Using Simulation for Execution Monitoring and On-line Rescheduling with Uncertain Durations

Julien Bidot and **Philippe Laborie**
ILOG S. A.
9, rue de Verdun - B. P. 85
94253 Gentilly Cedex, France
{jbidot, plaborie}@ilog.fr

J. Christopher Beck
Cork Constraint Computation Centre
University College Cork, Ireland
c.beck@4c.ucc.ie

Thierry Vidal
L. G. P. /ENIT
47, av. d'Azereix - B. P. 1629
65016 Tarbes Cedex, France
thierry@enit.fr

Abstract

The problem we tackle is on-line rescheduling with temporal uncertainty, activity durations are uncertain and activity end times must be observed during execution. In this paper we will assume we have a representation of the uncertainty of each activity duration in the form of probability distributions which are used in the simulation of schedule execution. We use the simulations to monitor the execution of the schedule and in particular to estimate the quality of the schedule and the end times of the activities. Given an initial schedule, the schedule starts execution and we must decide when to reschedule. We propose and explore a non-monotonic technique where each time we reschedule we can completely change the existing schedule except for those activities that have already started (or finished) execution. This paper explicitly addresses the basis on which the decision to reschedule is made by investigating three simple measures of the data provided by simulation.

Introduction

Nowadays planning and scheduling must take into account incomplete information and/or potential changes in the environment, i.e. uncertainties (Smith, Frank, & Jónsson 2000). The central issue is to design robust planning and scheduling techniques, aimed at guaranteeing the feasibility and the quality of the executed schedule. Several ways to get a more robust schedule have already been investigated (Herroelen & Leus 2002; Davenport, Gefflot, & Beck 2001; Davenport & Beck 2000; Wu, Byeon, & Storer 1999; Le Pape 1995; Smith 1994). One way among others is to keep one and only one fixed schedule to execute, but reschedule when it appears the quality of the currently executing schedule degrades. This approach is relevant as far as rescheduling is fast enough w.r.t. the scheduling execution, i.e. when the dynamics of the system are low. Relevant questions are then: how and when the quality should be assessed? How and when we should reschedule? How do we assess the robustness of the approach? The paper intends to partially answer these questions.

The problem we tackle is on-line rescheduling with temporal uncertainty, activity durations are uncertain and activity end times must be observed during execution. In this paper we will assume we have a representation of the uncertainty of each activity duration in the form of probability distributions which are used in the simulation of schedule execution. We use the simulations to monitor the execution of the schedule and in particular to estimate the quality of the schedule (i.e., the makespan) and the end times of the activities. In the technique that we propose and explore, those data are used to decide when to reschedule; when such a decision is taken, the initial schedule is forsaken and a new one is computed, except for those activities that have already started (or finished) execution. The global process is hence clearly a non-monotonic on-line scheduling strategy which is used in a reactive way. This paper addresses the basis on which the decision to reschedule is made by investigating three simple measures of the data provided by simulation.

Definitions

In this paper we focus on the problem of execution monitoring and the decision to reschedule for a Job-Shop Scheduling Problem (JSSP) with uncertain durations. In this section, we define the JSSP, present our assumptions about the evolution of the uncertainty of activities during execution, and give a number of definitions used throughout the paper.

A JSSP is composed of jobs. A job is a set of activities that have to be performed in a given order i.e. there is a precedence constraint between each pair of activities in this set. Each activity is assigned to a single resource. Each resource can process one activity at a time, resources are modeled as unary resources (resource capacity of unit capacity). A solution to a JSSP is the assignment of a start time to each activity in order to satisfy the temporal and resource constraints and to optimize one or several criteria. Our objective, which is often used, is to minimize the makespan of the JSSP i.e. to minimize the time between the start time of the first activity executed and the end time of the last activity executed. A JSSP is a generic problem formulation suitable for many application problems, not only resources in a shop, namely when one has specific sequences of activities requiring resources. Hence a JSSP may be a sub-problem of a more general planning problem.

Each uncertain activity duration is modeled by a variable that follows a probability distribution and is associated with a domain. Any probability law can be used (normal, uniform, etc.) however as the duration of an activity must be strictly positive, the laws are truncated. At execution, we know the effective duration of an activity only when the activity ends and we learn this piece of information from the environment. The only decision is when to start the execution of each activity. As a consequence, we know at each moment, during the execution which are the activities that have been executed, the ones that are currently executed, and the ones that have not been started yet. When an activity is still executing and its minimum possible duration has been exceeded, our information on the uncertainty can be updated since the set of possible durations is now reduced. Therefore the probability distribution is further truncated and renormalized. We update data at specific points in time, i.e. the moments when we want to decide if we reschedule or not, and hence we need to know the precise current state.

Suppose we have an activity whose execution started at date $t = 0$ and whose duration, which is between 0 and 50, follows a normal law of mean 20 and standard deviation 10. This activity duration is depicted by the dotted curve on Figure 1. Now suppose that at date $t = 25$ the activity is still executing, we will assume that the probability distribution of the end of the activity is the initial law truncated on the interval [25,50] and renormalized as shown by the solid curve on Figure 1. More formally, if the initial probability law of an activity is described by a distribution function $p_0(d)$, and if the activity has been executed since $d0$ units of time, the current distribution is defined by the probability distribution $p_{d0}(d)$ as follows:

$$\forall d \in [0, d0], p_{d0}(d) = 0 \tag{1}$$

$$\forall d > d0, p_{d0}(d) = \frac{p_0(d)}{\int_{d0}^{+\infty} p_0(t)dt} \tag{2}$$

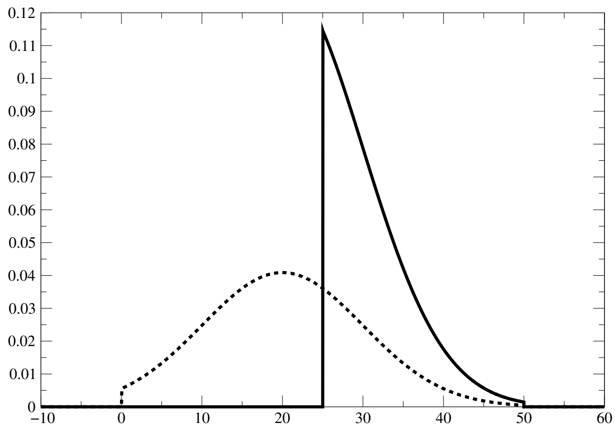

Figure 1: Two truncated normal laws

The following three definitions are used in this paper:

- The **effective** duration of an activity is the duration that is observed during execution. The effective schedule is the sequence of activities obtained after execution: all the activity durations have been observed. An **execution scenario** is the set of all effective durations.
- An **indicative** schedule is a solution that is generated by using fixed activity durations. In an indicative schedule the activities on each resource are totally ordered. The first indicative schedule is calculated off-line by using the mean activity durations. During execution a new indicative schedule is constructed if we decide to reschedule; in that case this new indicative schedule is calculated by using both the effective durations of the activities already executed and the mean durations of the other activities.

Non-monotonic Approach

Execution Model Description

The initial (off-line) schedule starts execution and we must decide when to reschedule. The main idea behind the criteria investigated here is that we use simulation to determine when to reschedule. We start execution using the indicative schedule and a simple execution policy: activities are started as soon as possible following the precedence constraints in the original problem, the activity sequences defined by the indicative schedule, and the effective activity durations that are observed. During execution of the schedule, we choose to evaluate the need for rescheduling whenever an activity ends. The first step consists then in updating the truncated probability distributions representing our view of the uncertain variables, as discussed in the last section. The second step is to calculate a chosen rescheduling criterion that depends on estimated values of some variables in the problem. If the estimated value varies too greatly from the indicative value, we reschedule.

Our estimations of the values for the variables we are interested in are generated through the use of simulation. According to the currently executing schedule and the current probability distributions, we randomly generate 1,000 execution scenarios whenever an activity ends. Each scenario allows us to calculate the simulated values for the variables we are monitoring and over the 1,000 simulations we calculate their means and standard deviations. The estimated value of the variable is its mean value over the simulations. If we decide to reschedule, the current schedule is ignored and a new schedule is generated except for the activities that have already started or finished executing.

Three Variations

The first criterion consists in monitoring the makespan and is defined as follows: we reschedule when the estimated makespan M_{est} is greater than the rescheduling threshold, $M_{est} > \frac{M_{ind}}{\sigma}$ where M_{ind} is the indicative makespan and σ is the sensitivity factor. This criterion will not result in rescheduling if activity durations are shorter than expected. In other words, it does not allow "opportunistic" rescheduling that would take advantage of unexpectedly short execution times.

A second variation, based on the first, is opportunistic because it reschedules based on the absolute difference between the estimated and indicative makespans. Rescheduling is done when the absolute difference between the estimated makespan M_{est} and the indicative makespan M_{ind} is greater than the rescheduling threshold. In this case, our rescheduling threshold is the mean of all activity durations of the deterministic problem, D, divided by σ. More formally, we reschedule when: $|M_{est} - M_{ind}| > \frac{D}{\sigma}$.

The two versions based on makespan monitoring are *a priori* rather crude: we mainly take into account the observed durations of the activities of the critical path. If these activities do not slip too much then, as a consequence, the makespan will not slip too much either. It is possible however, that the estimated makespan remains approximately stable while the executions of the activities that do not belong to the critical path are such that it is possible to find a much better solution by rescheduling. We thus propose a third variation of the approach that takes into account each activity duration. First we consider at each time the set A of the n activities that had not finished execution the last time we rescheduled. $\overline{|D_{diff}|}$ is then defined as the mean of the absolute differences between the estimated and indicative end times denoted end_{est} and end_{ind} for such activities: $\overline{|D_{diff}|} = \frac{\sum_A |end_{est} - end_{ind}|}{n}$. We reschedule each time $\overline{|D_{diff}|}$ is greater than our previous rescheduling threshold, i.e. when $\overline{|D_{diff}|} > \frac{D}{\sigma}$.

Implementation Issues

We need variables and data structures to maintain the probability distributions of all activities and other measurements like the makespan because we get information about the length of each duration during execution.

A random variable is defined by a definition interval $d = [x_{min}, x_{max}]$ and a probability distribution function f. The definition interval is uniformly discretized into n steps and for each step $x_i = x_{min} + \frac{x_{max} - x_{min}}{n} i$ with $i \in [0, n]$. Once this initialization has been performed, a random value $x \in d$ can be computed in $O(1)$.

A histogram is a data structure that allows us to store the probability distribution of a given random variable that depends on several random variables. For example, the makespan of the schedule is a random variable that depends on the durations of the activities on the critical paths. Histograms provide estimations of the mean and standard deviation of the random variable. We simply incrementally maintain $s_1 = \sum_{i=1}^{n} x_i$ and $s_2 = \sum_{i=1}^{n} x_i^2$ where the x_i are the n values recorded on the histogram. Each histogram requires $O(1)$ in memory; the time to update the aggregated quantities s_1 and s_2 with a new value is in $O(1)$. Means and standard deviations are also accessed in $O(1)$ and respectively calculated with $\frac{s_1}{n}$ and $\sqrt{\frac{n \times s_2 - s_1^2}{n^2 - n}}$.

In addition a precedence graph is generated from the constraint network: each node represents an activity and each arc represents a precedence constraint between two activities. We then topologically sort the precedence graph and use this ordering in the simulations. The complexity of a single simulation is in $O(n + m)$ where n is the number of activities and m is the number of precedence constraints; it is in $O(n)$ for a JSSP.

Experimental Study

The instances of the JSSP with imprecise durations are generated from classical JSSP instances. Each activity is associated with a normal probability distribution with mean, p, corresponding to the duration in the classical instance, and with standard deviation $\alpha \times p$ with $\alpha > 0$. α characterizes the degree of uncertainty of the problem. The higher α the higher the standard deviation of each activity so the more uncertainty in the system. α is constant and equals 0.3 for each experiment done in this study.

During schedule execution, whenever we are informed by the environment that an activity has ended, we update all our data structures, and simulate the continued execution of the schedule. The updating frequency is sufficiently low to permit us to run 1,000 simulations each time. When the end of an activity is observed, there might be other concurrent activities still being executed for which distributions are updated. Scheduling and rescheduling are done using the constraint programming approach with a standard chronological backtracking algorithm and a time limit of one second. The new schedule is the best solution found within the time limit. On the tested problem instances, the optimal schedule was often found. All scheduling is done with ILOG Scheduler 5.3 (ILOG 2002). Unless otherwise noted, the results come from running 100 different execution scenarios per value of the sensitivity factor. In this experimental study we only experiment with one problem, future work will explore a set of problems.

Results

We use a measure of the computational effort and schedule quality to evaluate our rescheduling criteria. The former is measured by the number of times that we reschedule, while for the latter we use the estimated makespan. The problem we tackle here is called la11 (Lawrence 1984). It is a JSSP with 20 jobs and 5 resources. The optimal makespan with the mean durations is 1,222.

Makespan Monitoring

In this subsection we display the results for the first variation when the makespan is monitored. For a single example execution scenario, Figure 2 and 3 show the mean and the standard deviation of the estimated makespan over time. We monitor the makespan each time an activity or several activities finish. We observe the mean of the estimated makespan can change quite a lot at a given time if an activity of the critical path ends at this time. For this execution scenario and a sensitivity factor $\sigma = 1$, 18 reschedulings occur, the mean of the first estimated makespan is 1,362.07 and the final mean makespan is 1,253.32.

Figure 4 and 5 represent the mean and the standard deviation of the number of reschedulings as the sensitivity factor σ changes. We notice the maximum number of reschedulings is 101, i.e. the number of activities plus one. The minimum number of reschedulings equals 0 when the sensitivity factor is too small (less than 0.91 in this case). We also notice a quick drop for $0.98 \leq \sigma \leq 1.2$. It is not surprising that the number of reschedulings quickly approaches 101 when $\sigma > 1$ with this rescheduling criterion. With regard to the standard deviation we notice it is maximum and equal to 26.6 for $\sigma \approx 1.05$ and is approximately

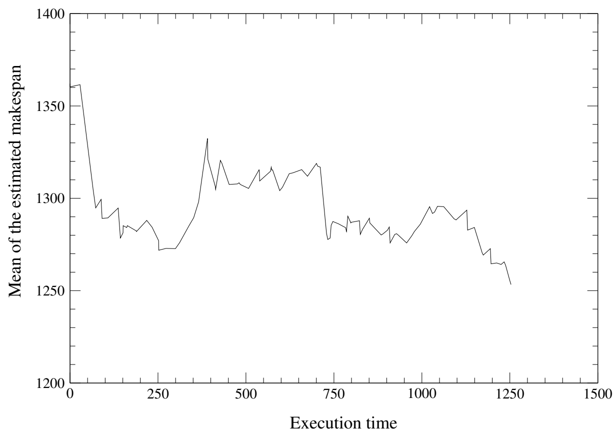

Figure 2: Mean of the estimated makespan for a single example scenario with makespan monitoring

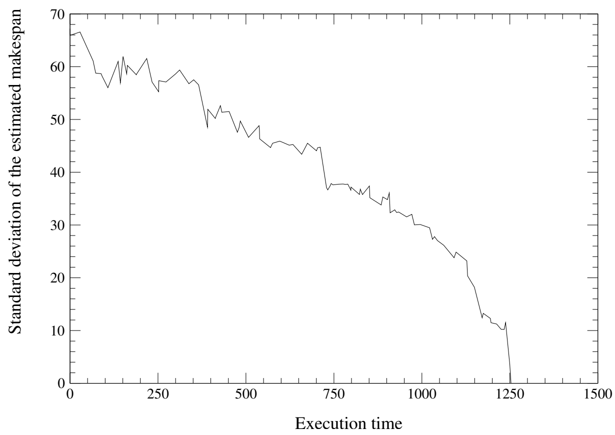

Figure 3: Standard deviation of the estimated makespan for a single example scenario with makespan monitoring

0 for $\sigma < 0.87$ and $\sigma > 1.05$. It is difficult to accurately predict the actual number of reschedulings for $\sigma \approx 1$.

Figure 6 and 7 represent the mean and standard deviation of the estimated makespan versus the sensitivity factor σ. We notice a quick drop of the mean between about 0.95 and 1.01. We observe that the mean effective (final) makespan is always less than the first mean estimated makespan $M_{est} = 1,349.87$ and the higher sensitivity factor, the lower the mean final makespan with more reschedulings.

Absolute Makespan Monitoring

Turning to our second criterion, we expect to find better results if we also reschedule when activities end earlier than estimated.

For the same execution scenario as the one used for Figure 4 and a sensitivity factor $\sigma = 10.7$, the criterion based on absolute makespan results in 34 reschedulings. The mean of the first estimated makespan is 1,362.07 and the final mean makespan is 1,249.72. This final value is, in fact, the optimal makespan for the durations of the execution scenario. The shapes of the curves representing the mean and the standard deviation of the estimated makespan are very similar to those on Figure 2 and 3.

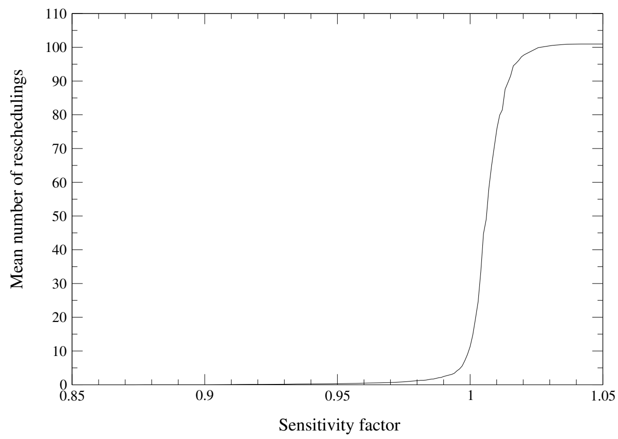

Figure 4: Mean number of reschedulings with makespan monitoring

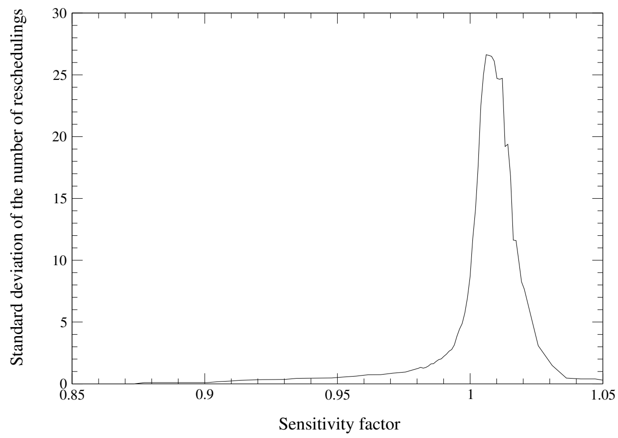

Figure 5: Standard deviation of the number of reschedulings with makespan monitoring

Figure 8 and 9 represent the mean and the standard deviation of the number of reschedulings versus the sensitivity factor σ. The minimum number of reschedulings equals 0 when the sensitivity factor is too small (near to 0). With regard to the standard deviation we notice it is maximum and is approximately 5.1 for $\sigma = 32.11$ and approximately stabilizes at 2.8 for $\sigma > 118$.

Figure 10 and 11 represent the estimated makespan versus the sensitivity factor σ. The mean and standard deviation of the estimated makespan are plotted. We notice a quick drop of the mean for $0.27 \leq \sigma \leq 10.7$; it approximately stabilizes at 1,257. The standard deviation is very unstable for a low σ, $0.27 \leq \sigma \leq 26.76$ and then gradually stabilizes until it reaches about 69.5. We still observe that the effective (final) makespan is always less than the first estimated makespan

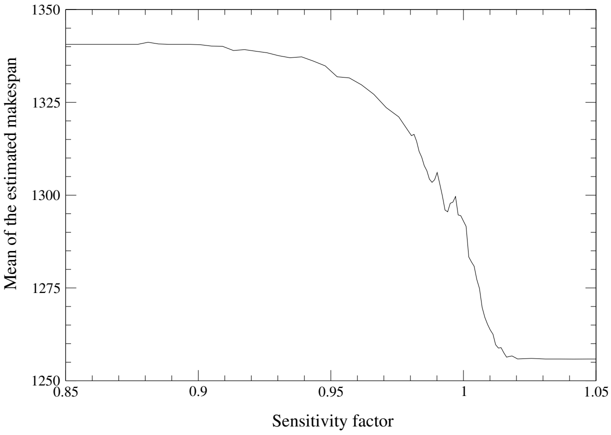

Figure 6: Mean of the estimated makespan with makespan monitoring

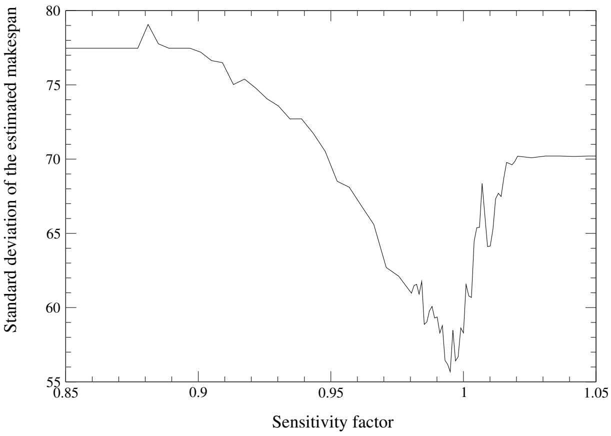

Figure 7: Standard deviation of the estimated makespan with makespan monitoring

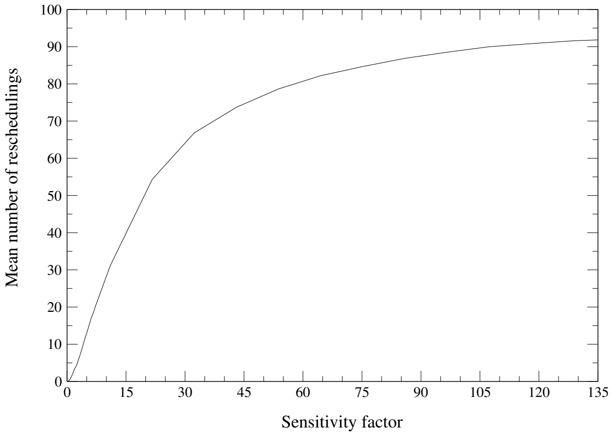

Figure 8: Mean number of reschedulings with absolute makespan monitoring

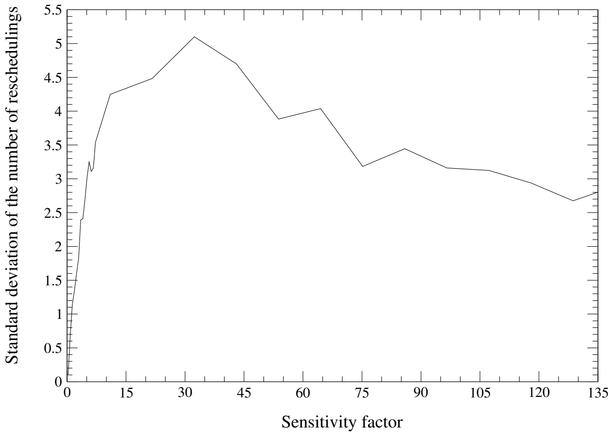

Figure 9: Standard deviation of the number of reschedulings with absolute makespan monitoring

$M_{est} = 1,349.87$ and the higher sensitivity factor, the lower the final makespan with more reschedulings. These results support our intuition that rescheduling to take advantage of positive events (i.e., activities ending sooner than estimated) can lead to stronger performance.

Monitoring of Activity End Times

The first two criteria are myopic because they only focus on the makespan. Makespan is an aggregate view of the "state" of the schedule since it depends only on the activities on the critical path. In addition, when the standard deviation of the estimated makespan is large, the mean of the estimated makespan does not provide accurate information. We thus investigate a third rescheduling criterion that takes into account all the activity durations individually.

For the same execution scenario as the one used for Figure 4 and a sensitivity factor $\sigma = 9$, our third criterion results in 29 reschedulings. The mean of the first estimated makespan and the final mean makespan are the same as those seen with the absolute makespan criterion. The shapes of the curves representing the mean and the standard deviation of the estimated makespan are very similar to those on Figure 2 and 3.

Figure 12 and 13 represent the mean and the standard deviation of the number of reschedulings versus the sensitivity factor σ. The minimum number of reschedulings equals 0 when the sensitivity factor is too small ($0 < \sigma < 0.5$ in this case). With regard to the standard deviation we notice it is maximum and is approximately 3.6 for $8 \leq \sigma \leq 31$ and approximately stabilizes at 1.7 for $\sigma > 60$.

Figure 14 and 15 represent the mean and standard deviation of the estimated makespan versus the sensitivity factor σ. We notice a quick drop of the mean for $0.5 \leq \sigma \leq 5$; it approximately stabilizes at 1,258. The standard deviation is very unstable for a low σ ($0.5 \leq \sigma \leq 5$) and then gradually stabilizes until it approximately reaches 68.5.

These results show that this rescheduling criterion performs better than the absolute makespan criterion because

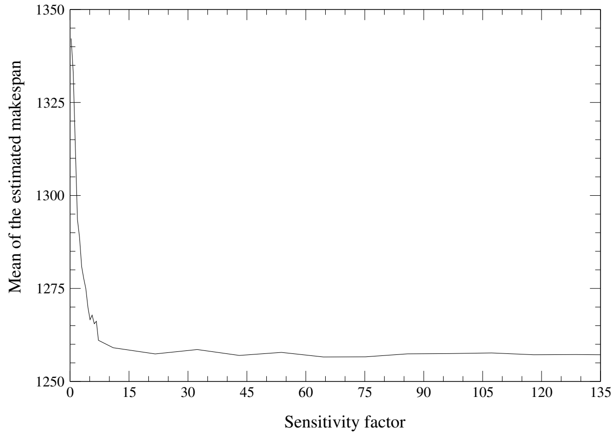

Figure 10: Mean of the estimated makespan with absolute makespan monitoring

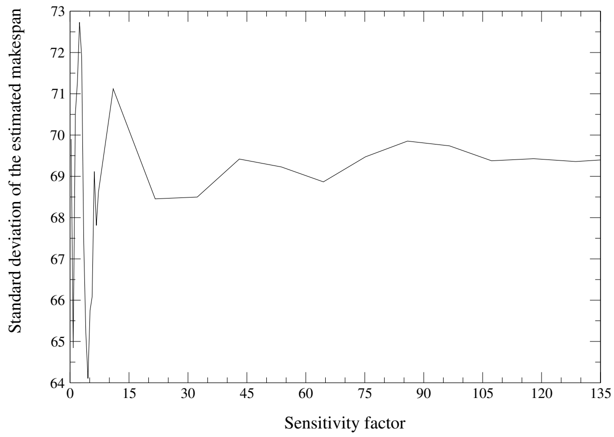

Figure 11: Standard deviation of the estimated makespan with absolute makespan monitoring

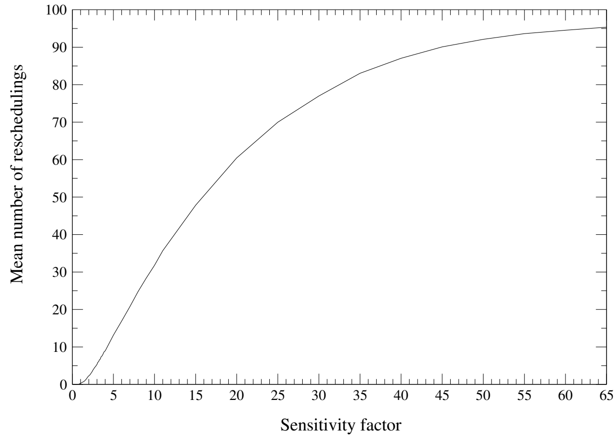

Figure 12: Mean number of reschedulings with monitoring of activity end times

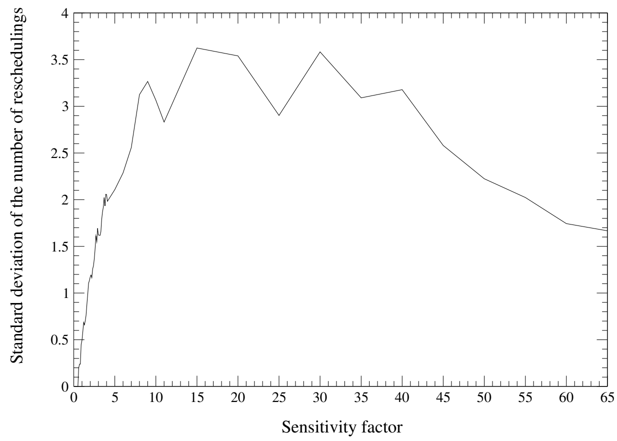

Figure 13: Standard deviation of the number of reschedulings with monitoring of activity end times

we are now opportunistic w.r.t. each activity execution (not only w.r.t. the activities belonging to the critical path). We thus observe that the effective (final) makespan is always less than the first estimated makespan $M_{est} = 1,349.87$ and the higher sensitivity factor, the lower the final makespan with more reschedulings.

Global Comparison

Figure 16 shows the mean estimated makespan obtained for each of the three variations versus the mean number of reschedulings. Whatever the rescheduling criterion, monitoring permits to improve schedule quality. These three curves confirm that the criterion based on the activity end time monitoring is almost always the best in terms of quality: for a given computational effort, this rescheduling technique provides the smallest makespan. We get approximately the same quality with the three rescheduling criteria when the computational effort is either very low or very high (mean number of reschedulings less than 3 or greater than 82). For a mean computational effort (mean number of reschedulings between 20 and 70) the rescheduling criteria taking into account positive events improve quality.

Future Work: Monotonic Approach

We would also like to investigate another approach which only performs monotonic decisions, that is, unlike the first approach, when a given decision has been taken to order activities, it cannot be changed later on; remember that in the approach investigated in this paper, the indicative schedule could be recomputed from scratch. The problem with such a monotonic approach is that we must be very careful when taking a decision. On the one hand we have to wait until uncertainty around the decision is low enough so that the decision is well informed. On the other hand we cannot wait too long because the schedule is executing and we do not just want to have a reactive and myopic decision process. In this approach, we have decided to schedule the overall problem piece by piece during the execution. Determining

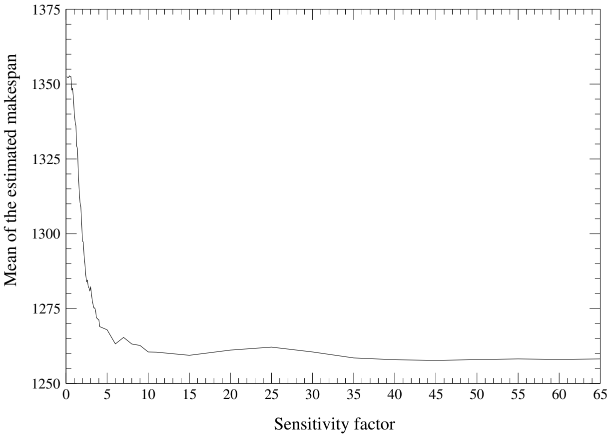

Figure 14: Mean of the estimated makespan with monitoring of activity end times

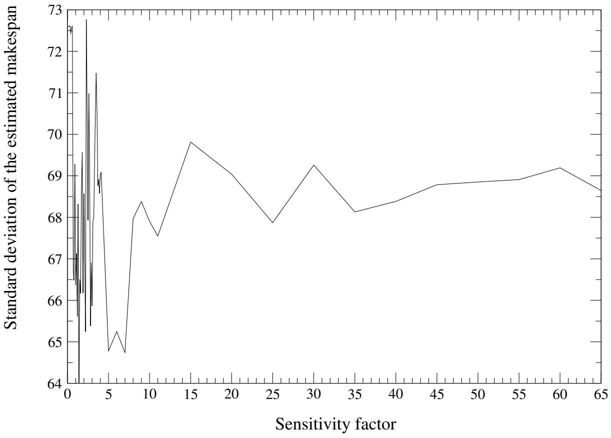

Figure 15: Standard deviation of the estimated makespan with monitoring of activity end times

when to schedule a new set of activities and how to select the set to be scheduled will be done using information from simulation. This approach is particularly suited for applications where there is a great distance between the executing agent and the global planning and scheduling system, like in space applications (Chien *et al.* 1999), in multi-agent cases when the schedule is used as a reference by other agents and should hence remain unchanged, at least in the short time range, or in applications where the dynamics are high.

More precisely, suppose that we are at a given time t and we are executing a schedule. We assume that part of the problem has been scheduled up to a given time horizon t_D. On the one hand, of course, we do not want to wait until t_D before scheduling subsequent activities because we would then have very little time to react and could not easily come up with a good decision. On the other hand we do not want to make decisions too far in the future in regions where there is still a lot of uncertainty. Furthermore, we do not want to take the ordering decisions one by one, which would be

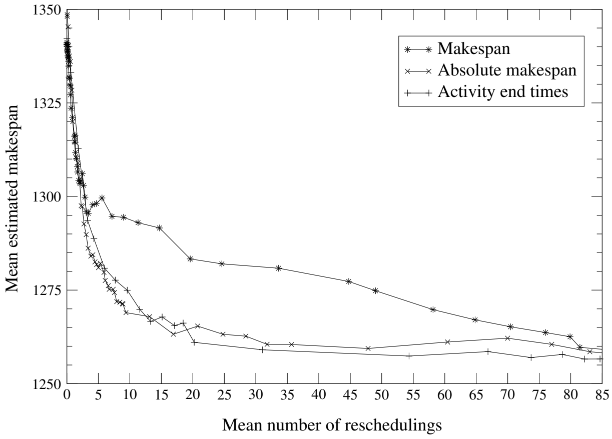

Figure 16: Mean estimated makespan for the three criteria

very myopic and certainly lead to a poor schedule quality but rather, select a subset of activities and perform a combinatorial search on this sub-problem in order to minimize negative impact on schedule quality. So there are two questions here: (1) how to design a condition that will be monitored during the execution and say "now, we can extend the current schedule by optimizing a new part of the problem" and (2) when this condition is triggered, how to select the set of activities to be scheduled. For both questions, we will use information coming from the evolving probability distributions and simulation.

To answer the first question we propose to evaluate the trade-off between the fact that $t_D - t$ should be large enough to have time to perform a combinatorial search leading to a good solution and the fact that execution has advanced far enough to get reduced uncertainty on the estimated end times of the scheduled part. This last piece of information is given by a simulation technique that computes the probability distributions of these end times.

When the condition above is satisfied, we still need to select an additional set of activities to schedule. We do not want to select a too big problem because: (i) we do not have an infinite time to schedule it (in any case less than $t_D - t$) and (ii) we do not want to take decisions too far in the future as they concern data with too much uncertainty. For each job we collect still unordered activities by chronological order in accordance with precedence constraints in such a way that the estimated standard deviation of the end time of the last activity of the collection is less than a given threshold. The sub-problem is then defined as all the collected activities on all the jobs. Once a nearly-optimal solution has been found to this sub-problem, this solution is added to the scheduled set and the process continues.

It should be noted that one can view this monotonic technique as a compromise between purely reactive or dispatching techniques (i.e. when we only schedule the next activity that has not yet executed on each resource) and proactive ones (i.e. we take into account uncertainty off-line and never reconsider the schedule at execution). The larger the horizon, the more proactive; the smaller the horizon, the more reactive.

Conclusion

This paper presents a non-monotonic approach to deciding when to reschedule based on the use of simulation. We simulate on-line, at specific points in time, the execution of the remainder of the indicative schedule. Three rescheduling criteria were explored. According to the results, rescheduling permits to improve schedule quality. The computational effort necessary to reach a given schedule quality depends on the rescheduling criterion. This work in progress seems to be promising since there are a lot of new directions we can follow to better understand and tune our techniques in order to adapt to different applications.

References

Chien, S.; Knight, R.; Sherwood, R.; and Rapideau, G. 1999. Using iterative repair to increase the responsiveness of planning and scheduling for autonomous spacecraft. In *Proceedings of the Workshop "Scheduling and Planning meet Real-time Monitoring in a Dynamic and Uncertain World"*.

Davenport, A. J., and Beck, J. C. 2000. A survey of techniques for scheduling with uncertainty. Unpublished manuscript available at http://www.eil.utoronto.ca/profiles/chris/chris.papers.html.

Davenport, A. J.; Gefflot, C.; and Beck, J. C. 2001. Slack-based techniques for building robust schedules. In *Proceedings of the Sixth European Conference on Planning*.

Herroelen, W., and Leus, R. 2002. Project scheduling under uncertainty - survey and research potentials. In *Eighth International Workshop on On-line Scheduling and Planning*.

ILOG. 2002. ILOG Scheduler 5.3: Reference Manual and User's Manual. Technical report, ILOG S. A.

Lawrence, S. 1984. *Resource-constrained project scheduling: an experimental investigation of heuristic scheduling techniques (Supplement)*. Ph.D. Dissertation, Graduate School of Industrial Administration, Carnegie Mellon University, Pittsburgh, Pennsylvania, United States of America.

Le Pape, C. 1995. Experiments with a distributed architecture for predictive scheduling and execution monitoring. In *Artificial Intelligence in Reactive Scheduling*. Chapman & Hall, R. Kerr and E. Szelke edition. 129–145.

Smith, D. E.; Frank, J.; and Jónsson, A. K. 2000. Bridging the gap between planning and scheduling. *Knowledge Engineering Review*.

Smith, S. F. 1994. OPIS: A methodology and architecture for reactive scheduling. In *Intelligent Scheduling*. San Francisco: Morgan Kaufmann, M. Zweben and M. S. Fox edition. 29–66.

Wu, S. D.; Byeon, E.; and Storer, R. H. 1999. A graph-theoretic decomposition of the job-shop scheduling problem to achieve scheduling robustness. *Operations Research* 47:113–123.