with strict inequality at least one $\bar{\mathbf{y}}$. A cut dominated by no other cuts is said to be Pareto optimal. A Pareto optimal cut can be generated by solving the model in Table 7, where $v(\bar{\mathbf{y}})$ is the optimal solution value of the slave problem at current iteration. The objective maximizes strength of the cut for relative interior y^0 of the feasible space defined in master constraints, and constraints of the model specify the feasible space of the slave problem. We implement Pareto optimal cut generation scheme on large scale instances of J20 and J30 and observe an average reduction of 18.6% and 23.9% of running time required to solve to optimality, respectively.

When the slave problem becomes infeasible during iterating process, an alternative way is to change sequencing decisions on a $(a > 1)$ pairs of activities simultaneously with respect to the current POS $\bar{\mathbf{y}}$ generated, rather than only one pair of activities, as of the RHS parameter of Equation 23. In other words, the feasibility cut can be improved as

$$\sum_{i,j|\bar{y}_{ij}^\rho=1} (1 - y_{ij}) + \sum_{i,j|\bar{y}_{ij}^\rho=0} y_{ij} \geq a. \tag{30}$$

The motivation for varying step size a is that, for large scale problem instances, restricting only one sequencing decision to be revised at each iteration may not be efficient enough to restore feasibility. We vary the value of a from 1 to 5, and observe that the number of infeasible iterations for almost all instances get reduced to some extent. But there is no strict monotonicity between the running time and values of a. To show this trend, we randomly pick one instance from each data set and report the running time in logarithm value with different a values in Figure 2. We observe that, as the step size slightly increases from 1, the running times required for all instances decreases obviously. But as a increases, there seems exist a threshold where step size taking higher values would bring negative effect on algorithm efficiency. Unfortunately, we are not able to find a universal optimal value of a for best algorithm performance, because it is highly instance dependent. Note that in Figure 2, the running time was solved optimality for all instances, except for J10 instance at $a = 5$, where optimality is comprised and higher robust makespan is returned. One direct insight from our observation is that, a large step size a may be over-corrected for restoring feasibility during benders iterations, especially for small scale problems. But it would still be promising to design a good step size value for improving algorithm efficiency for specific problem, especially when the size is large.

Now we compare BACCHUS with the existing approaches SORU-H (Varakantham, Fu, and Lau 2016) and FPVL (Fu et al. 2012) to solving RCPSP/max with durational uncertainty. All three works aim for exploring α-robust makepan. Thus, a straightforward metric to perform a computational comparison is on the value of α-robust makespan. Table 8 summaries the average values of α-robust makespan obtained from BACCHUS, FPVL, SORU-H over 270 instances of all three benchmark sets for $\alpha = 0.1$ and $\sigma = 0.5$. We first compare values of robust makespan with FPVL as both solutions returned in BACCHUS and FPVL are POSs. The experimental results show that BACCHUS outperforms FLPV in generating more robust POSs.

Instead of scheduling policy exploration, SORU-H focuses on generating the start time schedule. Though BACCHUS has the same robust makespan objective, the ways of

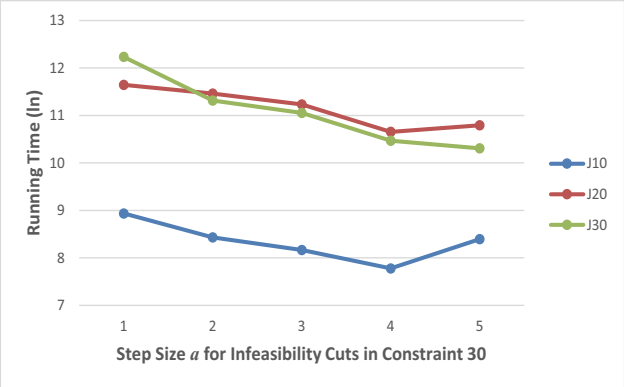

Figure 2: Effect of Step Size a for Infeasibility Cuts on Algorithm Efficiency

	BACCHUS	FLPV	SORU-H
J10	47.26	51.3	48.43
J20	82.34	84.6	79.8
J30	107.27	113.3	103.2

Table 8: Comparison with FLPV and SORU-H on α-robust makespan

tackling uncertainty in BACCHUS and SORU-H are totally different. In SORU-H, temporal constraints are always honored and resource violation is the only concern for schedule infeasibility. BACCHUS solves instead hard resource constraints and soft temporal constraints. Given the difference of solution results and approaches between SORU-H and BACCHUS, robust makespan may not be a proper index for making comparison of solution performances. A promising direction of our future work is to implement the resulted solutions from BACCHUS and SORU-H in real world scheduling environment and evaluate the performance by examining the real probability of failure under different uncertainty scenarios. Thus, though the values of robust makespan for BACCHUS is slightly higher than SORU-H for J20 and J30, we still believe that BACCHUS can provide an alternative solution in the form of more flexible POSs for decision makers to efficiently tacking project scheduling problem under uncertainty.

Conclusion

In this work, we proposed BACCHUS: a solution method for handling uncertainty in scheduling, consisting of four phases: 1) modeling RCPSP/max in the form of MILP and partition into the POS determination and the start time scheduling process; 2) developing Benders decomposition algorithm and proposing cut generation scheme by exploring the problem structure of RCPSP/max; 3) computing for robust POSs that can be executed within risk controlled performance guarantee of feasibility and optimality; 4) accelerating Benders iterations from heuristic based techniques. Experimental results demonstrate BACCHUS outperforms the best approach on robust POS generation by deriving better robust makepan. This is also the first work that potential risk on temporal constraint violations are proactively managed in POS construction to hedge against uncertainty.

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.

Benders, J. 1962. Partitioning procedures for solving mixed-variables programming problems. *Numerische Mathematik* 4:238–252.

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.

Bonfietti, A.; Lombardi, M.; and Milano, M. 2014. Disregarding duration uncertainty in partial order schedules? yes, we can! In *Integration of AI and OR Techniques in Constraint Programming - 11th International Conference, CPAIOR*, 210–225.

Costa, A. M. 2005. A survey on benders decomposition applied to fixed-charge network design problems. *Comput. Oper. Res.* 32(6):1429–1450.

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.

Fu, N.; Lau, H. C.; and Varakantham, P. 2015. Robust execution strategies for project scheduling with unreliable resources and stochastic durations. *J. Scheduling* 18(6):607–622.

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.; Schwindt, C.; and Sprecher, A. 1998. *Benchmark Instances for Project Scheduling Problems*. Kluwer Academic Publishers, Boston. 197–212.

Li, H., and Womer, K. 2009. Scheduling projects with multi-skilled personnel by a hybrid milp/cp benders decomposition algorithm. *J. Scheduling* 12(3):281–298.

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.

Policella, N.; Cesta, A.; Oddi, A.; and Smith, S. 2009. Solve-and-robustify. *Journal of Scheduling* 12:299–314.

T. L. Magnanti, R. T. W. 1981. Accelerating benders decomposition: Algorithmic enhancement and model selection criteria. *Operations Research* 29(3):464–484.

Varakantham, P.; Fu, N.; and Lau, H. C. 2016. A proactive sampling approach to project scheduling under uncertainty. In *AAAI*.