Journal of Scheduling manuscript No.
(will be inserted by the editor)

Robust Execution Strategies for Project Scheduling with Unreliable Resources and Stochastic Durations

Na FU · Hoong Chuin LAU · Pradeep VARAKANTHAM

Received: date / Accepted: date

Abstract The Resource Constrained Project Scheduling Problem with minimum and maximum time lags (RCPSP/max) is a general model for resource scheduling in many real-world problems (such as manufacturing and construction engineering). We consider RCPSP/max problems where the durations of activities are stochastic and resources can have unforeseen breakdowns. Given a level of allowable risk, α, our mechanisms aim to compute the minimum robust makespan execution strategy. Robust makespan for an execution strategy is any makespan value that has a risk less than α. The risk for a makespan value, M given an execution strategy is the probability that a schedule instantiated from the execution strategy will not finish before M given the uncertainty over durations and resources.

We make three key contributions: (a) Firstly, we provide an analytical evaluation of resource breakdowns and repairs on executions of activities; (b) We then incorporate such information into a local search framework and generate execution strategies that can absorb resource and durational uncertainties; and (c) Finally, to improve robustness of resulting strategies, we propose Resource Breakdown Aware Chaining procedure with three different metrics. This chaining procedure computes resource allocations by predicting the effect of breakdowns on robustness of generated strategies. Experiments show effectiveness of our proposed methods in providing more robust execution strategies under uncertainty.

Na FU · Hoong Chuin LAU · Pradeep VARAKANTHAM
E-mail: nafu@smu.edu.sg E-mail: hclau@smu.edu.sg E-mail: pradeepv@smu.edu.sg
School of Information Systems,
Singapore Management University,
80 Stamford Road, 178902 Singapore

Keywords Project scheduling · Risk management · Robustness and sensitivity analysis · Uncertainty modeling

1 Introduction

Research on scheduling has typically considered problems with fixed resource capacities and deterministic durations. However, in real-world environments, it is difficult to stick to the baseline schedule due to external uncontrollable events such as manpower unavailability, machine breakdowns, weather changes, etc. Existing literature has referred to such external events as disruptions. Broadly, this paper is motivated towards managing such disruptions in the context of project scheduling problems. In disruption management for project scheduling [38], there are three categories of disruptions: project network disruptions, activity disruptions and resource disruptions. Existing research in project scheduling has individually considered activity disruptions (durational variability for activities) [3,12,15,28, 35,37] and resource disruptions (due to breakdowns) [24–27]. Research in this paper builds over this existing research along two dimensions: (a) We solve RCPSP/max where there exist both activity and resource disruptions; and (b) We consider the more realistic risk aware objective to hedge against uncertainty associated with activity and resource disruptions.

A direct consequence of considering disruptions is a delay in the scheduled completion time of the project. A second consequence of stochasticity due to disruptions is the additional computational complexity over and above solving the underlying deterministic scheduling problem. For example, in the infinite-resource project scheduling problem where processing times have two possible discrete values, the problem of computing the expected makespan (or any point on the cumulative distribution of the optimal makespan), is #P-hard [18, 29]. It has also been shown that for the one-machine scheduling problem with stochastic durations, the problem of computing a policy (i.e., execution strategy) maximizing the probability that some job completes exactly at the deadline is PSPACE-hard [10]. A one-machine scheduling problem with probabilistic durations was also considered in [9], with an objective to capture the likelihood that a schedule yields actual performance no worse than a given target level. This has been shown to be NP-hard even though the underlying deterministic problem can be solved in polynomial time.

In this paper, the deterministic problem of interest is the Resource Constrained Project Scheduling Problem with minimum and maximum time lags (RCPSP/max). It represents important scheduling problems in manufacturing, logistics and project management. Though these problems have been shown to be NP-hard [2], local search based techniques [13] have achieved great success in solving these problems. Taking a cue from this, Fu *et al* [15] combined techniques from robust optimization with local search and solved RCPSP/max with durational uncertainty from a risk management perspective to obtain a Partial Order Schedule (POS) [32] (a type of execution strategy). More specifically, given a level of risk $0 < \varepsilon \leq 1$, the goal is to obtain the minimum *robust makespan* POS.

As indicated earlier, we consider RCPSP/max where there are both activity and resource disruptions. Our approach extends the robust local search mechanism proposed for RCPSP/max with uncertain activity durations [15]. Precisely, we are interested in the following problem: given activity disruptions described by a random variable with mild distributional support (such as mean and variance) for each activity, and resource breakdown described by exponential distributions for the time between failure as well as repair time for each resource, construct a minimum robust makespan POS. Concretely, we make three key contributions over our previous work in [15]: (a) We propose Resource Breakdown Aware Chaining procedure with three different metrics to obtain execution strategies based on resource breakdown and repair distributions; (b) We develop a mechanism that translates resource breakdowns to (further) duration variability of activities (extending on the work of [26]); and (c) We extend the local search framework from [15] for computing an execution strategy (using the chaining algorithm in (a)) that seeks to find the minimum *robust makespan*, updated to account for the further durational variability obtained in (b). It should be noted that we have extended the work in our conference paper [16] using the contribution in (a).

The rest of the paper is organized as follows. In the next section, we present existing relevant research, followed by a brief background and solution concepts referred to in this paper. We then present the mechanism for mapping resource breakdown to duration variability in Section 4. In Section 5, we describe the three metrics that are used in the chaining procedure to improve robustness of execution strategies. In Section 6, we present the extended robust local search framework that incorporates the chaining procedure and the measurement of upper bound on robust makespan of execution strategies using the updated durational variability. Finally, we describe our experimental setup and results in Section 8.

2 Literature Review

In the recent decades, there has been a growing interest to account for uncertainty in scheduling [1, 3, 5, 19, 28, 35]. Broadly, one may classify the techniques to tackle scheduling with uncertainty into two categories: *Proactive Scheduling* is to design a priori schedule or a schedule policy (i.e. execution strategy) that take into account the possible uncertainty that may occur; *Reactive Scheduling* modifies or re-optimizes the baseline schedule when an unexpected event occurs. This work focuses on proactive scheduling, where the goal is to compute an execution strategy that is robust to uncertainties from different sources.

The main idea of *proactive* techniques is to build a global solution which does not need to be revised during execution. One line of work of proactive scheduling is to consider design of good schedule policies (e.g. [30]) that provide online decision rules such that at time t, the policy decides which task(s) may start and which resource(s) to assign. In [11], resource-based policies for RCPSP with uncertain durations was applied, where the decision times are determined by the realization of durational uncertainty. The objective of that work is to determine a project execution policy with minimized execution cost. Another example is the notion of Partial Order Schedule (POS) defined in [32] which seeks to retain temporal flexibility whenever the problem constraints allow it and can often absorb unexpected deviation from predictive assumptions. They considered robustness measures such as fluidity and flexibility. Different methods of generating POSs were compared in the work of [33] with respect to robustness of resulting schedules. In [17] , POS was adopted as an execution strategy when addressing RCPSP/max with durational uncertainty. In this work, we solve a much more complex problem, where both resource breakdowns and durational uncertainty are present.

The problem of minimizing schedule instability for RCPSP with stochastic resource availability was considered in [24–27]. Schedule instability is defined as the expected weighted deviation between the computed and the realized schedules. In [25,26], the expected values of durational extensions due to resource breakdowns were calculated and used for allocating time buffer into an initial, unbuffered schedule to absorb the impact of breakdowns. Our work differs from this line of work in two aspects. Firstly, as opposed to generating one baseline schedule, we construct a POS that represents a set of feasible schedules and hence is more flexible. Secondly, we compute both expected and variance values of durational extensions due to resource breakdowns and use them in designing chaining heuristics for improving robustness of POS (instead of one schedule). In [27] , reactive scheduling procedures were described to make feasible online choices that deviate as little as possible from the baseline schedule when disturbances occur. In our work, we solve the project scheduling problems under uncertainty proactively. In [24], proactive strategies were proposed to generate baseline schedule with or without resource or time buffers. A list scheduling based reactive strategy and a tabu search based heuristic were then applied to revert to a feasible schedule online. In that work, the interrupted activity has to be restarted from scratch each time it is interrupted, while in our work, we assume a preempt-resume setting. A possible extension of our work is to combine the online techniques proposed in [24] and [27] with our methods, thereby allowing us to implement proven robust execution strategies and good online behaviors.

3 Preliminaries

In this section, we briefly describe the notation, the scheduling models and the solution concepts relevant to this paper. A random variable will be denoted by $\tilde{x}$ and bold face lower case letters such as $\mathbf{x}$ represent vectors.

3.1 Deterministic RCPSP/max

The RCPSP/max [2] is defined as follows. There are N activities $\{a_1, a_2..., a_N\}$, where each activity a_i $(i = 1, ...N)$ is to be executed for a certain amount of time units without preemption. Each activity a_i has a fixed *duration* or *processing time* d_i, which is assumed to be a non-negative real number or non-negative integer number. In addition, "source" and "sink" activities a_0 and a_{N+1} with $d_0 = d_{N+1} = 0$ are introduced to represent the beginning and the completion of the project, respectively. A start time *schedule* $\boldsymbol{ss}$ is an assignment of start times to all activities $a_1, a_2..., a_N$, i.e. a vector $\boldsymbol{ss} = (st(a_1), st(a_2), ...st(a_N))$ where $st(a_i)$ represents the start time of activity a_i and $st(a_0)$ is assumed to be 0. Let $et(a_i)$ be the end time of activity a_i. Since durations are deterministic and preemption is not allowed, we then have

$$st(a_i) + d_i = et(a_i) \tag{1}$$

And the project *makespan* which is also the start time of the final "sink" activity $st(a_{N+1})$ equals

$$st(a_{N+1}) = max_i et(a_i) \tag{2}$$

Schedules are subject to two kinds of constraints, *temporal constraints* and *resource constraints.*

Temporal constraints restrict the time lags between activities. A *minimum time lag* T_{ij}^{min} between the start time of two different activities a_i and a_j enforces the following constraint:

$$st(a_j) - st(a_i) \geq T_{ij}^{min}. \tag{3}$$

Specially, $T_{ij}^{min} = 0$ means that activity a_j cannot be started before activity a_i begins. A *maximum time lag* T_{ij}^{max} between the start time of two different activities a_i and a_j ensures that the following condition holds:

$$st(a_j) - st(a_i) \leq T_{ij}^{max}. \tag{4}$$

$T_{ij}^{max} = 0$ means that activity a_j cannot be started after activity a_i begins. A schedule $\boldsymbol{ss} = (st(a_1), ...st(a_N))$ is *time feasible*, if all the time lag constraints are satisfied at the start times $st(a_i)$ $(i = 1, ...N)$.

A resource unit is reusable and available for another activity once it is no longer used by the current activity. Each type of resource, k $(k = 1, 2..., K)$ has a limited capacity, C_k. Each activity a_i requires r_{ik} units of resource of type k. Let $A(t) = \{i \in \{1, 2...N\} | st(a_i) \leq t \leq et(a_i)\}$ be the set of activities which are being processed at time instant t. A schedule is *resource feasible* if at each time instant t, the total demand for a resource k does not exceed its capacity C_k, i.e. $\sum_{i \in A(t)} r_{ik} \leq C_k$. A schedule $\boldsymbol{ss}$ is called *feasible* if it is both time and resource feasible. The objective of the deterministic RCPSP/max scheduling problem is to find a feasible schedule so that the project makespan is minimized.

3.2 RCPSP/max with Durational Uncertainty and Robust Makespan

As for durational uncertainty, we follow the same representation as in [15]: Irrespective of the distribution type of the deviation, the duration of an activity can be specified as a sum of its mean value, d_i^0 and its deviation, $\tilde{z}_i$ with mean value as 0 ($E[\tilde{z}_i] = 0$):

$$\tilde{d}_i = d_i^0 + \tilde{z}_i, \tag{5}$$

The random variables, $\{\tilde{z}_i\}$, corresponding to deviation are assumed to be independent of each other. This is a standard assumption that has been employed and justified in existing work (e.g, [3] and [7]).

Similar to the deterministic setting, time lags in the stochastic setting are also provided between start times of activities (*start-to-start*) and they remain deterministic. Unlike the deterministic case, other types of constraints (*end-to-start* etc.) cannot be converted into deterministic *start-to-start* constraints. However, it should be noted that our techniques will still be applicable to all types of time lag constraints. In this paper, for purposes of exposition, we present our techniques assuming the temporal dependencies are provided as *start-to-start* constraints.

In the deterministic setting, start time schedules can be computed and values of makespan can be used to evaluate the performance of the schedule. However, when uncertainty is involved, the project makespan becomes a random variable and the schedule is replaced by an execution strategy. *Partial Order Schedule* (POS) [32] is used as an execution strategy for the scheduling project under uncertainty.

Let $\tilde{S}_{N+1}(\mathbf{x}, \tilde{\mathbf{z}})$ be the random variable used to represent the starting time of the sink activity, a_{N+1} given a POS, $\mathbf{x}$ and stochasticity associated with durations of random variables, $\tilde{\mathbf{z}}$. Given a prescribed level of risk $0 < \varepsilon \leq 1$, we have the following key definitions relevant to the objective of solving RCPSP/max with durational uncertainty:

– Robust makespan of a POS $\mathbf{x}$, $\mathcal{M}_{\mathbf{x},\tilde{\mathbf{z}}}$: $(1-\varepsilon)$ quantile of the cumulative distribution function for $\tilde{S}_{N+1}(\mathbf{x}, \tilde{\mathbf{z}})$. Formally:

$$P(\tilde{S}_{N+1}(\mathbf{x}, \tilde{\mathbf{z}}) \leq \mathcal{M}_{\mathbf{x},\tilde{\mathbf{z}}}) \geq (1-\varepsilon) \quad \textbf{or}$$
$$P(\tilde{S}_{N+1}(\mathbf{x}, \tilde{\mathbf{z}}) \geq \mathcal{M}_{\mathbf{x},\tilde{\mathbf{z}}}) \leq \varepsilon \tag{6}$$

– Upper bound on robust makespan of a POS $\mathbf{x}$: Since, there is no specific information regarding the distributions of $\tilde{\mathbf{z}}$ and there exist complex interdependencies between activities during execution, it is not trivial to compute $\mathcal{M}_{\mathbf{x},\tilde{\mathbf{z}}}$ exactly. Hence, we use Chebyshev inequality based upper bound that will henceforth be represented as $\hat{\mathcal{M}}_{\mathbf{x},\tilde{\mathbf{z}}}$.

– Minimum robust makespan, $\mathcal{M}^*_{\tilde{\mathbf{z}}}$: Lowest value of robust makespan for any POS for the given RCPSP/max with durational uncertainty.

– Minimum upper bound on robust makespan: Given the limitation with respect to computing exact robust makespan, we can only compute minimum upper bound on robust makespan, henceforth referred to as , $\hat{\mathcal{M}}^*_{\tilde{\mathbf{z}}}$.

While the aim is to compute minimum robust makespan, due to the lack of exact distributions of uncertainty, the objective in solving a given RCPSP/max with durational uncertainty is to find the POS that has the minimum upper bound on robust makespan.

3.3 Partial Order Schedule

Partial Order Schedule (POS) was first proposed in [32]. It is defined as a set of activities, which are partially ordered such that any schedule with total activity order consistent with the partial order is resource and time feasible. A POS is represented by a graph where a node represents an activity and the edges represent precedence constraints between the activities. Once an activity partial order is generated, scheduling policies (e.g. ES-policy) [36] can help compute start-times by starting the activities as early as possible w.r.t. to original and partial order-based precedence constraints. Within a POS, each activity retains a set of feasible start times, which provide the flexibility to respond to unexpected disruptions.

A POS can be constructed from a given schedule via a chaining algorithm. Chaining is a procedure of dispatching activities to different resource units based on temporal and resource feasibility. A chain is a sequence of activities, which is associated with a single resource unit. During the chaining process, each activity can be allocated to one or more resource chains based on the resource requirement of the activity. Once an activity is scheduled to be executed on a resource unit, an additional edge (indicating precedence constraint) is added between the last activity of the selected chain and this activity so as to eliminate possible resource conflicts.

The basic chaining algorithm proposed in [32] can be described as follows. A feasible schedule is first obtained using a simple greedy heuristic. Consequently, the POS can be constructed through a chaining method as follows: First, the set of activities are sorted according to their start times given in the feasible solution; Then, all activities are allocated to different chains in that order, where each chain corresponds to a unit of a certain type of resource. A chain is called *available* for an activity if the end time of the last activity allocated on this chain is no greater than the start time of the activity in the feasible schedule. Once an activity is allocated on a chain, a precedence constraint between this activity and the last activity of the chain is posted. For those activities that require more than one unit of one or more types of resources, they will be allocated to a number of chains with the number equal to the overall number of resource units required by the activity.

To reduce inter-dependencies between activities as much as possible during the chaining procedure, two heuristics were developed in [31]: (a) Activities that require more than one resource units are allocated to the same subset of chains. (b) Activities with a precedence constraint defined in the original problem are allocated to the same set of chains. One direct advantage of such approaches is that synchronization points of a solution can be reduced so that flexibility can be improved.

3.4 Decision Rule - Segregated Linear Approximation (SLA)

In the context of project scheduling under uncertainty, we define a decision rule as the quantitative dependence of activity start times on uncertainty deviation in durations associated with other activities. As earlier, let the random variable $\tilde{S}_v(\mathbf{x}, \tilde{\mathbf{z}})$ denote the start time of activity v, where $\mathbf{x}$ represents the POS, and $\tilde{\mathbf{z}}$ represents the deviation in durations of activities. Following the linear decision rule framework in [4], variable $\tilde{S}_v(\mathbf{x}, \tilde{\mathbf{z}})$ is assumed to be affinely dependent on a subset of N primitive random variables (in this case, deviations in durations of activities) $\tilde{z}_k$ $(k = 1, ...N)$:

$$\tilde{S}_v(\mathbf{x}, \tilde{\mathbf{z}}) = c_v^0 + \sum_{k=1}^{N} c_{v,k}(\mathbf{x})\tilde{z}_k, \tag{7}$$

where c_v^0 represents the earliest start time of activity v under POS $\mathbf{x}$, $c_{v,k}(\mathbf{x})$ encodes how activity k is related to activity v in the POS.

In the work by Chen et al [8], each primitive random variable $\tilde{z}$ is represented by two *segregated* random variables $\tilde{z}^+$ and $\tilde{z}^-$:

$$\begin{aligned} &\tilde{z} = \tilde{z}^+ - \tilde{z}^-, \\ &\tilde{z}^+ = \max\{\tilde{z}, 0\}, \tilde{z}^- = \max\{-\tilde{z}, 0\}. \end{aligned} \tag{8}$$

Thus, the segregated linear approximation has the following general form:

$$\tilde{S}_v(\mathbf{x}, \tilde{\mathbf{z}}) = c_v^0 + \sum_{k=1}^{N} \{c_{v,k}^+(\mathbf{x})\tilde{z}_k^+ + c_{v,k}^-(\mathbf{x})\tilde{z}_k^-\}. \tag{9}$$

The idea of adopting the linear bound expression as an approximation technique in scheduling is to make the computation of robust makespan efficient. During the project execution, activities are either connected in series or in parallel, and the start time of an activity is only dependent on the starting times of all its preceding activities. Thus, all variables for modelling start times of activities can be defined as functions of other variables through the addition operator (to model serial activities) and/or the maximum operator (to model parallel activities).

We now formally describe the computation of $\tilde{S}(\mathbf{x}, \tilde{\mathbf{z}})$ by first representing durational uncertainty using segregated random variables and then deriving expressions for both the sum and maximum of random variables as linear approximation of segregated variables. Let d^0 denote the mean processing time of uncertain duration $\tilde{d}$, we can view the negative segregated random variable $\tilde{z}^- = max\{d^0 - \tilde{d}, 0\}$ as *earliness*, i.e., the difference of its processing time from the mean time d^0 when the activity is completed no later than its mean time. Similarly, we can represent the positive segregated random variable $\tilde{z}^+ = max\{\tilde{d} - d^0, 0\}$ as *lateness*, i.e., the difference of its processing time from the mean time d^0 when the activity is completed no earlier than its mean time. The uncertain duration $\tilde{d}$ is thus composed of three components: its *mean* d^0, *lateness* $\tilde{z}^+$, and *earliness* $\tilde{z}^-$,

$$\tilde{d} = d^0 + \tilde{z}^+ - \tilde{z}^-. \tag{10}$$

For a normally distributed duration, i.e., $\tilde{z} \sim N\{0, \sigma\}$, the respective values of mean and variance for the segregated variables can be summarized as:

$$\begin{aligned} &E[\tilde{z}^+] = E[\tilde{z}^-] = \sigma/\sqrt{2\pi}, \\ &Var[\tilde{z}^+] = Var[\tilde{z}^-] = (\pi - 1)\sigma^2/2\pi. \end{aligned} \tag{11}$$

The proof is provided in Appendix A.

In [21], a robust optimisation method for solving short-term production scheduling problems where uncertainty is characterised by specific probability distributions was presented. However, there are significant differences with out work: (a) Unlike their algorithms, our mechanisms are applicable for any kind of probability distribution as long as only the mean and variance of distributions are known; (b) We focus on a softer version of robustness, namely the alpha-robust makespan, whereas they focus on optimizing for the worst setting of uncertainty.

3.4.1 Serial Activities

Consider M activities connected serially, i.e., precedence constraints (either input or due to competing for same resource) across the M activities form a chain. The start time of activity v starting after the M-activity project can be expressed as:

$$\tilde{S}_v(\mathbf{x}, \tilde{\mathbf{z}}) = \sum_{i=1}^{M} (d_i^0 + \tilde{z}_i^+ - \tilde{z}_i^-). \tag{12}$$

Thus, the random variable $\tilde{S}_v(\mathbf{x}, \tilde{\mathbf{z}})$ has a mean value of $\sum_{i=1}^{M} d_i^0$ and uncertainty captured by a random variable, which has a positive segregated component of $\sum_{i=1}^{M} \tilde{z}_i^+$ and a negative segregated component of $-\sum_{i=1}^{M} \tilde{z}_i^-$.

3.4.2 Parallel Activities

Consider some set C of activities that are executed concurrently, the SLA decision rule for representing the start time of activity v that begins after those activities in set C is given by[1]

$$\tilde{S}_v(\mathbf{x}, \tilde{\mathbf{z}}) \leq \max_{i \in C}\{d_i^0\} + \sum\nolimits_{i \in C} \tilde{z}_i^+. \tag{13}$$

Thus, the random variable $\tilde{S}_v$ has an upper bound with mean value of $\max_{i \in C}\{d_i^0\}$ and uncertainty captured by a random variable with positive segregated components given by $\sum_{i \in C} \tilde{z}_i^+$ and no negative segregated component. By overlooking the earliness, we give a rough estimation with respect to the ending time of parallel activities. Note that such approximation is only applied in measuring the upper bound on robust makespan of POS $\mathbf{x}$ and not in deciding the real start times of activities when executing $\mathbf{x}$ online.

More specifically, the output of solving RCPSP/max involves a POS that is represented as a graph with activities as vertices and precedence constraints between activities as the edges. Given a POS graph, $\mathbf{x} = (V, E)$, where V is the set of activities and E is the set of temporal dependencies (an edge (u, v) represents a temporal dependency that states that activity u has to be completed before activity v can be started). For any activity $v \in V$, the decision rule for computing its start time is defined recursively as follows:

$$\tilde{S}_v(\mathbf{x}, \tilde{\mathbf{z}}) = \max_{(u,v) \in E}\{d_u^0 + \tilde{z}_u + \tilde{S}_u(\mathbf{x}, \tilde{\mathbf{z}})\}. \tag{14}$$

Eqn 14 is a recursive expression that is defined as a combination of *sum* and *maximum* on a set of random variables. Note that our aim here is to compute a POS, which is only an ordering of activities scheduled on various resources, with the robust makespan defined in Section 3.5. Through our work, we do not make any assumption on online execution like the exact start time of activities, as that can be quickly computed during the execution of the POS.

[1] It should be noted that we have used $\leq$ for comparing two stochastic variables. We have overloaded the $\leq$ operator and when we have $X \leq Y$, where X and Y are random variables, for all realizations of uncertainty, the outcome associated with random variable X will always be less than that of Y. Since the RHS ignores realizations of $\tilde{z}_i^-$, any realisation of actual start time, LHS (which considers the same realisations of $\tilde{z}_i^+, \forall i \in C$ as RHS along with a subtraction value in the realisation of $\tilde{z}_i^-, \forall i \in C$) would be less than the RHS.

Since, in both cases (*sum* and *max*) the decision variable $\tilde{S}_v(\mathbf{x}, \tilde{\mathbf{z}})$ can be expressed linearly on a subset of random segregated variables, the recursive computation in Eqn 14 is straightforward.

3.5 Upper Bound on Robust Makespan

Since we do not assume any specific distribution information for activity durations, computing the precise probability of successful execution of $\mathbf{x}$ is computationally very difficult. Hence, we compute an upper bound of the robust makespan for a given POS, $\mathbf{x}$. $\tilde{S}_{N+1}(\mathbf{x}, \tilde{\mathbf{z}})$ is the random variable used to represent start time of sink activity, a_{N+1}, and $\tilde{\mathbf{z}}$ represents the uncertainty about activity durations. From one sided Chebyshev's Inequality, we have:

$$\begin{aligned}\hat{\mathcal{M}}_{\mathbf{x},\tilde{\mathbf{z}}} \geq E[\tilde{S}_{N+1}(\mathbf{x}, \tilde{\mathbf{z}})] + \sqrt{\frac{1-\varepsilon}{\varepsilon}}\sqrt{Var[\tilde{S}_{N+1}(\mathbf{x}, \tilde{\mathbf{z}})]} \\ \implies P(\tilde{S}_{N+1}(\mathbf{x}, \tilde{\mathbf{z}}) \geq \hat{\mathcal{M}}_{\mathbf{x},\tilde{\mathbf{z}}}) \leq \varepsilon\end{aligned} \tag{15}$$

where $E[\cdot]$ and $Var[\cdot]$ represent the expected value and variance of the argument random variable/distribution respectively. We use the following tightest value of upper bound when performing local search, i.e.,

$$\hat{\mathcal{M}}_{\mathbf{x},\tilde{\mathbf{z}}} = E[\tilde{S}_{N+1}(\mathbf{x}, \tilde{\mathbf{z}})] + \sqrt{\frac{1-\varepsilon}{\varepsilon}}\sqrt{Var[\tilde{S}_{N+1}(\mathbf{x}, \tilde{\mathbf{z}})]}.$$

Thus, without knowing specific distribution, the objective of solving RCPSP/max with durational uncertainty can be rephrased as:

$$\mathbf{x}^* = \arg\min_{\mathbf{x}} \hat{\mathcal{M}}_{\mathbf{x},\tilde{\mathbf{z}}}$$

3.6 Robust Local Search Algorithm

This section will present how the decision rule approximation introduced by SLA is integrated with the robust makespan definition and local search mechanisms to provide a solution for the problems represented by RCPSP/max with durational uncertainty in [15]. Algorithm 1 provides the robust local search algorithm guided by decision rule using SLA. Given the RCPSP/max with durational uncertainty and the level of risk ($0 < \varepsilon \leq 1$), the algorithm returns a locally minimal POS in terms of upper bound on robust makespan.

Robust local search is performed on the neighborhood set of activity lists. An activity list (al) is defined as a precedence-constrained feasible sequence that is used by heuristics to generate earliest start time schedules in solving the standard RCPSP problem [22]. Different activity lists are explored by local moves. The set

Algorithm 1 Robust Local Search

```
1: Generate an activity list al randomly
2: Find a start time schedule, $ss$ randomly according to al
3: if al $\in F$ then
4:     $POS \leftarrow chaining(ss)$
5:     Compute $\hat{\mathcal{M}}_{now}$ according $POS$
6:     Update $\hat{\mathcal{M}}^*_{min}$ as $\hat{\mathcal{M}}_{now}$
7: else
8:     Record the first activity $a$ which cannot be scheduled
9: end if
10: for $i \leftarrow 1$ to Max_Iteration do
11:     if al $\in I$ then
12:         Shift activity $a$ ahead in al randomly as al'
13:     else
14:         Select two activities $b$ and $c$ in al randomly
15:         Swap $b$ and $c$ in al as al'
16:     end if
17:     Find randomized start time schedule $ss'$ according to al'
18:     if al$' \in F$ then
19:         $POS' \leftarrow chaining(ss')$
20:         Compute $\hat{\mathcal{M}}_{upd}$ according to $POS'$
21:         if al $\in I$ or $\hat{\mathcal{M}}_{upd} \leq \hat{\mathcal{M}}_{now}$ then
22:             $\hat{\mathcal{M}}_{now} \leftarrow \hat{\mathcal{M}}_{upd}$
23:             al $\leftarrow$ al$'$
24:             if $\hat{\mathcal{M}}_{upd} \leq \hat{\mathcal{M}}^*_{min}$ then
25:                 $\hat{\mathcal{M}}^*_{min} \leftarrow \hat{\mathcal{M}}_{upd}$
26:             end if
27:         end if
28:     else if al $\in I$ then
29:         al $\leftarrow$ al$'$
30:     else
31:         $p \leftarrow rand(0, 1)$
32:         if $p < 0.01$ then
33:             al $\leftarrow$ al$'$
34:             Record the first activity $a$ which cannot be scheduled
35:         end if
36:     end if
37: end for
```

of activity lists which result in feasible (or infeasible) schedules is defined as F (or I).

The procedure starts by randomly generating an activity list **al** (Line 1), which is a sequence of activities. In Line 2, a schedule ss is produced based on ordering of activities in the activity list **al**. It should be noted that in generating **al**, the order is derived from only the minimum time lags. However, when we translate RCPSP/max into a distance graph in schedule generation (Line 2), both minimum and maximum time lags are taken into account. The schedule generation works as follows: based on the activity list, we first perform domain reduction on the distance graph using the Floyd-Warshall algorithm, so that the feasible range of the start time for each activity based on the temporal constraints can be obtained. We then schedule each activity sequentially based on the order position in the activity list. For each activity, we first pick a start time randomly from the feasible domain and evaluate resource constraints for the duration of that activity (i.e. check if the resource amount used by that activity exceeds the current resource capacity). Once the start time for an activity is set, we run the shortest path algorithm to reduce domains for the remaining activities and update current resource capacity. If the resource constraints are not satisfied, we will try to set the start time randomly again for a prescribed maximum numbers of retries. Once the start time of current activity is set, we proceed iteratively to the next activity according to the activity list until the whole schedule is generated.

For addressing RCPSP/max with uncertainty, our focus is on exploring POS which is chained from a deterministic schedule. In Line 4, *chaining*() is employed to generate a POS from the generated baseline schedule. $Max\_Iteration$ refers to the maximum number of iterations in the robust local search. Two different types of local moves are applied here. To converge quickly to an activity list in F, the first local move is designed to schedule the activity that is causing a temporal or resource conflict to an earlier time. It will randomly shift ahead the first activity which cannot be scheduled in the current activity list (Line 12). When an activity list is in F, the second local move will randomly pick two activities and swap them in the current activity list, while satisfying the non-negative minimal time lag constraints (Lines 14-15). The move will be accepted, if it results in a smaller or equal $\hat{\mathcal{M}}_{upd}$ value (Lines 18-29). To explore different activity lists, a small probability will be included to accept the move which leads to an infeasible schedule (Lines 31-35). The probability to move from an activity list in F to one in I is set to 0.01. The minimal value will be saved as $\hat{\mathcal{M}}^*_{min}$.

4 Accounting for Resource Breakdowns in RCPSP/max

We extend the above local search framework with considerations on the impact of unexpected resource breakdowns to the robust makespan of an execution strategy.

4.1 Representation of Resource Breakdowns

Similar to existing research on resource constrained scheduling [26], we assume resources are renewable. Our approaches will compute a resource allocation for activities *before* project execution and this resource assignment remains fixed until the project finishes.

For a resource unit j of resource type k, we model the time between failures as a random variable X_{jk} and the time needed to repair that unit as a random variable Y_{jk}. Suppose that both X_{jk} and Y_{jk} are exponentially distributed. Then the corresponding cumulative distributions $F_{X_{jk}}(x)$ and $G_{Y_{jk}}(y)$ are given by:

$$F_{X_{jk}}(x) = 1 - e^{-\lambda_{jk} x}, \quad G_{Y_{jk}}(x) = 1 - e^{-\mu_{jk} y}, \tag{16}$$

where $1/\lambda_{jk}$ and $1/\mu_{jk}$ are expected values of X_{jk} and Y_{jk}, respectively.

4.2 Analytical Effect of Resource Breakdown

Our approach is to increase durations of activities to account for resource breakdowns. Extending on existing research [26], we will now analytically measure the impact of resource breakdowns on an activity's duration. Given that we have a fixed resource assignment to each activity, we can precisely translate a resource breakdown as delay in the activity currently using it. Note that since we also consider the activity to already have an innate duration variability, this adds further variability to the activity. (We will consider both in Section 6). Where we depart from [26] in our analysis is that we also measure the variance in addition to the mean values of the resulting duration variability.

We first assume that resource breakdowns can only occur during activity execution. For simplicity, we assume only one resource unit can break down at any time. Whenever a resource unit breaks down, it will be immediately sent for repair before it becomes available again. In the meantime, the activity on executing with a resource unit when it breaks down is interrupted and has to wait for the resource to be repaired. It can resume from the point where execution was interrupted. Note that throughout the lifetime of an activity, there may be multiple interruptions and we assume that any two interruptions are independent from each other.

Based on duration representation of activity a_i with process time variability in Eqn 5, we now model activity duration with the additional variability from resource breakdowns as:

$$\tilde{d_i} = d_i^0 + \tilde{z}_i + \tilde{\delta_i}, \tag{17}$$

where d_i^0 is the deterministic duration when no interruption occurs, $\tilde{z}_i$ being natural variability inherent in natural process time, and $\tilde{\delta_i}$ represents variability from breakdowns. Thus, $\tilde{d_i}$ is the real duration of activity a_i under interruptions due to both natural variability and variability from resource breakdowns.

Let N_i represent the number of interruptions due to resource breakdowns during the execution of activity a_i and R_{ij} describe the time that activity a_i waits before it resumes at the j^{th} interruption. According to definitions, we have the following relationships:

$$\tilde{\delta_i} = \Sigma_{j=0}^{N_i} R_{ij}, \tag{18}$$

where $\tilde{\delta_i}$ is the total delays due to resource breakdowns and R_{i0} equals to zero, which represents the case that no resource breakdown occurs during the execution of activity a_i. For simplicity, due to the independence relationship we assumed before, we henceforth rewrite R_{ij} as R_i. Then, from Equation 18, we can calculate the expected value of $\tilde{\delta_i}$ as follows:

$$E[\tilde{\delta_i}] = \sum\nolimits_{j=0}^{\infty} jE[R_i]P(N_i = j) = E[R_i]E[N_i]. \tag{19}$$

To further compute the variance of $\tilde{\delta_i}$, we need the following Lemma:

Lemma 41. *[34] Let $X_1, X_2, ...X_N$ be independent and identically distributed random variables with mean $E[X]$ and variance $Var[X]$. The sum has the type*

$$T = \sum\nolimits_{i=1}^{N} X_i,$$

where N *is a random variable with a finite mean and variance and X_i are independent of* N*. Then the variance value of* T *is*

$$Var[T] = [E[X]]^2 Var[N] + E[N]Var[X].$$

From Lemma 41, the variance of $\tilde{\delta_i}$ can be calculated as:

$$Var[\tilde{\delta_i}] = [E[R_i]]^2 Var[N_i] + E[N_i]Var[R_i]. \tag{20}$$

From Eqns 19 and 20, mean and variance values of $\tilde{\delta_i}$ directly depend on mean and variance values N_i and R_i. Appendix B provides the detailed calculation of required probabilistic property on N_i and R_i. Based on those information, values of $E[\tilde{\delta_i}]$ and $Var[\tilde{\delta_i}]$ can be determined as:

$$\begin{aligned} E[\tilde{\delta_i}] &= d_i^0 \sum_{k=1}^{K} \sum_{j=1}^{r_{ik}} \frac{\lambda_{jk}}{\mu_{jk}}, \\ Var[\tilde{\delta_i}] &= d_i^0 \sum_{k=1}^{K} \sum_{j=1}^{r_{ik}} \frac{2\lambda_{jk}}{(\mu_{jk})^2} + \sigma_i^2 (\sum_{k=1}^{K} \sum_{j=1}^{r_{ik}} \frac{\lambda_{jk}}{\mu_{jk}})^2. \end{aligned} \tag{21}$$

5 Resource Assignment Heuristics for Chaining

We provide a simplified overview of our robust local search in Fig 1 to demonstrate the role of chaining procedure and resource assignments heuristics. Typically, there exist multiple feasible resource allocations for every activity during the chaining procedure (for constructing a POS). It should be noted that different assignments can lead to different execution strategies (or POSs), each with a different level of robustness. Thus, robust makespan may be affected with varying resource assignments. In [16], resources are firstly ranked based on the mean time to breakdown $(1/\lambda)$ and activities are greedily allocated to the first available resource based on that order. However, such a greedy assignment is not

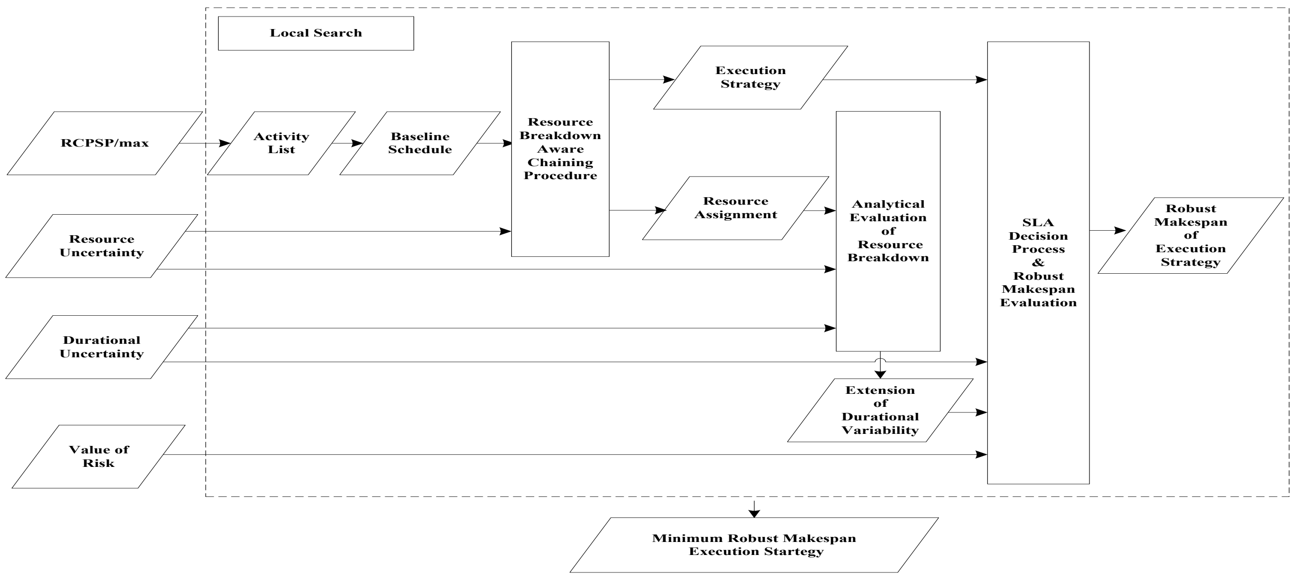

Local Search
RCPSP/max
Resource Uncertainty
Durational Uncertainty
Value of Risk
Activity List
Baseline Schedule
Resource Breakdown Aware Chaining Procedure
Execution Strategy
Resource Assignment
Analytical Evaluation of Resource Breakdown
Extension of Durational Variability
SLA Decision Process & Robust Makespan Evaluation
Robust Makespan of Execution Strategy
Minimum Robust Makespan Execution Startegy

Fig. 1 Overview of Local Search.

sensitive to resource repair distributions. Therefore, we build on the latest chaining technique for POS [31] by proposing heuristics for assigning activities to resource chains (or units) based on both breakdown (λ) and repair (μ) distribution parameters.

A key observation in this regard is the dependence of Equation 21 on the terms λ/μ and λ/μ^2. This implies that expected value and variance of strategy robustness are dependent directly on uncertainty parameters relevant to resource breakdowns. Based on this observation, we propose our chaining algorithm called Resource Breakdown Aware Chaining. Intuitively, we will assign longer duration activities to less unreliable resources, where the unreliability is characterized by any of the three heuristics:

- Mean Heuristic (MH), where unreliability is defined by λ/μ
- Variance Heuristic (VH), where unreliability is defined by λ/μ^2
- Mean and Variance Heuristic (MVH), where unreliability is a combination of λ/μ and λ/μ^2.

Algorithm 2 provides the pseudo code for resource breakdown aware chaining. Lines 4 - 8 and Lines 20 - 29 constitute the chaining technique [31] introduced in Section 3 that aims to improve the flexibility (or parallelism). We improve this chaining procedure to incorporate sensitivity to resource breakdowns in Lines 9 - 18. Depending on the exact heuristic employed, the last activity in a resource chain (or unit) is swapped so that a longer duration activity is scheduled on a more robust resource.

Algorithm 2 Resource Breakdown Aware Chaining, RBAC (Activity a, Schedule ss, Heuristic h)

```
1: $C(a) \leftarrow$ Find set of available chains, $C(a)$ for activity $a$ based on $S$
2: $U(a) \leftarrow$ Find set of unavailable chains, $U(a)$ for activity $a$ based on $S$
3: $P(a) \leftarrow$ Collect chains from $C(a)$ with last activity of chain preceding $a$ in problem
4: if $P(a) \neq \phi$ then
5:     $k \leftarrow$ Get an available chain in $P(a)$
6: else
7:     $k \leftarrow$ Get an available chain in $C(a)$
8: end if
9: Let $b$ denote the last activity of chain $u$
10: if $\exists\ u \in U(a),\ k \in C(b), dur_a > dur_b$ then
11:     if h = "MH", $\lambda_k/\mu_k > \lambda_u/\mu_u$ then
12:         swap activity $b$ onto chain $k$, $k \leftarrow u$
13:     else if h = "VH", $\lambda_k/\mu_k^2 > \lambda_u/\mu_u^2$ then
14:         swap activity $b$ onto chain $k$, $k \leftarrow u$
15:     else if h = "MVH",$\lambda_k/\mu_k > \lambda_u/\mu_u$, $\lambda_k/\mu_k^2 > \lambda_u/\mu_u^2$ then
16:         swap activity $b$ onto chain $k$, $k \leftarrow u$
17:     end if
18: end if
19: Post constraint between between last activity of chain $k$ (denoted as $last(k)$) and activity $a$
20: if $a$ requires more than one resource unit then
21:     $C1(a) \leftarrow$ chains in $C(a)$ which have last activity as last(k)
22:     $C2(a) \leftarrow C(a) \setminus C1(a)$
23:     for all resource units required by $a$ do
24:         choose an available chain belonging to $C1(a)$
25:         if chain above is not feasible then
26:             choose an available chain belonging to $C2(a)$
27:         end if
28:     end for
29: end if
```

6 Extended Robust Local Search Framework

We now extend the robust local search framework with additional considerations on the impact of resource breakdowns to the upper bound on robust makespan of a given POS. First, we revise the segregated linear decision rules introduced in Section 3.4. The idea is to be able to separate the effects of the innate duration variability and the delay due to resource breakdowns. For this purpose, we introduce two notions: a real activity and a dummy activity. For each activity a_i, we divide it into a "real" activity a_i' and a "dummy" activity a_i'' as illustrated in Figure 2. Thus, duration of an activity is

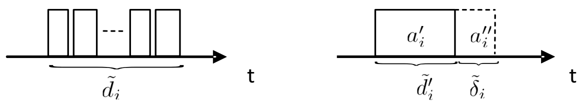


Fig. 2 "Real" Activity and "Dummy" Activity.

a sum of duration of its "real" part and the duration of the "dummy" part. i.e. $\tilde{d}_i = \tilde{d}_i' + \tilde{\delta}_i$, where the duration of its dummy activity is modeled as $\tilde{\delta}_i$. We then model the duration of the real part as:

$$\tilde{d}_i' = d_i^0 + \tilde{z}_i = d_i^0 + \tilde{z}_i^+ - \tilde{z}_i^-, \tag{22}$$

where the segregated random variables $\tilde{z}_i^+$ and $\tilde{z}_i^-$ represent lateness and earliness, respectively.

In a scheduling context, an activity will start either after the end of an activity (i.e. in series) or after the end of multiple activities occurring simultaneously (i.e. in parallel). Now we describe the computation of $\tilde{S}_v(\mathbf{x}, \tilde{\mathbf{z}}, \tilde{\delta})$ (compared with $\tilde{S}_v(\mathbf{x}, \tilde{\mathbf{z}})$ under only durational uncertainty in Section 3.4) by representing both resource and durational uncertainties as follows.

– Serial Activities: The start time of activity v starting after the M-activity project is expressed as:

$$\tilde{S}_v(\mathbf{x}, \tilde{\mathbf{z}}, \tilde{\delta}) = \sum_{i=1}^{M} (d_i^0 + \tilde{\delta}_i + \tilde{z}_i^+ - \tilde{z}_i^-). \tag{23}$$

Mean and variance of the segregated variables $\tilde{z}^+$ and $\tilde{z}^-$ are known, and mean and variance of variation $\tilde{\delta}_i$ can be computed according to Eqn 21, hence mean and variance of $\tilde{S}_v$ can be easily computed.

– Parallel Activities: Consider some set C of activities that are executed concurrently, the upper bound on the start time of activity v starting after the parallel project network is represented as follows:

$$\tilde{S}_v(\mathbf{x}, \tilde{\mathbf{z}}, \tilde{\delta}) \leq \max_{i \in C}\{d_i^0\} + \sum_{i \in C} \tilde{z}_i^+ + \sum_{i \in C} \tilde{\delta}_i. \tag{24}$$

Note that by summing up delays, we are computing an expression for upper bound of the ending time of any parallel project networks. Similarly, mean and variance of the segregated variables are known, and mean and variance of variation $\tilde{\delta}_i$ can be computed according to Eqn 21, hence mean and variance values of $\tilde{S}_v$ are easy to compute.

Based on the segregated linear decision rule under both resource and durational uncertainties, the upper bound on robust makespan of POS $\mathbf{x}$ which will be used in our local search framework is defined as follows:

Definition 1. *Given $0 < \varepsilon \leq 1$ and the start time of sink node given by $\tilde{S}_{N+1}(\mathbf{x}, \tilde{\mathbf{z}}, \tilde{\delta})$, the upper bound on robust makespan, $\hat{\mathcal{M}}_{\mathbf{x}, \tilde{\mathbf{z}}, \tilde{\delta}}(\varepsilon)$, is defined as*

$$E[\tilde{S}_{N+1}(\mathbf{x}, \tilde{\mathbf{z}}, \tilde{\delta})] + \sqrt{\frac{1-\varepsilon}{\varepsilon}} \sqrt{Var[\tilde{S}_{N+1}(\mathbf{x}, \tilde{\mathbf{z}}, \tilde{\delta})]}. \tag{25}$$

We are now ready to present our improved Robust Local Search which is capable of handling resource and durational uncertainties. The goal of the local search mechanism is to find a local minimum of upper bound on robust makespan, $\hat{\mathcal{M}}^*_{\tilde{\mathbf{z}}, \tilde{\delta}}$. The detailed description of our Robust Local Search with Resource Breakdown Aware Chaining and MVH heuristic (RLS_{MVH}) is presented in Algorithm 3.

Algorithm 3 RLS_{MVH}

```
1: rank resource units in non-increasing order of reliability
2: initialize activity list $AL$
3: set $i \leftarrow 0; \hat{\mathcal{M}}^*_{min} \leftarrow \infty$
4: while $i < Max\_Iteration$ do
5:     generate baseline schedule $ss$ according to $AL$
6:     initialize the POS
7:     for all $i \leq N$ do
8:         call RBAC($a_i$, $ss$, "MVH") to add to $POS$
9:     end for
10:    calculate robust makespan $\hat{\mathcal{M}}_{now}$ according to $POS$
11:    if $\hat{\mathcal{M}}_{now} < \hat{\mathcal{M}}^*_{min}$ then
12:        set $\hat{\mathcal{M}}^*_{min} \leftarrow \hat{\mathcal{M}}_{now}$
13:    end if
14:    $i \leftarrow i + 1$
15:    apply local move on $AL$ to generate new $AL$
16: end while
17: return $\hat{\mathcal{M}}^*_{min}$
```

The worst-case complexity of the robust local search algorithm RLS_{MVH} is given as follows. The sorting process for resources with respect to reliability costs $O(K \cdot C^{max} \cdot log(K \cdot C^{max}))$, where K is the number of types of resources and C^{max} is the maximum capacity among all resources. At each iteration of local search, baseline schedule generation costs $O(N \cdot (N^3 + H \cdot K \cdot L))$, where N is the number of activities to be scheduled, H is the maximum planning horizon[2], and L is the prescribed

[2] Maximum planning horizon is typically computed by assuming activities will be executed in a series and is the sum of durations of activities along with temporal lags

maximum number of retries for each activity on setting the randomized start time. The cost for constructing and evaluating POS is $O(N \cdot logN + N \cdot K \cdot C^{max})$. Thus, the worst-case complexity of RLS_{MVH} is

$$O(K \cdot C^{max} \cdot log(K \cdot C^{max}) + T \cdot N^4 + T \cdot N \cdot H \cdot K \cdot L + T \cdot N \cdot K \cdot C^{max}), \quad (26)$$

where T is the number of iterations in local search[3].

7 Discussions on Infeasibility

In this section, we provide discussions on cases where we encounter infeasibility due to uncertain durations and/or unreliable resources.

Firstly, the maximum time lags between activities can be violated for some of the realizations of activity durations and/or unreliable resources. For such realizations, there exists no feasible schedule and hence makespan is infinity. However, since we are interested in robust makespan (a value of makespan that is greater than $(1-\varepsilon)$*100 percentage of all instances), the value robust makespan returned by our approach is correct as long as the level of risk, ε includes the probability of seeing infeasible instances.

Secondly, it is possible that the POS constructed from a baseline schedule corresponding to RCPSP/max with expected duration generates schedules that are not executable. The inexecutability arises due to a bad ordering of activities in the POS that violate maximum time lags (due to construction from expected durations). This type of infeasibility was discussed in our earlier work [15] in great detail.

To understand the impact of both these infeasibilities, we simulated the execution of POS provided by our approach 500 times and computed the probability of encountering an infeasibility due to either one of the two reasons described above. We did this for all 270 problem instances in each of our three benchmark sets J10, J20 and J30. We observe that all benchmark problems have a very small set of infeasible instances: the expected value and variance of the infeasibility probability over all instances in J10 were 0.0578 and 0.0075, in J20 are 0.0506, 0.0059, in J30 are 0.0406 0.0047, respectively. That is to say, only 4-6% of the instances were infeasible and by including this in our risk value, ε we are able to account for these infeasibilities.

[3] Since the complexity is linear in H, in the worst case, our algorithm has pseudo polynomial complexity. However, if H is within a constant factor of N, the worst complexity will be linear

8 Experimental Evaluation

In this section, we compare the quality of the execution strategies generated by our robust local search approach with the new chaining algorithms against robust local search with existing chaining algorithms.

8.1 Experimental Setup

The instances considered for RCPSP/max with resource and durational uncertainties were obtained by extending the three benchmark sets available for RCPSP/max, J10, J20 and J30 as specified in PSPLib [23]. Each set contains 270 instances with duration for each activity ranging between 1 and 10. The maximum number of activities for J10, J20 and J30 are 10, 20 and 30, respectively. For each activity a_i, we set the fixed part d_i^0 of the stochastic duration as the corresponding deterministic duration given by benchmarks, and assume durational uncertainty is normally distributed, i.e. $\tilde{z}_i \sim N(0, \sigma)$. We set $\sigma = 0.1$ and run algorithms on four increasing levels of risk $\varepsilon = \{0.01, 0.05, 0.1, 0.2\}$. As for resource uncertainty, we set λ and μ each randomly picked from one of the intervals, respectively: $1/\lambda \in \{[5, 15], [15, 25], [25, 35]\}$, $1/\mu \in \{[0.25, 0.33], [0.33, 0.5], [0.5, 1]\}$. Intuitively, $1/\lambda$ and $1/\mu$ represent a decent range of values for the expected value of the time between breakdowns and the time for repair.

We compare robust local search with different chaining procedures: a) (RLS_{MH}) is the robust local search with mean heuristic chaining ; b) (RLS_{VH}) is the robust local search with variance heuristic chaining; c) (RLS_{MVH}) is the robust local search with combined mean variance chaining; d) (RLS_{FBC}) refers to the robust local search with Forward-Backward Chaining in [16].

The number of local search iterations for robust local search was set to 1000. To reduce the stochasticity effects of robust local search, we average over 10 random executions for each problem instance. Our code was implemented in C++ and executed on a Core(TM)2 Duo CPU 2.33GHz processor under FedoraCore 11 (Kernel Linux 2.6.29.4-167.fc11.i586).

8.2 Comparisons between Chaining Procedures

To test the performance of our proposed chaining procedures, we compare the average robust makespans of 270 problem instances under different levels of risk. Table 1 shows how different chaining procedures and risk levels ε affect robust makespan over different data sets, each of which are generated randomly with $1/\lambda$ and $1/\mu$

set to [5,15] and [0.5,1], respectively. While we considered one set of values for μ and λ, we observed similar results for other range of values. All runs on all instances of J10, J20 and J30 completed in the less than 10 seconds each. Hence, we do not provide a detailed comparison with respect to run-times. Here are the key conclusions:

– Irrespective of problem scale, as the level of risk ε increases, the value of robust makespan decreases with all four algorithms. This is an expected result, where the lower risk that the scheduler is willing to take, the higher is the robust value of generated execution strategy;
– Irrespective of problem scale and the level of risk, our proposed methods (RLS_{MH},RLS_{VH} and RLS_{MVH}) can provide better robust makespan than results reported in [16]. This indicates that considering robustness chaining helps improving robustness of generated execution strategy;
– Irrespective of problem scale and the level of risk, RLS_{MVH} performs better than $RLS_{MH}$and RLS_{VH}, but we do not see any superiority between RLS_{MH} and RLS_{VH}.

Prob	Algo	$\varepsilon = 0.2$	$\varepsilon = 0.1$	$\varepsilon = 0.05$	$\varepsilon = 0.01$
J10	RLS_{MH}	86.13	90.47	96.36	120.55
	RLS_{VH}	87.28	91.68	97.65	122.22
	RLS_{MVH}	84.43	88.76	94.65	118.83
	RLS_{FBC}	90.13	94.60	100.68	125.67
J20	RLS_{MH}	274.02	280.17	288.54	322.92
	RLS_{VH}	255.76	261.86	270.15	304.26
	RLS_{MVH}	241.03	247.16	255.5	289.74
	RLS_{FBC}	328.68	334.93	343.42	378.27
J30	RLS_{MH}	337.44	344.7	354.55	395.1
	RLS_{VH}	331.12	338.36	348.2	388.68
	RLS_{MVH}	309.78	317.03	326.89	367.43
	RLS_{FBC}	353.58	360.97	371.01	412.21

Table 1 Comparison of RLS_{MH}, RLS_{VH}, RLS_{MVH} against RLS_{FBC}.

8.2.1 Effect of Breakdown Parameters on Robustness

We now show the effect of different breakdown parameters on robustness of generated POSs. We consider all the values of μ and λ shown in Table 2. While the risk value ε is set to 0.2 for these results, similar results were observed for other epsilon values as well.

The key conclusion is that the minimum robust makespan increases when λ increases, and it decreases when μ increases, as mean and variance parts of durational extension due to resource breakdowns increase when λ increases according to Eqn 21, while the values decrease when μ increases.

Prob		$1/\mu \in$	*[0.25,0.33]*	*[0.33,0.5]*	*[0.5,1]*
J10	$1/\lambda \in$	*[5,15]*	57.37	63.4	84.43
		[15,25]	49.82	52.18	58.1
		[25,35]	48.33	50.1	54.43
J20	$1/\lambda \in$	*[5,15]*	95.86	123.08	241.03
		[15,25]	81.99	85.61	96.48
		[25,35]	79.55	82.31	89.41
J30	$1/\lambda \in$	*[5,15]*	143.8	187.26	309.78
		[15,25]	106.41	111.11	143.58
		[25,35]	103.71	107.14	116.41

Table 2 Minimum Robust Makespan with Different Breakdown Parameters.

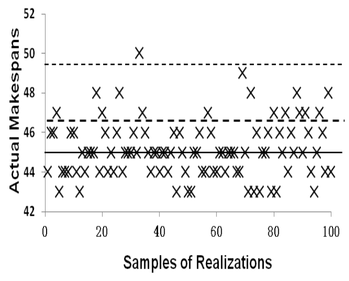
8.3 Simulation

Next, we evaluate the quality of obtained execution strategies with respect to the difference between robust makespans calculated by our approach and the makespans that are actually achieved when executing the schedules obtained from our execution strategies. For this purpose, we first generate 100 scenarios of durational variability (normally distributed with mean 0 and standard deviation 0.1) and resource breakdowns. Each scenario of resource uncertainty corresponds to the pair of time between breakdowns and repair time exponentially distributed with with $1/\lambda$ and $1/\mu$ set to [15,25] and [0.5,1], respectively. We test all 270 instances of each benchmark set with the level of risk $\varepsilon = 0.2$ and obtain the respective POSs. Then we compute the actual makespans of schedules derived from the respective POSs under the given realization samples.

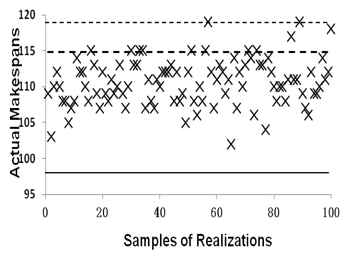
Figure 3 provides the simulation results for three randomly chosen instances from J10, J20 and J30 benchmark sets and similar results can also be obtained from other instances. In each graph, we provide three lines: computed f'^*, actual f'^* by simulation and deterministic makespan; and the actual realization for each of the 100 uncertainty realizations. It should be noted that even in the case with a variance of 0.1, the worst case gap between calculated and actual makespans is not significant.

9 Conclusion

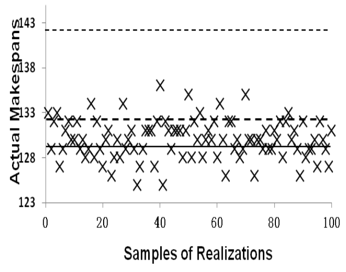
In this paper, we provided a scalable architecture for solving RCPSP/max with stochastic durations and unreliable resources. The major contribution of the paper is the Resource Breakdown Aware Chaining procedure with three different metrics (MH, VH and MVH) to obtain execution strategies based on resource breakdown and repair distributions. Furthermore, we have provided formal mechanisms for (i) predicting the effect of resource breakdowns and repairs on the minimum ro-



(a) J10 Instance.



(b) J20 Instance.



(c) J30 Instance.

Fig. 3 Comparison of Gap between Actual Minimum Robust Makespan and Minimum Robust Makespan Calculated from Robust Local Search RLS_{MVH}. The cross marks indicate the Actual Makespans for different scenarios of durational and resource uncertainty. Various lines starting from the top indicate: Computed f'^*, Actual f'^* by Simulation, and Deterministic Makespan.

bust makespan; (ii) suggesting more robust resource allocations. Experimental results illustrate improved robustness of execution strategies with the new chaining technique.

There are two key assumptions in this research that we hope to relax in the future:

– Firstly, we have assumed exponential distributions for times between resource breakdowns and repair times once resources break down. An immediate relaxation of this assumption would be to address RCPSP/max with only mean and variance of resource uncertainty (rather than any specific distribution). Furthermore, different decision rules contributed by [15] for computing schedules with respect to execution strategy and realizations of uncertainty would also be applied for generating robust execution strategies.
– While we consider different resource allocations during the chaining procedure, the resource assignment will remain fixed during execution once it is generated. We hope to relax this using the online approaches contributed in [24] and [27].

Appendix

A Proof of Eqn 11

From Eqn 8,we can express $\tilde{z}^+$ and $\tilde{z}^-$ by the following set of equations:

$$\tilde{z}^+ = (\tilde{z} + |\tilde{z}|)/2, \quad \tilde{z}^- = (|\tilde{z}| - \tilde{z})/2. \tag{27}$$

As $\tilde{z} \sim N\{0, \sigma\}$, $|\tilde{z}|$ follows half-normal distribution with the expected value and variance given by,

$$E[|\tilde{z}|] = 2\sigma/\sqrt{2\pi}, \quad var[|\tilde{z}|] = \sigma^2(1 - 2/\pi). \tag{28}$$

From definitions of $\tilde{z}^+$, $\tilde{z}^-$ and Equation 28, we have,

$$\begin{aligned} &E[\tilde{z}^+] = E[\tilde{z}^-] = \sigma/\sqrt{2\pi}, \\ &Var[\tilde{z}^+] = E[(\tilde{z}^+)^2] - (E[\tilde{z}^+])^2 = (\pi - 1)\sigma^2/2\pi. \end{aligned} \tag{29}$$

Similarly, $Var[\tilde{z}^-] = (\pi - 1)\sigma^2/2\pi$.

B Proof of Eqn 21

To determine values of $E[\tilde{\delta_i}]$ and $Var[\tilde{\delta_i}]$, we need to calculate mean and variance values of N_i and R_i. The detailed mathematical derivation will be shown in the following subsections.

B.1 Number of Interruptions.

In this subsection, we analyze random variable N_i describing the total number of interruptions due to resource breakdowns experienced by activity throughout its execution.

Lemma B1. *[20] Let $X_1, X_2, ...X_n$ be independent exponential random variables with parameters $\lambda_1, \lambda_2, ...\lambda_n$, respectively. Then the minimum of these random variables follows an exponential distribution; that is,*

$min\{X_1, X_2, ...X_n\} \sim EXP(\sum_{i=1}^{n} \lambda_i)$.

During execution, activity a_i would be interrupted as soon as one of these resource units allocated to a_i breaks down. Suppose T_i is a random variable representing the minimum time to failure over all resource units allocated to activity a_i. We can represent T_i as follows.

$$T_i = min\{X_{11}, ..., X_{r_{i1}1}, ..., X_{1K}, ..., X_{r_{iK}K}\}. \tag{30}$$

We then rewrite T_i as:

$$T_i = min\{M_1, M_2, ..., M_K\}, \tag{31}$$

where M_k is a random variable representing the minimum time to breakdown of all resource units of resource type k needed by activity a_i, and $M_k = min\{X_{1k}, X_{2k}, ...X_{r_{ik}k}\}$.

Since $X_{jk} \sim EXP(\lambda_{jk})$, using Lemma B1, we have,

$$M_k \sim EXP(\textstyle\sum_{j=1}^{r_{ik}} \lambda_{jk}). \tag{32}$$

Similarly, following Eqns 31, 32 and Lemma B1, we can see that T_i is also exponentially distributed with the parameter $\sum_{k=1}^{K} \sum_{j=1}^{r_{ik}} \lambda_{jk}$, so that

$$E[T_i] = 1/\textstyle\sum_{k=1}^{K} \sum_{j=1}^{r_{ik}} \lambda_{jk}. \tag{33}$$

Without consideration of resource variability, duration of activity i is represented as the sum of its mean d_i^0 and natural variability $\tilde{z}_i$ with zero mean, i.e.,

$$E[\tilde{z}_i] = 0. \tag{34}$$

Let $\{z_i\}$ denote the realization set of durational variability $\tilde{z}_i$. Thus, for a certain realization z_i, duration of activity a_i is a deterministic value, i.e., $d^0 + z_i$.

Lemma B2. *[20] Suppose the time between consecutive occurrences of arrivals follows an exponential distribution with parameter λ. Let* X(t) *be the number of occurrences by time* t*, then* X(t) *follows a Poisson distribution with parameter $\lambda * t$.*

Using Lemma B2 by Hillier and Lieberman [20], the number of interruptions of activity a_i during its execution $d_i^0 + z_i$, follows a Poisson distribution with the mean number of occurrences given by $(d_i^0 + z_i)/E[T_i]$. In other words, mean and variance values for the number of interruptions for the realization are:

$$E[N_i|z_i] = Var[N_i|z_i] = (d^0 + z_i)\sum_{k=1}^{K}\sum_{j=1}^{r_{ik}} \lambda_{jk}. \tag{35}$$

The above shows the analysis on the number of interruptions for fixed duration, now we extend that with consideration of stochastic duration. According to Law of Total Expectation [6], the expected value of N_i is the same as the conditional expected value of N_i given $\tilde{z}_i$, i.e.,

$$E[N_i] = E_{\tilde{z}_i}[E_{N_i|\tilde{z}_i}[N_i|\tilde{z}_i]]. \tag{36}$$

Based on the representation of the mean number of interruptions for the deterministic case in Eqn 35 and Eqn 34, mean of N_i can be calculated as:

$$\begin{aligned} E[N_i] &= E_{\tilde{z}_i}[E_{N_i|\tilde{z}_i}[N_i|\tilde{z}_i]] \\ &= E_{\tilde{z}_i}[(d^0 + \tilde{z}_i)\sum_{k=1}^{K}\sum_{j=1}^{r_{ik}} \lambda_{jk}] \\ &= d^0 \sum_{k=1}^{K}\sum_{j=1}^{r_{ik}} \lambda_{jk}. \end{aligned} \tag{37}$$

According to Law of Total Variance [6], variance of the number of interruptions of activity a_i can be represented as:

$$Var[N_i] = E[Var[N_i|\tilde{z}_i]] + Var[E[N_i|\tilde{z}_i]]. \tag{38}$$

According to our analysis for the variance representation of the deterministic case in Eqn 35, we have,

$$\begin{aligned} E[Var[N_i|\tilde{z}_i]] &= E[(d^0 + \tilde{z}_i)\sum_{k=1}^{K}\sum_{j=1}^{r_{ik}} \lambda_{jk}] \\ &= d^0 \sum_{k=1}^{K}\sum_{j=1}^{r_{ik}} \lambda_{jk}, \end{aligned} \tag{39}$$

and

$$\begin{aligned} Var[E[N_i|\tilde{z}_i]] &= Var[(d^0 + \tilde{z}_i)\sum_{k=1}^{K}\sum_{j=1}^{r_{ik}} \lambda_{jk}] \\ &= \sigma_i^2 (\sum_{k=1}^{K}\sum_{j=1}^{r_{ik}} \lambda_{jk})^2. \end{aligned} \tag{40}$$

Thus, the variance value of the number of interruptions can be represented by:

$$Var[N_i] = d^0 \sum_{k=1}^{K}\sum_{j=1}^{r_{ik}} \lambda_{jk} + \sigma_i^2 (\sum_{k=1}^{K}\sum_{j=1}^{r_{ik}} \lambda_{jk})^2. \tag{41}$$

B.2 Total Vacancy Time.

Lemma B3. *[26] Let* X *and* Y *be independent random variables that are both exponentially distributed, respectively with parameter λ and μ. The probability that* X *will be smaller than* Y *is then:*

$$P(X < Y) = \lambda/(\lambda + \mu).$$

Let P_{jk} denote the probability that the interruption for activity a_i is caused by the resource unit j of resource type k. That is, among all resource units used by activity a_i, this resource unit has the smallest time to breakdown. Then P_{jk} can be calculated as:

$$P_{jk} = P(X_{jk} < \min_{l=1,...K; i=1,...r_{il}; (i,l)\neq(j.k)} X_{il}). \tag{42}$$

From Eqn 42 and Lemma B3, we can rewrite P_{jk} as:

$$\begin{aligned} P_{jk} &= \lambda_{jk}/(\lambda_{jk} + \sum_{l=1,...K; i=1,...r_{il}; (i,l)\neq(j.k)} \lambda_{il}) \\ &= \lambda_{jk}/\sum_{l=1}^{K}\sum_{i=1}^{r_{il}} \lambda_{il}. \end{aligned} \tag{43}$$

Since we assume that only one unit of resource is allowed to break down at a time, we can calculate the mean and variance values of R_i describing the vacancy time for a_i to wait once interrupted as:

$$\begin{aligned} &E[R_i] = \sum_{k=1}^{K}\sum_{j=1}^{r_{ik}} P_{jk} E[Y_{jk}]. \\ &Var[R_i] = E[R_i^2] - E^2[R_i] \\ &= \sum_{k=1}^{K}\sum_{j=1}^{r_{ik}} P_{jk} E[Y_{jk}^2] - E^2[R_i] \\ &= \sum_{k=1}^{K}\sum_{j=1}^{r_{ik}} P_{jk}(Var[Y_{jk}] + E^2[Y_{jk}]) - E^2[R_i]. \end{aligned} \tag{44}$$

Here, random variable Y_{jk} represents the time needed to repair the resource unit j of resource type k once broken. Since we assume that $Y_{jk} \sim EXP(\mu_{jk})$, then we can obtain:

$$E[Y_{jk}] = 1/\mu_{jk}, \quad Var[Y_{jk}] = 1/(\mu_{jk})^2. \tag{45}$$

Therefore, from Eqns 19, 20, 37, 41, 44 and 45, mean and variance values of the stochastic part of processing time can be derived as:

$$\begin{aligned} &E[\tilde{\delta_i}] = d_i^0 \sum_{k=1}^{K}\sum_{j=1}^{r_{ik}} \frac{\lambda_{jk}}{\mu_{jk}}, \\ &Var[\tilde{\delta_i}] = d_i^0 \sum_{k=1}^{K}\sum_{j=1}^{r_{ik}} \frac{2\lambda_{jk}}{(\mu_{jk})^2} + \sigma_i^2 (\sum_{k=1}^{K}\sum_{j=1}^{r_{ik}} \frac{\lambda_{jk}}{\mu_{jk}})^2. \end{aligned} \tag{46}$$

References

1. Aytug, H., Lawley, M.A., McKay, K., Mohan, S., Uzsoy, R.: Executing production schedules in the face of uncertainties: A review and some future directions. In: European Journal of Operational Research, vol. 165(1), pp. 86–110 (2005)
2. Bartusch, M., Mohring, R.H., Radermacher, F.J.: Scheduling project networks with resource constraints and time windows. Annals of Operations Research **16**, 201–240 (1988)
3. Beck, J.C., Wilson, N.: Proactive algorithms for job shop scheduling with probabilistic durations. Journal of Artificial Intelligence Research **28**(1), 183–232 (2007)

4. Ben-Tal, A., Nemirovski, A.: Robust optimization - methodology and applications. Mathematical Programming **92**, 453–480 (2002)
5. Bidot, J., Vidal, T., Laborie, P., Beck, J.C.: A theoretic and practical framework for scheduling in a stochastic environment. Journal of Scheduling **12**, 315–344 (2009)
6. Billingsley, P.: Probability and Measure, 3 edn. Wiley-Interscience (1995)
7. Brucker, P., Drexl, A., Möhring, R., Neumann, K., Pesch, E.: Resource-constrained project scheduling: Notation, classification, models, and methods. European Journal of Operational Research **112**(1), 3–41 (1999)
8. Chen, X., Sim, M., Sun, P., Zhang, J.: A linear decision-based approximation approach to stochastic programming. Operations Research **56**(2), 344–357 (2008)
9. Daniels, R., Carrillo, J.: Beta-robust scheduling for single-machine systems with uncertain processing times. IIE Transactions pp. 977–985 (1997)
10. Dean, B.C., Goemans, M.X., Vondrák, J.: Approximating the stochastic knapsack problem: The benefit of adaptivity. In: Proceedings of the 45th Annual IEEE Symposium on Foundations of Computer Science, pp. 208–217 (2004)
11. Deblaere, F., Demeulemeester, E., Herroelen, W.: Proactive policies for the stochastic resource-constrained project scheduling problem. European Journal of Operational Research **214**(2), 308–316 (2011)
12. Deblaere, F., Demeulemeester, E., Herroelen, W., Van de Vonder, S.: Robust resource allocation decisions in resource-constrained projects. Decision Sciences **38**(1), 5–37 (2007)
13. Demeulemeester, E.L., Herroelen, W.S.: Project scheduling : a research handbook. Kluwer Academic Publishers, Boston (2002)
14. Eden, C., Williams, T., Ackermann, F., Howick, S.: The role of feedback dynamics in disruption and delay on the nature of disruption and delay (d&d) in major projects. Journal of the Operational Research Society **51**(3), 291–300 (2000)
15. Fu, N., Lau, H.C., Varakantham, P., Xiao, F.: Robust local search for solving rcpsp/max with durational uncertainty. Journal of Artificial Intelligence Research **43**, 43–86 (2012)
16. Fu, N., Lau, H.C., Xiao, F.: Generating robust schedules subject to resource and duration uncertainties. In: Proceedings of International Conference on Automated Planning and Scheduling, pp. 83–90 (2008)
17. Fu, N., Varakantham, P., Lau, H.C.: Towards finding robust execution strategies for rcpsp/max with durational uncertainty. In: Proceedings of International Conference on Automated Planning and Scheduling, pp. 73–80 (2010)
18. Hagstrom, J.N.: Computational complexity of PERT problems. Networks **18**, 139–147 (1988)
19. Herroelen, W., Leus, R.: Project scheduling under uncertainty: Survey and research potentials. In: European Journal of Operational Research, vol. 165(2), pp. 289–306 (2005)
20. Hillier, F.S., Lieberman, G.J.: Introduction to Operations Research. McGraw-Hill Science/Engineering/Math (2004)
21. Janak, S., Lin, X., Floudas, C.: A new robust optimization approach for scheduling under uncertainty :ii. uncertainty with known probability distribution. Computers and Chemical Engineering **31**, 171–195 (2007)
22. Kolisch, R., Hartmann, S.: Experimental investigation of heuristics for resource-constrained project scheduling: An update. European Journal of Operational Research (2005)
23. Kolisch, R., Schwindt, C., Sprecher, A.: Benchmark Instances for Project Scheduling Problems, pp. 197–212. Kluwer Academic Publishers, Boston (1998)
24. Lambrechts, O., Demeulemeester, E., Herroelen, W.: Proactive and reactive strategies for resource-constrained project scheduling with uncertain resource availabilities. Journal of Scheduling **11**, 121–136 (2008)
25. Lambrechts, O., Demeulemeester, E., Herroelen, W.: A tabu search procedure for developing robust predictive project schedules. International Journal of Production Economics **111**(2), 493–508 (2008)
26. Lambrechts, O., Demeulemeester, E., Herroelen, W.: Time slack-based techniques for robust project scheduling subject to resource uncertainty. Annals OR **186**(1), 443–464 (2011)
27. Lambrechts, O., Demeulemeester, E.L., Herroelen, W.S.: Exact and suboptimal reactive strategies for resource-constrained project scheduling with uncertain resource availabilities. In: Proceedings of the 3rd Multidisciplinary International Conference on Scheduling : Theory and Applications, pp. 575–577 (2007)
28. Lombardi, M., Milano, M.: 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, pp. 569–583. Springer-Verlag (2009)
29. Möhring, R.H.: Scheduling under uncertainty: Bounding the makespan distribution. In: Computational Discrete Mathematics, pp. 79–97 (2001)
30. Möhring, R.H., Stork, F.: Linear preselective policies for stochastic project scheduling. Mathematical Methods of Operations Research **52**(3), 501–515 (2000)
31. Policella, N., Cesta, A., Oddi, A., Smith, S.: Solve-and-robustify. Journal of Scheduling **12**, 299–314 (2009)
32. Policella, N., Smith, S.F., Cesta, A., Oddi, A.: Generating robust schedules through temporal flexibility. In: Proceedings of International Conference on Automated Planning and Scheduling, pp. 209–218 (2004)
33. Rasconi, R., Cesta, A., Policella, N.: Validating scheduling approaches against executional uncertainty. Journal of Intelligent Manufacturing **21**(1), 49–64 (2010)
34. Rice, J.A.: Mathematical Statistics and Data Analysis. Duxbury Press (2001)
35. Rodríguez, I.G., Vela, C.R., Puente, J., Hernández-Arauzo, A.: Improved local search for job shop scheduling with uncertain durations. In: Proceedings of International Conference on Automated Planning and Scheduling (2009)
36. Vonder, S., Demeulemeester, E., Herroelen, W.: A classification of predictive-reactive project scheduling procedures. Journal of Scheduling **10**(3), 195–207 (2007)
37. Van de Vonder, S., Demeulemeester, E., Herroelen, W.: Proactive heuristic procedures for robust project scheduling: An experimental analysis. European Journal of Operational Research **189**(3), 723–733 (2008)
38. Zhu, G., Bard, J., Yu, G.: Disruption management for resource-constrained project scheduling. Journal of the Operational Research Society **56**, 365–381 (2005)