Robust Local Search and Its Application to Generating Robust Schedules

Hoong Chuin LAU
School of Information Systems
Singapore Management University

Thomas OU
The Logistics Institute Asia Pacific
National University of Singapore

Fei XIAO
School of Information Systems
Singapore Management University

Abstract

In this paper, we propose an extended local search framework to solve combinatorial optimization problems with data uncertainty. Our approach represents a major departure from scenario-based or stochastic programming approaches often used to tackle uncertainty. Given a value $0 < \epsilon \leq 1$, we are interested to know what the robust objective value is, i.e. the optimal value if we allow an ϵ chance of not meeting it, assuming that certain data values are defined on bounded random variables. We show how a standard local search or metaheuristic routine can be extended to efficiently construct a decision rule with such guarantee, albeit heuristically. We demonstrate its practical applicability on the Resource Constrained Project Scheduling Problem with minimal and maximal time lags (RCPSP/max) taking into consideration activity duration uncertainty. Experiments show that, partial order schedules can be constructed that are robust in our sense without the need for a large planned horizon (due date), which improves upon the work proposed by Policella *et al.* 2004.

Introduction

Many optimization problems in planning and scheduling problems are NP-hard, and local search methods have been effectively employed to solve them. While these methods work well for deterministic versions of the problems, they often do not and cannot take data uncertainty into consideration, and hence it is not clear how these methods will perform when we apply them to solve the same optimization problems under uncertainty.

In the deterministic setting, a fitness or objective function is used to evaluate the 'goodness' of a solution. Without loss of generality, suppose we are concerned with a minimization problem and X is the feasible solution set. The goal is to seek a best solution, x^* with the minimal fitness value, i.e. $x^* = \arg\min_{x \in X} f(x)$.

In a corresponding uncertainty setting, we assume there exist data pertubations $\tilde{z}$, and the fitness function $f(x, \tilde{z})$ assumes different values under different scenarios, i.e. different realizations of the uncertain data parameters $\tilde{z}$. The goal will be to construct a policy/strategy that decides how x is to be instantiated as uncertainty parameters are realized dynamically. In the context of planning and scheduling, this refers to the plan/schedule execution strategy in a dynamic environment as uncertainty unfolds. Note that works that focus on scenarios-based generation may not give a tractable approach as the number of scenarios may grow exponentially with the size of the problem although there are research being done, such as Tarim *et al.* 2006 that reduces the number of scenarios required.

In this paper, we are interested in the following robust optimization problem: given $0 < \epsilon \leq 1$, find the minimum (i.e. optimal) value V^* and a policy such that we have 1-ϵ probability guarantee that, over all possible realizations of uncertain parameters $\tilde{z}$, the objective value of the resulting feasible solution x instantiated by the policy does not exceed V^*. Note that for some realizations of $\tilde{z}$, a feasible solution may not even exist, and in this paper, we ignore such scenarios. This value V^* is term as the *robust objective value*.

In other words, we offer the planner the option to choose a level of risk (ϵ), and our model will produce solutions whose values will not be worse than the robust value (V^*) by more than ϵ probability, under all possible realizations of uncertainty. Clearly, the higher the value of ϵ (i.e. the more risk adverse), the more conservative the robust value will be found. It is also worth noting that one may be tempted set V^* to a sufficient large value that guarantees the probability bound. In this paper, we are interested to find the smallest such value (i.e. least upper bound). In the scheduling context for instance, this translates to the problem of finding the minimum (1-ϵ)-guaranteed makespan under different realizations of uncertainty.

While optimization under uncertainty has been a well-studied topic, both in the AI and Operations Research communities, recent advances in robust optimization have only begun to propose tractable approximation models to solve linear stochastic optimization problems with uncertainties (e.g. Sim *et al.* 2007). The objective of this paper is to integrate concepts from robust optimization into classical local search to solve combinatorial optimization problems under uncertainty tractably. We will present an application of robust local search to generate robust partial order schedules (POS) for the Resource-Constrained Project Scheduling Problem with minimal and maximal time lags (RCPSP/max) under uncertainty.

The structure of the paper is as follows. A literature review is presented in section 2. We will present notations and our proposed robust local search framework in sections 3 and 4. Section 5 describes the application of our framework to RCPSP/max. Computational experiments on the RCPSP/max benchmark problems are provided in Section 6. Section 7 provides overall conclusions and offers some suggestions for future research.

Literature Review

In a practical scheduling environment, external unexpected events may happen and disrupt the scheduled completion time of the project. In recent years, there have been a growing number of proposed approaches that attempt to tackle this problem. For a complete survey of recent works on project scheduling, one may refer to Herroelen & Leus 2005.

One may broadly classify the decision-making approaches to tackle scheduling with uncertainty into 2 categories:

- Proactive scheduling that designs an a priori schedule that takes into account the possible uncertainty that may occur.
- Reactive scheduling that revises or re-optimizes the baseline schedule when an unexpected event occurs.

In this paper, we are concerned with proactive scheduling against uncertainty. One can divide the works in this area into 2 types: a) Executing a base-line schedules that is buffered against uncertainty, and b) Executing a temporally flexible scheduling policy.

Aytug *et al.* 2005 have highlighted the importance of baseline schedules in production scheduling. Herroelen & Leus 2004 raised the importance of having a baseline schedule that can absorb disruptions in a multi-project environment where it may be necessary to make advance bookings of key staff or equipment to guarantee their availability. The paper is concerned with the development of a stable baseline schedule in a multi-project environment that can absorb disruptions in activity durations without affecting the planning of other activities, such that coordination of resources and material procurement for each activity can be performed efficiently. They assume that during project execution, activities cannot be started before their foreseen starting time in the baseline schedule. The objective is to minimize the expected weighted deviation in activity start times.

A main argument against baseline schedules is that they are brittle in the face of unpredictable execution dynamics and can quickly become invalidated, a partial order schedule (POS) on the other hand retains temporal flexibility whenever problem constraints allow it and can often absorb unexpected deviation from predictive assumptions.

A POS is defined by Policella *et al.* 2007 as a set of activities which are partially ordered such that any possible complete order that is consistent with the initial partial order, is a resource and time feasible schedule. A POS consists of a set of feasible schedules that can be represented by a temporal graph. Mathematically, a POS can be represented by a temporal graph in which any activity is associated by a node and the edges representing the constraints between the activities. Within a POS, each activity retains a set of feasible start times, and these options provide a basis for responding to unexpected disruptions.

POS can be viewed as an intermediate approach between the use of a baseline schedule and the completely dynamic approach. Such approaches are based on the consideration that robustness can be increased by introducing flexibility in the schedule generation phase. An attractive property of a POS is that reactive response to many external changes can be accomplished via simple propagation in an underlying temporal network (a polynomial time calculation), only when an external change exhausts all options for an activity it is necessary to recompute a new schedule from scratch.

On a separate front, recent advances in robust optimization (Ben-Tal & Nemirovski 2002, Sim *et al.* 2006) has shown promising results in immunizing uncertainty in optimization against infeasibility while preserving the tractability of the model. Sim *et al.* 2007 proposes a new decision rule model to approximate recourse decisions that is computationally tractable using a mathematical programming approach. This approach, while tractable, may still be computationally challenging in solving large-scale optimization problems as compared to local search. An interesting new area of research that motivates this paper is the integration of robust optimization concepts and techniques into local search thereby producing a computationally efficient methodology for coping with large-scale combinatorial optimization under uncertainty, particularly in the scheduling context.

Preliminaries

Ben-Tal & Nemirovski 2002 classifies the variables in a stochastic optimization problem into 2 types: *Adjustable* and *Non-Adjustable variables*.

Definition 1. *Non-Adjustable variables are a priori decisions that must be made before the actual realization of the uncertainty.*

Definition 2. *Adjustable variables (also known as recourse variables) are 'wait-and-see' variables that can adjust themselves when part of the uncertain data become known.*

For example, in a scheduling problem such as RCPSP with uncertain task durations, the non-adjustable variables will represent the execution policy (in the case of Policella 2005, the POS) that need to be constructed apriori, while the adjustable variables are associated with the actual start times of the tasks, which will be set with respect to the execution policy and dynamic realizations of uncertainty.

Notationally, let a random variable be denoted by $\tilde{x}$ and bold face lower case letters such as $\mathbf{x}$ represent vectors. A primitive random variable $\tilde{z}_k$ is one which has zero mean. Examples of a primitive random variable include $U(-a, a)$ (uniform distribution between constants $-a$ and a) and $N(0, \sigma)$ (normal distribution with mean 0 and variance σ^2).

We assume that every uncertain parameter $\tilde{r}$ is equal to the sum of its nominal value (mean) r^0 and its deviation, represented by one (or possibly more) primitive random variable $\tilde{z}$. For example, in RCPSP, each uncertain task duration $\tilde{r}_k$ is represented by its mean value r_k^0 and its deviation represented by a single primitive random variable $\tilde{z}_k$.

A decision rule specifies how an adjustable variable is to be set with respect to the uncertainty parameters and non-adjustable variables. Let $\tilde{\mathbf{z}}$ and $\mathbf{x}$ denote the set of primitive random variables and non-adjustable variables respectively. In the linear decision rule framework proposed by Ben-Tal & Nemirovski 2002, each adjustable decision variable is assumed to be affinely dependent on a subset of some N number of primitive random variables:

$$V(\mathbf{x}, \tilde{\mathbf{z}}) = v^0 + \sum_{k=1}^{N} v_k(\mathbf{x})\tilde{z}_k \tag{1}$$

where each $v_k(x)$ $(1 \leq k \leq N)$ is a coefficient derived from $\mathbf{x}$. In RCPSP for example, $\mathbf{x}$ represents the POS to be generated; each task is associated with an adjustable variable $V(\mathbf{x}, \tilde{\mathbf{z}})$, where v^0 represents the earliest start time of this task under the POS, and v_k encodes how task k is related to this task in the POS. This will be further elaborated in the section on "Robust RCPSP/max".

For simplicity, we will henceforth rewrite $v_k(\mathbf{x})$ as v_k.

Segregated Random Variables

In the recent work of Sim *et al.* 2007, each primitive random variable $\tilde{z}_k$ can be represented by 2 *segregated* random variables $\tilde{z}_k^+$ (read z-plus) and $\tilde{z}_k^-$ (z-minus):

$$\tilde{z} = \tilde{z}^+ - \tilde{z}^- \tag{2}$$
$$\tilde{z}^+ = max\,\{\tilde{z}, 0\} \tag{3}$$
$$\tilde{z}^- = max\,\{-\tilde{z}, 0\} \tag{4}$$

In the following table, we give examples of the respective values of mean μ_p, μ_m and variance ${\sigma_p}^2$, ${\sigma_m}^2$ for the segregated variables $\tilde{z}^+$ and $\tilde{z}^-$.

$\tilde{z}$	$Var(\tilde{z})$	${\sigma_p}^2$, ${\sigma_m}^2$	μ_p,μ_m
$U(-a,a)$	$\frac{a^2}{3}$	$\frac{5a^2}{48}$	$\frac{a}{4}$
$N(0,\sigma)$	σ^2	$\frac{(\pi-1)\sigma^2}{2\pi}$	$\frac{\sigma}{\sqrt{2\pi}}$

Table 1: Values of the mean and variance for the segregated variables under Uniform and Normal Distribution

Segregated Linear Decision Rule

Under the segregated linear decision rule framework of Sim *et al.* 2007, each adjustable decision variable is assumed to be affinely dependent on a set of some N segregated random variables $\left\{\tilde{z}_1^+, \tilde{z}_1^-, \ldots, \tilde{z}_N^+, \tilde{z}_N^-\right\}$. Hence, a segregated linear decision rule has the following general form:

$$V(\mathbf{x}, \tilde{\mathbf{z}}) = v^0 + \sum_{k=1}^{N} \left\{v_k^+ \tilde{z}_k^+ + v_k^- \tilde{z}_k^-\right\} \tag{5}$$

As we will show below, a segregated linear decision rule allows us to easily obtain an upper bound on a subset of random variables (see Eqn 9, which is not possible in the linear decision rule framework proposed in Ben-Tal & Nemirovski 2002.

Given the mean and variance for each segregated variable $E(\tilde{z}_k^+) = E(\tilde{z}_k^-) = \mu_k$, $Var(\tilde{z}_k^+) = \sigma_{pk}^2$ and $Var(\tilde{z}_k^-) = \sigma_{mk}^2$, we can express the expected value and variance of any adjustable variable as:

$$E[V(\mathbf{x}, \tilde{\mathbf{z}})] = v^0 + \sum_{k=1}^{N} \left\{v_k^+ \mu_k + v_k^- \mu_k\right\} \tag{6}$$

$$Var[V(\mathbf{x}, \tilde{\mathbf{z}})] = \sum_{k=1}^{N} \left\{ \left[v_k^+ \sigma_{pk}\right]^2 + \left[v_k^- \sigma_{mk}\right]^2 - 2 v_k^+ v_k^- \mu_k \right\} \tag{7}$$

Segregated Linear Decision Rule in Scheduling

Adjustable variables are dependent on one another. In a scheduling context, tasks are often connected in series or parallel, and the start time of a task is dependent on the start times of the preceding tasks. Thus, adjustable variables are functions of other adjustable variables through the addition operator (to model serial activities) and/or the maximum operator (to model parallel activities). More details will be discussed on the Robust RCPSP section below.

Given some M number of adjustable variables, we may express its sum as an adjustable variable in the form of a segregated linear decision rule as follows:

$$\begin{aligned} &\textstyle\sum_{i=1}^{M} V_i(\mathbf{x}, \tilde{\mathbf{z}}) \\ &= \textstyle\sum_{i=1}^{M} v_i^0 + \sum_{k=1}^{N} \left\{ \sum_{i=1}^{M} v_{i,k}^+ \tilde{z}_k^+ + \sum_{i=1}^{M} v_{i,k}^- \tilde{z}_k^- \right\} \end{aligned} \tag{8}$$

Similarly, given some set C of adjustable variables, we may also express the upper bound on the maximum of these variables as an adjustable variable in the form of a segregated linear decision rule:

$$\begin{aligned} &\max_{i \in C}\{V_i(\mathbf{x}, \tilde{\mathbf{z}})\} \\ &\leq \max_{i \in C}\{v_i^0\} + \textstyle\sum_k \left\{ \max_{i \in C}\{v_{i,k}^+\} \tilde{z}_k^+ \right\} \\ &+ \textstyle\sum_k \left\{ \max_{i \in C}\{v_{i,k}^-\} \tilde{z}_k^- \right\} \end{aligned} \tag{9}$$

Robust Local Search

In this section, we propose how a classical local search algorithm may be extended in order to explicitly handle data uncertainty so as to solve the robust optimization problem presented in the Introduction.

Robust Optimization Problem Formulation

Given that the objective function for our problem is a function of non-adjustable variables $\mathbf{x}$ and uncertainty random variables $\tilde{\mathbf{z}}$, we can view the objective function as an adjustable fitness function $V(\mathbf{x}, \tilde{\mathbf{z}})$. Recall that the robust optimization problem is to find the minimum value V^* for which the following probability bound is observed:

$$P(V(\mathbf{x}, \tilde{\mathbf{z}}) \leq V^*) \geq 1 - \epsilon \tag{10}$$

From the one-sided Chebyshev's Inequality, we can obtain a bound for the robust objective value V^* as a function of its expected value and variance of the robust fitness function, i.e.:

$$\begin{aligned} &E[V(\mathbf{x}, \tilde{\mathbf{z}})] + \sqrt{\tfrac{1-\epsilon}{\epsilon}} \sqrt{Var[V(\mathbf{x}, \tilde{\mathbf{z}})]} \leq V^* \\ &\Rightarrow P(V(\mathbf{x}, \tilde{\mathbf{z}}) \leq V^*) \geq 1 - \epsilon \end{aligned} \tag{11}$$

Hence, we can reformulate our robust optimization problem as the following optimization model:

$$\begin{aligned} \min \quad & V^* \\ s.t. \quad & E[V(\mathbf{x}, \tilde{\mathbf{z}})] + \sqrt{\tfrac{1-\epsilon}{\epsilon}}\sqrt{Var[V(\mathbf{x}, \tilde{\mathbf{z}})]} \leq V^* \end{aligned} \tag{12}$$

From this model, we can now derive the robust fitness function which will be used in our proposed local search framework:

Definition 3. *Given* $0 < \epsilon \leq 1$ *and the adjustable fitness function* $V(\mathbf{x}, \tilde{\mathbf{z}})$ *defined above, the robust fitness function,* $f(x, \tilde{z}, \epsilon)$*, is defined as*

$$f(\mathbf{x}, \tilde{\mathbf{z}}, \epsilon) = E[V(\mathbf{x}, \tilde{\mathbf{z}})] + \sqrt{\frac{1-\epsilon}{\epsilon}}\sqrt{Var[V(\mathbf{x}, \tilde{\mathbf{z}})]} \tag{13}$$

Hence, the goal of a local search is to find a local minima of f given $\tilde{\mathbf{z}}$ and ϵ. Note that the function f can be computed efficiently via Eqns $5, 6$, which is a linear function of nominal values with the mean and variance of the segregated variables. This is important since local search typically requires the fitness function to be computed many times.

Robust Local Search Algorithm

This section will present how our concepts of segregated linear decision rule and robust fitness function can be integrated with a standard local search to solve the Robust Optimization Problem defined above. Our proposed algorithm is outlined as follows. Steps i, ii, v and vi are standard steps in a local search algorithm. Steps iii and iv represent our departure from standard local search to deal with uncertainty.

i. **Generate Initial Solution**
This is usually obtained using a simple greedy heuristic.

ii. **Generate Neighbourhood of Solutions**
Generate a pool of neighbour solutions from the current solution.

iii. **Assume Segregated Linear Decision Rule for all adjustable variables and check feasibility**
For each candidate solution $\mathbf{x}$ in the solution pool, derive the coefficients $v_k(\mathbf{x})$ for each adjustable variable. Subsequently, for each solution check constraint violation and reject those that are not feasible.

iv. **Evaluate Robust Fitness Function** f
For each feasible solution $\mathbf{x}$, evaluate f to obtain the robust objective values. The solution with the lowest robust objective value is the current best robust solution.

v. **Apply Penalty (optional)**
Some advanced local search strategies may require a penalty to be applied to prevent it from being caught at a local minima. In the case of tabu-search for example, a tabu-list is updated when a tabu move is applied. In the case of iterated local search, a perturbation move will be applied to the current local minima.

vi. **Termination Criteria**
If the termination criteria is met, return the solution with the lowest robust fitness function value else repeat the optimization cycle by determining the next move.

Robust RCPSP/max

Research on resource constrained project scheduling problem has been mostly concerned with the generation of a precedence and resource feasible baseline schedule that minimizes the deterministic makespan of the project. However, in practice, projects are often unable to observe to its given baseline schedule with many failing to meet the expected project deadlines. This leading to the incurrence of extra cost as a result of late delivery penalties.

Therefore there exists a practical need to study the tackling of the scheduling problem with uncertainty being taken into consideration. Although there exist research that looks at the problem of stochastic availability of the resources, the scope of this paper will only cover the aspect of uncertainty in terms of the stochastic duration of activities in project scheduling. Möhring 2001 reports that the problem of finding the distribution a given project makespan with activity duration uncertainty itself is a $\sharp P$-Complete problem in general. Thus, it is even more computationally challenging to consider the project scheduling problem with resource constraint in the uncertainty context.

Our approach takes the proactive viewpoint in generating partial-order schedules that can be executed within a project completion time which we term the *robust make-span* with a probability of at least $1 - \epsilon$.

In the following 4 subsections, we will first define the RCPSP/max problem, followed by the various metrics of robustness for partial order schedules. We will then discuss the formulation of our robust model and finally present our proposed robust local search algorithm to search for the best robust POS.

RCPSP/max Problem Formulation

The standard RCPSP/max has the following inputs:

- V: Set of N activities $\{a_0, a_1, \ldots, a_n, a_{n+1}\}$ represented as nodes in a graph. Activity a_0 is inserted that denotes the dummy start node and a_{n+1} denotes the dummy end node.
- d_i: Activity processing time for activity i.
- C_k: Constant resource capacity for resource type k over entire horizon.
- E_p: Set of temporal constraints between activities represented as directed edges in a graph.
- r_{ik}: Resource requirement for activity i for resource type k.

A schedule is an assignment of start times to activities $a_0, a_1, \ldots a_n, a_{n+1}$, i.e. a vector $S = (s_0, s_1, \ldots, s_n, s_{n+1})$ where s_i denotes the start time of activity a_i. The time at which activity a_i has been completely processed is called its completion time and is denoted by V^i. Since we assume that processing times are deterministic and preemption is not permitted, completion times are determined by $V^i = s_i + d_i$. Schedules are subjected to both temporal and resource constraints. Resource constraints, in the form of $\sum_{i \in A_t} r_{ik} \leq C_k \forall t, k$ ensures that the resource usage for each resource type k does not exceed the capacity at any point of time during the execution of the project in which A_t is the set of active activities at time t. Temporal constraints contained in the set E_p, designate arbitrary minimum and maximum time lags between the start times of any two activities, $l_{ij}^{min} \leq s_j - s_i \leq l_{ij}^{max}$. The objective is to minimize the make-span, V^n of the whole project.

In the robust optimization counterpart, we assume that the processing time of each activity to be uncertain. The problem we need to solve is given a value ϵ, find a POS for a RCPSP/max problem such that the robust make-span V^* of the POS is minimized, and the probability that the POS will be completed within V^* is at least $1 - \epsilon$.

Measuring the Robustness of Schedules

Robustness of Partial Order Schedules In this subsection, we shall look at the various metrics in measuring the robustness of a POS. The first metric is taken from Aloulou & Portmann. 2003 defines the metric flexibility, $flex$ as follows:

$$flex = \sum_{i=1}^{n} \sum_{j>i \wedge a_i \nprec a_j \wedge a_j \nprec a_i}^{n} \frac{1}{n\,(n-1)} \tag{14}$$

Flexibility measures the number of pairs of activities in the solution which are not reciprocally related by simple precedence constraints. The rationale is that when two activities are not related it is possible to move one without moving the other one. Hence, the higher the value of $flex$ the lower the degree of interaction among the activities.

The second metric by Smith *et al.* 1998 defines the average width, relative to the temporal horizon, of the temporal slack associated with each pair of activities (a_i, a_j) as shown in the following equation:

$$fldt = \sum_{i=1}^{n} \sum_{j=1 \wedge j \neq i}^{n} \frac{slack(a_i, a_j)}{H \times n \times (n-1)} \times 100 \tag{15}$$

where H is the horizon of the problem defined above, n is the number of activities, $slack(a_i, a_j)$ is the width of the allowed distance interval between the end time of activity a_i and the start time of activity a_j , and 100 is a scaling factor. This metric characterizes the fluidity of a solution, i.e., the ability to use flexibility to absorb temporal variation in the execution of activities. Furthermore it considers that a temporal variation concerning an activity is absorbed by the temporal flexibility of the solution instead of generating deleterious domino effects (the higher the value of $fldt$, the lower the risk, i.e., the higher the probability of localized changes).

Note that all the above metrics do not explicitly take into consideration the distribution of the uncertain durations. More precisely, all metrics are concerned with measuring the number of solutions contained within a POS within a fixed horizon H, which is usually a large value. Unfortunately, these metrics do not give an accurate measure of how large the makespan can be if we take into consideration the uncertainty of the durations. Now if we know some distributional information such as variance, we will be able to generate POS that are robust in the sense that we can measure the precise probability of successful execution of the generated POS, or conversely, given a probability, we can find a robust makespan such that all executions will have a makespan bounded by that robust makespan with the probability guarantee.

Robust Model Formulation

In our proposed model, we do not assume the probability distribution type of the activity duration because in real practice it is difficult to justify such assumption. The processing time of each activity is represented by a random variable $\tilde{d}$. We denote $d^0 = E[\tilde{d}]$ as the mean processing time.

We can view the negative segregated random variable $\tilde{e} = \max\{d^0 - \tilde{d}, 0\}$ as representing the probability distribution of the earliness i.e. the distribution of the difference of its processing time from the mean time d^0 when the activity is completed before its mean time. Similarly, we can view the positive segregated random variable $\tilde{l} = \max\{\tilde{d} - d^0, 0\}$ represents the lateness i.e. the distribution of the difference of its processing time from the mean time d^0 when the activity is completed after its mean time. The uncertain processing time of an activity is thus composed of 3 components its *mean* d^0, *earliness* $\tilde{e}$, and *lateness* $\tilde{l}$.

$$\tilde{d} = d^0 + \tilde{l} - \tilde{e} \tag{16}$$

The value of $E\left[\tilde{l}_k \tilde{e}_k\right]$ equals to zero, since an activity can only either be early or late.

Segregated Linear Decision Rule

We use the segregated linear decision rule to represent the earliest starting time $S(\tilde{z})$ and finishing time $V(\tilde{z})$ for each activity.

Serial Activities Consider two activities a_1 and a_2 that are connected serially, in which a_2 starts after a_1 is completed. Assuming no lag times in between activities. The starting time and ending time for a_1 are:

$$S_1(\tilde{l}, \tilde{e}) = 0 \tag{17}$$

$$V_1(\tilde{l}, \tilde{e}) = S_1(\tilde{l}, \tilde{e}) + \tilde{d}_1 = d_1^0 + \tilde{l}_1 - \tilde{e}_1 \tag{18}$$

The starting and ending time for a_2 are:

$$S_2(\tilde{l}, \tilde{e}) = V_1(\tilde{l}, \tilde{e}) = d_1^0 + \tilde{l}_1 - \tilde{e}_1 \tag{19}$$

$$V_2(\tilde{l}, \tilde{e}) = S_2(\tilde{l}, \tilde{e}) + \tilde{d}_2 = (d_1^0 + d_2^0) + (\tilde{l}_1 + \tilde{l}_2) - (\tilde{e}_1 + \tilde{e}_2) \tag{20}$$

In general, we can express the end time of the n-th activity of a serial N-activity project network using Eqn 8 as follows:

$$\begin{aligned} V_n(\tilde{l}, \tilde{e}) &= V_{n-1}(\tilde{l}, \tilde{e}) + \tilde{d}_n \\ &= \textstyle\sum_{i=1}^{N} d_i^0 + \tilde{l}_i - \tilde{e}_i \end{aligned} \tag{21}$$

Parallel Activities Consider two parallel activities a_1 and a_2 that are executed concurrently. The completion time of the both activities can express an upper bound ending time of both activities from Eqn 9 as follows:

$$\begin{aligned} S_3(\tilde{l}, \tilde{e}) &= \max\left(V_1(\tilde{l}, \tilde{e}), V_2(\tilde{l}, \tilde{e})\right) \\ &\leq \max\left(d_1^0, d_2^0\right) + (\tilde{l}_1 + \tilde{l}_2) \end{aligned} \tag{22}$$

In general, we can express the upper-bound of the end time of any parallel N-activity project networks as follows:

$$S(a_{n+1}) \leq \max_{i \in N}\{d_i^0\} + \sum_{i=1}^{N} \tilde{l}_i \tag{23}$$

Thus, given any project network, we can formulate its earliest finishing time of the project as an adjustable variable in segregated linear decision rule using Eqns 21 and 23. We can define the adjustable project makespan variable as a robust fitness function to be used in a local search from Eqn 13, $V^*(x, \tilde{z}, \epsilon)$. We termed the value returned from the function $V^*(x, \tilde{z}, \epsilon)$ as the *robust makespan*, V^*

Algorithmic Design

Algorithm 1 shows the design of our robust local search. Given the a) RCPSP/max problem instance, b) mean and variance values of the segregated variables c) level of risk (ϵ), the algorithm will return the POS with the minimal robust makespan, V^*. In essence, we perform robust local search on the neighbourhood set of activity lists. An activity list is defined as a precedence-constraint feasible sequence commonly used by heuristics to generate earliest start time schedules using methods such as the serial schedule generation scheme in solving the standard RCPSP problem (see Kolisch & Hartmann 2005).

The following is a description of how Algorithm 1 works.

In our algorithm, different activity lists will be explored by local moves. In our context, we only consider the activity list as the sequence of activities which satisfy the non-negative minimal time lag constraint. When we consider maximal time lag constraint in RCPSP/max, scheduling each activities to its earliest possible start time based on the its order position in the activity list may restrict the schedule so much that it may not even return in a feasible schedule. Thus, when we schedule each activity sequentially based on its order position in the activity list, we will instead assign its starting time by randomly picking a time from its domain of feasible start times.

According to preliminary experiments, this new randomized approach returns more feasible solutions than the earliest start time one. After finding a feasible schedule, a POS will be generated by applying the chaining procedure proposed in Policella *et al.* 2004. Then, the V^* value will be computed according to the POS. Intuitively, using the randomized approach may return a schedule with a large baseline scheduled completion time. However, we can apply the shortest path algorithm on the resultant POS to generate the earliest start time schedule for a smaller makespan.

As mentioned above, it may be difficult to find a feasible schedule that satisfies minimal and maximal time lag constraints using activity list. In fact, we believe that in the set of all activity lists, many may not yield a feasible schedule. We overcome this problem in the following way. Define the set of activity lists which result in feasible (resp. infeasible) schedules as F (resp. I). We seek to design a local search algorithm with the following characteristics: a) Starting from an activity list in I, the local search should move to an activity list in F within a short time. b) Starting from

Algorithm 1 Robust Local Search Algorithm

```
Generate an activity list AL randomly
Find a feasible start time schedule, $S$ randomly according to AL
if $AL \in F$ then
    $POS \leftarrow chaining(S)$
    Computer $V^*_{now}$ according $POS$
    Update $V^*_{min}$ as $V^*_{now}$
else
    Record the first activity $a$ which can not be scheduled
end if
for $i \leftarrow 1$ to Max_Iteration do
    if $AL \in I$ then
        Shift activity $a$ ahead in AL randomly as AL'
    else
        Select two activity $b$ and $c$ in AL randomly
        Swap $b$ and $c$ in AL as AL'
    end if
    Find randomized start time schedule $S'$ according to AL'
    if $AL' \in F$ then
        $POS' \leftarrow chaining(S')$
        Compute $V^*$ according to $POS'$
        if $AL \in I$ or $V^* \leq V^*_{now}$ then
            $V^*_{now} \leftarrow V^*$
            $AL \leftarrow AL'$
            if $V^* \leq V^*_{min}$ then
                $V^*_{min} \leftarrow V^*$
            end if
        end if
    else if $AL \in I$ then
        $AL \leftarrow AL'$
    else
        $p \leftarrow rand(0, 1)$
        if $p < 0.01$ then
            $AL \leftarrow AL'$
            Record the first activity $a$ which can not be scheduled
        end if
    end if
end for
```

an activity list in F, the local search should move to the activity list with the minimal V^* value. c) We also diversify the exploration of activity lists in F by allowing the local search to move from an activity list in F to an activity list in I, since activity lists in the F region may not be reachable from one another by simple local moves. This has the flavor of strategic oscillation proposed in meta-heuristics research.

With all the above considerations in mind, we apply two different types of local moves. To converge quickly to an activity list in F, the first local move is designed to schedule the activity which meets temporal or resource conflict earlier. It will randomly shift ahead the first activity which cannot be scheduled in the current activity list. 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 sat-

				$V^*(x, \tilde{z}, \epsilon)$			$flex(x)$			$fldt(x)$		
	ϵ	%	cpu(s)	V^*	$flex$	$fldt$	V^*	$flex$	$fldt$	V^*	$flex$	fldt
J10	0.01	100	0.1	47.6	0.242	0.630	52.2	0.297	0.659	51.4	0.268	0.713
	0.05			46.8	0.244	0.632	51.1			50.4		
	0.1			46.5	0.247	0.634	50.8			50.1		
	0.2			46.3	0.244	0.632	50.6			49.9		
J20	0.01	96.2	0.7	79.9	0.237	0.624	87.7	0.282	0.655	87.6	0.249	0.751
	0.05			78.5	0.238	0.621	86.2			86.1		
	0.1			78.2	0.237	0.620	85.9			85.7		
	0.2			77.7	0.238	0.622	85.6			85.4		
J30	0.01	98.9	3.0	106.4	0.240	0.588	113.8	0.272	0.613	116.7	0.239	0.736
	0.05			104.5	0.237	0.586	112.0			114.9		
	0.1			104.4	0.238	0.586	111.6			114.5		
	0.2			103.4	0.236	0.590	111.3			114.1		

Table 2: Comparison of using robust makespan (V^*), flexibility ($flex$) and fluidity ($fldt$) as metrics of robustness on J10, J20, J30 benchmark problem sets

isfying the non-negative minimal time lag constraints. The move will be accepted, if it results in a smaller or equal V^* value. To explore more different activity lists, we introduce a small probability to accept the move which lead to an infeasible schedule. The detailed robust local search procedure is given in Algorithm 1.

In Algorithm 1, $chaining$ is the function to generate a POS from a baseline schedule (Policella *et al.* 2004). The probability to move from an activity list in F to one in I is set at 0.01. $Max\_Iteration$ is the maximal number of iterations of the robust local search. The minimal V^* value will be saved as V^*_{min}.

Experimental Evaluation

In this section we will present the experimental results of applying the robust local search algorithm to solve the robust optimization problem associated with RCPSP/max. Comparison is made between the POS obtained using the robust make-span V^* as its fitness function and those obtained using the flexibility and fluidity metrics as the fitness function. We run our robust local search algorithm on the standard RCPSP/max benchmark problem sets as defined in Kolisch *et al.* 1998. This benchmark consists of 3 sets J10, J20 and J30 of 270 problem instances of difference size 10 x 5, 20 x 5 and 30 x 5 (number of activities x number of resources).

For all the data set we assume that its nominal activity duration, d^0 is the deterministic activity duration given from benchmark. To model activity duration uncertainty, we let each activity's duration to be a random variable $\tilde{d} = d^0 + \tilde{z}$ where $\tilde{z} \sim N(0, 0.1)$. $\tilde{z}$ can be further split into its segregated variables: $\tilde{z} = \tilde{z}^+ - \tilde{z}^-$. By setting the standard deviation of $\tilde{z}$ to be 0.1, we like to investigate how the RCPSP/max robust model would react to small variation in the activity duration. The nominal duration of each activity in the benchmark ranges from 1 to 10 and averaging around 5. The values, $Var(\tilde{z}^+) = \sigma_p^2 = 0.0341$, $Var(\tilde{z}^-) = \sigma_m^2 = 0.0341$ and $E(\tilde{z}^+) = E(\tilde{z}^-) = \mu = 0.0399$ can be calculated using the formulas inside Table 1.

We run our robust local search algorithm across 4 increasing levels of risk, $\epsilon = \{0.01, 0.05, 0.1, 0.2\}$ with the $Max\_Iteration$ set to 1000 for the local search. For each benchmark set, the local search algorithm is repeated 3 times differently using different fitness function. First robust make-span V^* as the fitness function which is repeated 4 times each for different values of ϵ. Subsequently, the local search's fitness function is replaced with the flexibility and fluidity function. The values of the flexibility and fluidity used are normalized as in Policella *et al.* 2007. The search algorithm will attempt to return the best POS, POS_{flex} and POS_{fldt} with the maximal flexibility and fluidity respectively. We also compute the robust makespan for POS_{flex} and POS_{fldt}. All the results recorded in Table 2 are the average results over the subset of the solved problem instances. The % column reports the percentage of the number of problems that our algorithm solves over the total number of solved problem instances in each benchmark sets. The cpu column reports the average runtime for each benchmark cases. The algorithm is implemented in C++ and the CPU times reported are executed on a Pentium 4-3000 MHz processor under LINUX.

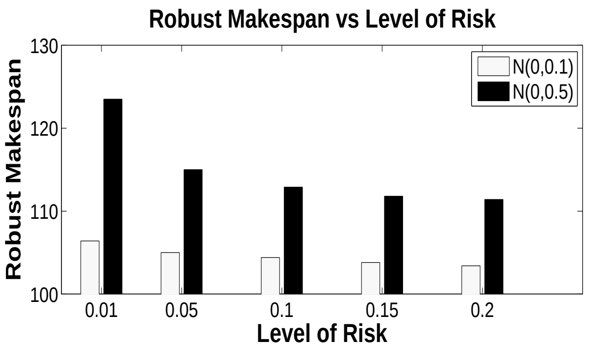

Figure 1: Robust makespan versus Level of Risk Curve (J30 benchmark)

From table 2, we first observe that, the robust make-span increases as the level of risk(ϵ) decreases. The lower the level of risk that the planner is willing to take, the higher the robust make-span value for the POSs generated. We repeat the experiment on the J30 set, but now increase the variability of each activity duration to $\tilde{z} \sim$N(0,0.5). The obtained results are shown in the chart in Fig 1. We can observe that as the level of risk increases, the robust-makespan decreases in a convex manner. In other words, we have to incur a much higher cost (a higher robust-makespan) as we become more conservative. Furthermore, this trend is more acute in case where the activity duration variability is high (compare N(0,0.5) with N(0,0.1)). Although this result is intuitive, our approach provides a method for how this tradeoff is quantified. With this tradeoff result, the planner may utilize the robust make-span versus risk(ϵ) curve to decide on the desired optimal robust POS.

Another observation we can obtain from the experiments is that by using flexibility and fluidity as the fitness function for local search, we obtain a POS with a higher robust make-span as compared to using robust make-span as the fitness function. This observation can be seen across different levels of risk (ϵ) as well in different problem set sizes. This suggests that having high fluidity does not imply robustness and thus we may not be required to buffer too much slack between activities, against conventional wisdom. Our result also does not seem to indicate any relationship between robust makespan and flexibility, as the values of flexibility fluctuates at different levels of ϵ for each problem set.

Conclusion

In this paper, we propose a simple yet efficient method to enrich local search so that it is capable of solving a variant of optimization problem under uncertainty: namely, given a level of confidence (or risk), how to find a robust objective value and a policy such that when uncertainty is dynamically realized, the policy execution will result in a solution whose value is as good as the robust value with the given level of confidence (risk). We applied our local search framework to tackle the RCPSP/max problem under uncertainty where a schedule policy (i.e. POS) need to be generated. Interestingly, our experimental results show that given mild activity uncertain distribution, we can find POS with better robust makespan that do not necessarily have high flexibility and fluidity. And as a by-product, we also managed to improve the results obtained in Policella *et al.* 2007.

The future direction of this work is promising - the most obvious avenue being the improvement of our framework, and its application to tackle other planning and scheduling problems. We also believe that our work can be a launch pad for designing systems that enable decision-makers to find solutions which are not only optimized, but also robust against uncertainty, in a computationally efficient manner.

Acknowledgments

The authors would like to thank Dr Melvyn Sim for his suggestions and guidance in integrating robust optimization with local search in a dynamic setting using segregated linear decision rule.

References

Aloulou, M.A. & Portmann., M.C. (2003). An efficient proactive reactive scheduling approach to hedge against shop floor disturbances. In *1st Multidisciplinary International Conference on Scheduling: Theory and Applications(MISTA)*, 337–362.

Aytug, H., Lawley, M.A., McKay, K., Mohan, S. & Uzsoy, R. (2005). Executing production schedules in the face of uncertainties: A review and some future directions. In *European Journal of Operational Research*, vol. 165(1), 86–110.

Ben-Tal, A. & Nemirovski, A. (2002). Robust optimization - methodology and applications. *Math. Prog. Series B*, **92**, 453–480.

Herroelen, W. & Leus, R. (2004). The construction of stable project baseline schedules. In *European Journal of Operational Research*, vol. 156(3), 550 – 565.

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

Kolisch, R. & Hartmann, S. (2005). Experimental investigation of heuristics for resource-constrained project scheduling: An update. *European Journal of Operational Research*.

Kolisch, R., Schwindt, C. & Sprecher, A. (1998). *Benchmark Instances for Project Scheduling Problems*, 197–212. Kluwer Academic Publishers, Boston.

Möhring, R.H. (2001). Scheduling under uncertainty: bounding the makespan distribution. *Computational Discrete Mathematics: advanced lectures*, 79–97.

Policella, N. (2005). *Scheduling with Uncertainty A Proactive Approach using Partial Order Schedules*. Ph.D. thesis, Universit‘a degli Studi di Roma La Sapienza.

Policella, N., Smith, S.F., Cesta, A. & Oddi, A. (2004). Generating robust schedules through temporal flexibility. In *International Conf. on Automated Planning and Scheduling (ICAPS)*, 209–218.

Policella, N., Cesta, A., Oddi, A. & Smith, S. (2007). From precedence constraint posting to partial order schedules: A csp approach to robust scheduling. In *AI Communications*, accepted.

Sim, M., Chen, X. & Sun, P. (2006). A robust optimization perspective of stochastic programming. *Operations Research*, accepted.

Sim, M., Chen, X. & Sun, P. (2007). A linear-decision based approximation approach to stochastic programming. *Operations Research*, accepted.

Smith, S.F., Cesta, A. & and, A.O. (1998). Profile-based algorithms to solve multiple capacitated metric scheduling problems. In *Artificial Intelligence Planning Systems*, 214–231.

Tarim, S.A., Manandhar, S. & Walsh, T. (2006). Stochastic constraint programming: A scenario-based approach. *Constraints*, **11(1)**, 53–80.