A Progressive Technique for Solving a Scheduling Problem with Uncertain Activity Durations and Alternative Resources

November 28, 2003

The scheduling technique presented here permits us to tackle a scheduling problem in which activity durations are uncertain: they are not precisely known before they finish execution, activities are associated with a set of possible resources, and resources can break down during execution: we do not precisely know when and how long resources break down before they effectively break down and are effectively repaired.

This technique permits us to construct a schedule solution piece by piece as long as execution goes along. The idea is to select, allocate and order a series of subsets of activities during execution by taking into account temporal, resource, and cost constraints. Allocation and ordering decisions made are never changed later on: this is a monotonic approach. Scheduling decisions are made during execution because we observe activity end times and apply a scheduling decision policy to set activity start times, we execute activities as soon as possible.

The problem with such a monotonic approach is that we must be very careful when taking an allocation decision or an ordering 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 a partial flexible schedule is executing and we do not just want to have a reactive and myopic decision process. Determining when to allocate and order a new subset of activities and how to select the subset to be al will be done using information from simulation.

More precisely, suppose that we are at a given time t and we are executing a partial flexible schedule. We assume that a subset of activities $A_{\text{allocorder}}$ of the problem have already been allocated and ordered given all constraints at t and the rest of the activities A_{pending} of the problem are not yet allocated and only ordered given the precedence constraints of the original problem. A subset of activities $A_{\text{executed}} \subseteq A_{\text{allocorder}}$ have already been executed and another one $A_{\text{executing}} \subseteq A_{\text{allocorder}}$ are executing. On the one hand, of course, we do not want to wait until the last activity of $A_{\text{allocorder}}$ finishes execution before allocating and ordering subsequent activities of A_{pending} 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 allocation and ordering decisions one by one, which would be 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 satisfy temporal, resource, and cost constraints. So there are two questions here: (1) how to design conditions that will be monitored during execution and say "now, we can extend the current partial flexible schedule by allocating and ordering a new part of the problem" and (2) when these conditions are triggered, how to select the subset of activities to be allocated and ordered. For both questions, we will use information coming from the evolving probability distributions and simulation.

1 When to Extend the Current Partial Flexible Schedule?

To extend the current partial flexible schedule we need to assess how long we still have to select, allocate and order a new subset of activities of A_{pending} and how big the uncertainty level of the allocated ordered activity end times is, i.e. we need to monitor two conditions during schedule execution.

1.1 Temporal Condition

Given $A_{\text{allocorder}}$ there exists a time point t_{D} that is the last time point before which we have to make an allocation decision and an ordering decision to anticipate execution. t_{D} is equal to the earliest end time of the activity of $A_{\text{allocorder}}$ that finishes at the earliest time in the subset of the allocated activities $A_{\text{allocorderlast}} \subseteq A_{\text{allocorder}}$ that are ordered at last positions in process plans. To determine t_{D} we assume that the activities of $A_{\text{allocorder}}$ are executed as soon as possible and the effective duration of these activities are the shortest possible ones. Figure 1 represents the execution of a partial flexible schedule of a 6×6 scheduling problem; $A_{\text{allocorder}}$ is the subset of the 19 shadowed activities; t_{D} is the earliest end time of the activity of p_1 ordered at the third position; $A_{\text{allocorderlast}}$ is composed of the six most shadowed activities that are allocated and at the last positions of process plans. A_{pending} is composed of the 17 white activities. We extend the current partial flexible schedule when $t_{\text{D}} - t < \delta t^{\text{min}}_{\text{progress}}$.

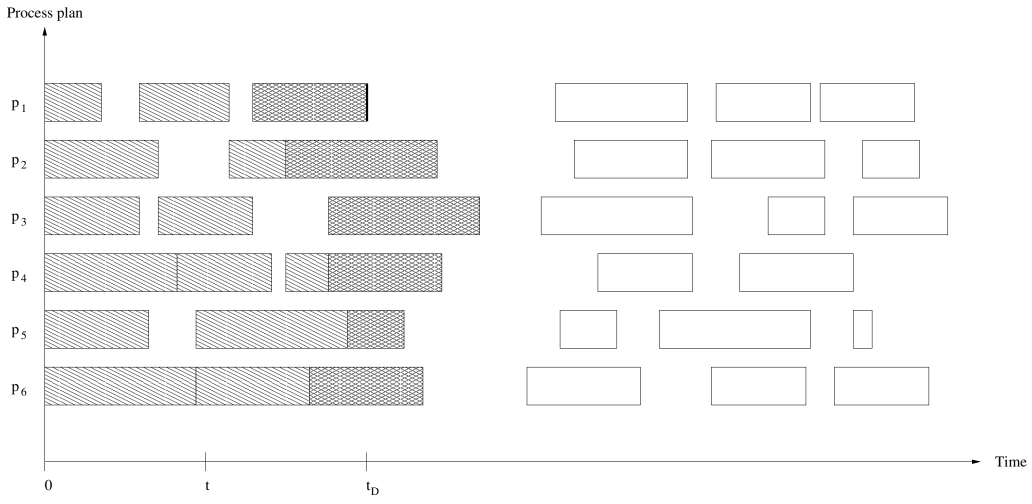

Figure 1: Example of the execution of a partial flexible schedule

1.2 Uncertainty Degree Condition

We propose to evaluate the trade-off between the fact that $t_{\text{D}} - t$ 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 expected end times of $A_{\text{allocorderlast}}$. This last piece of information is given by a simulation technique that computes the probability distributions of these end times. For computing the expected end times of the activities of $A_{\text{allocorderlast}}$ we run a given number of simulations (realizations) of the scheduled activity durations and the resource break downs. A precedence graph is generated from the constraint network of the partial flexible schedule: 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. For each activity we randomly pick a set of durations and for each resource we randomly pick a set of breakdowns: dates and durations of breakdowns; with these two sets of realizations we then update the precedence graph by assuming these scheduled activities are executed as soon as possible and the duration of an activity a_l allocated to a resource r_j is extended if r_j breaks down during the execution of a_l. Figure 2 shows an example of the simulation of an activity a_3 allocated to one of two alternative resources r_1 and r_3. a_3 executes from the date 5 to the date 20 if the resource allocated to it does not break down during its execution: it is the case if it is allocated to r_1 because r_1 breaks down at the date 25, after the end of its execution. However, if a_3 is allocated to r_3, then its end time is 45 because it is suspended after having being executed during 5 time units at the date 10 and resumes execution during 10 time units at the date 35.

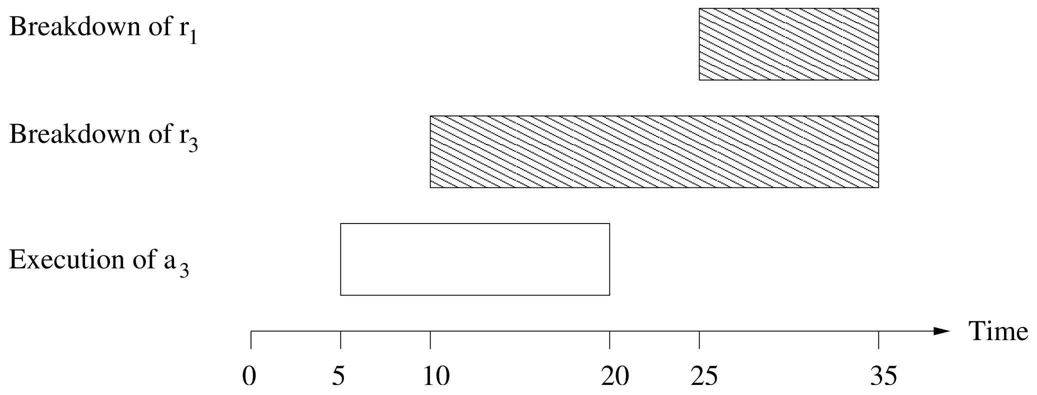

Figure 2: Example of simulation of an activity allocated to one of two alternative resources

When the biggest standard deviation of the end times of the activities of $A_{\text{allocorderlast}}$ is less than a given standard deviation $\sigma^{\min}_{\text{progress}}$ we can select a new subset of activities. In a more formal way, we extend the current partial flexible schedule when $max_{\forall a_l \in A_{\text{allocorderlast}}}(\sigma(end^{\text{exp}}_l)) < \sigma^{\min}_{\text{progress}}$, where end^{exp}_l is the expected end time of a_l.

2 How to Select the Subset of Activities to be Allocated and Ordered?

As soon as one of the two conditions defined in Subsection 1.1 and Subsection 1.2 is satisfied, we still need to select an additional subset of activities to allocate and order. We need to find a relevant order in which we iterate through the activities of A_{pending} and determine which of them are selected. Actually we do not want to select a too big problem because: (i) we do not have an infinite time to allocate and order it (in any case less than $t_{\text{D}} - t$) and (ii) we do not want to take allocation and ordering decisions too far in the future as they concern data with too much uncertainty. We thus need to monitor two conditions during selection process.

2.1 Selection Order

It is important to select activities of A_{pending} in a relevant order because we need to consider the activities that have priority in terms of resource contention and tardiness costs. We thus need a heuristic for giving the order in which we iterate trough the pending activities and assess them to determine if they are collected and integrated in the sub-problem. I propose to use a heuristic that works out a priority for each process plan. We then consider process plans one by one in the decreasing order of their priority and for each process plan p_i we select the pending activities of A_i in the chronological order given by the precedence constraints and stop when the two conditions defined in Subsection 2.2 are both verified. Figure 3 gives an example of such an order.

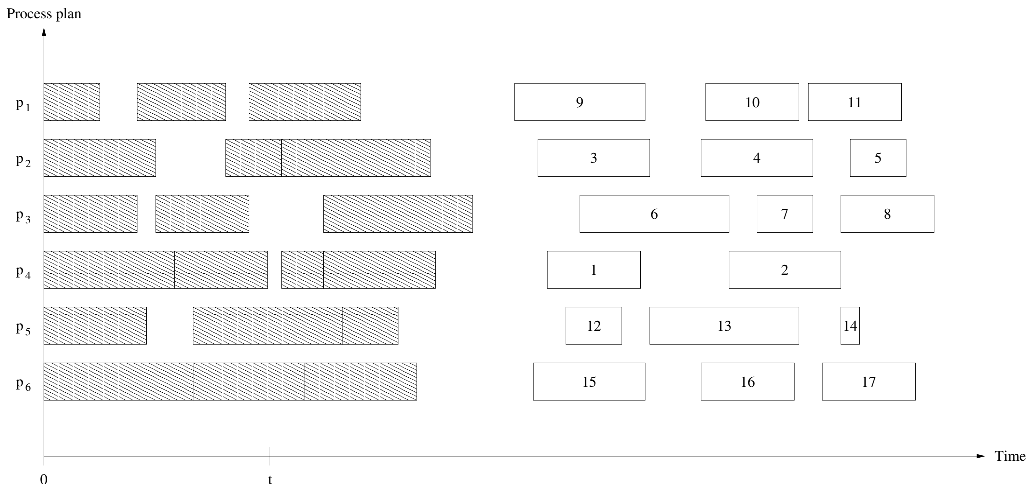

Figure 3: Example of the order in which pending activities are selected

2.1.1 Energetic Reasoning based Heuristic

We compute the priority of each process plan p_i by computing its expected tardiness cost $tardi_i^{\text{exp}}$ is. To estimate how late p_i will finish and thus how much $tardi_i^{\text{exp}}$ is we take resource constraints into account by extending the mean durations of the pending activities by using an energetic reasoning. We suppose that pending activities are elastic. We first compute the energy $energy_{jl}$ consumed by an activity a_l executed with a resource r_j: it is equal to the mean duration of a_l divided by the product of the allocation cost of r_j associated to a_l by the sum of the inverses of the allocation costs of all resources associated to a_l; in a formal way: $energy_{jl} = \frac{mDuration_l}{alloc_{jl} \times \sum_{\forall r_j \in R_l} \frac{1}{alloc_{jl}}}$. It is also useful to compute the energy consumed by the pending activities associated to r_j: $energy_j = \sum_{\forall a_l \in A_{\text{pending}} \cap A_j} energy_{jl}$. We can then compute the criticity degree $crit_l$ of each pending activity $a_l \in A_{\text{pending}}$: $crit_l = 1 + \sum_{\forall r_j \in R_l} \frac{energy_j - energy_{jl}}{energy_j}$. The criticity of an activity a_l represents how high the probability is that the effective duration of a_l will be greater than the original mean duration of a_l given resource constraints and allocation costs. We then extend the mean duration $mDuration_l$ of each pending activity $a_l \in A_{\text{pending}}$ with respect to how critical a_l is: $extendedMDuration_l = mDuration_l \times crit_l$. It is then possible to compute the expected tardiness cost of each process plan p_i by relaxing resource constraints on the pending activities and assuming activities are executed as soon as possible, the durations of the pending activities are equal to their extended mean duration and the durations of the allocated and ordered activities equal their mean duration: $tardi_i^{\text{exp}} = \phi_i \times max(endp_i^{\text{exp}} - due_i, 0)$, where $endp_i^{\text{exp}}$ is the expected end time of p_i. $tardi_i^{\text{exp}}$ gives us the priority π_i of p_i.

2.1.2 ATC Rule based Heuristic

We compute the priority of each process plan by using a modified version of the ATC rule: we redefine the weight in the ATC rule. Each weight $weight_i$ is equal to the sum of the expected tardiness and allocation costs of the process plan p_i: $weight_i = tardi_i^{\text{exp}} + \sum_{a_l \in A_i} alloc_l^{\text{exp}}$, where $tardi_i^{\text{exp}} = \phi_i \times max(endp_i^{\text{exp}} - due_i, 0)$ and $alloc_l^{\text{exp}}$ is the mean allocation cost associated to a_l. We set $k = 2$ in the formula to compute each priority π_i: $\pi_i = \frac{weight_i}{pendingDur_i} \times e^{\frac{-max(0, due_i - t - pendingDur_i)}{k \times \overline{pendingDur}}}$, where $pendingDur_i$ is the mean duration of the pending part of p_i and $\overline{pendingDur}$ is the average of the mean durations of the pending parts of the process plans different from p_i.

2.2 Temporal and Uncertainty Conditions

Given the selection order we have to make sure both that we collect enough activities to keep an anticipation with respect to execution and that the uncertainty degree of each end time of the collected activities is high enough: for each pending activity $a_l \in A_{\text{pending}}$ assessed we compute its expected end time end_l^{exp} by using simulation. We collect the pending activities $a_l \in A_{pending}$ while their mean end time is less than $t + \delta t_{\text{progress}}^{\text{max}}$ or the standard deviation of their end time is less than $\sigma_{\text{progress}}^{\text{max}}$. For computing these expected end times we relax the resource constraints on the pending activities and run a series of simulations by assuming activities are executed as soon as possible: for each activity a_l we randomly pick a set of possible durations and for each resource r_j that could possibly be allocated to a_l we randomly pick a series of breakdowns. If a_l is executed during a breakdown, then its duration is extended. The sub-problem to solve is then defined as all the collected activities. Figure 4 shows a simulation of an activity a_3 associated with three alternative resources r_1, r_3, and r_4. The duration of a_3 equals 15 time units, a_3 starts execution at 5 and finishes at 20; this assumes no associated resources break down during its execution. r_1 breaks down at 25 and is repaired at 30, r_3 breaks down at 10 and is fixed at 35, and r_4 breaks down at 15 and is repaired at 30. From these realizations we can compute the expected end time of a_3, end_3^{exp}: if a_3 is executed with r_1, then its execution is not suspended and finishes at 20; if a_3 is executed with r_3, then its execution is suspended at 10, resumes at 35 and finishes at 45; if a_3 is executed with r_4, then it is suspended at 15, resumes at 20 and finishes at 25. $end_3^{\text{exp}} = \frac{20+45+25}{3} = 30$, it is equal to the mean of the realizations of end_3.

3 How to Allocate and Order the Subset of Activities?

The set of all the collected activities on all the process plans is noted $A_{collected}$ and constitutes the sub-problem. The sub-problem is also composed of precedence constraints and cost constraints; it is deterministic: we suppose resources break down at their mean date and breakdown durations are equal to the mean breakdown durations; we also assume that the durations of the collected activities equal the mean durations of the collected activities.

This sub-problem can be solved by using constraint programming. We can add the two cost constraints to this sub-problem:

$$\sum_{\forall p_i \in P} currentTardi_i \leq K_{\text{tardiness}}$$

where $currentTardi_i = \phi_i \times max(currentEndp_i - currentDue_i, 0)$, $currentEndp_i$ is the end time of the last scheduled activity of p_i, and $currentDue_i = due_i - \sum_{\forall a_l \in A_{\text{pending}} \cap A_i} mDuration_l^{\text{min}}$.

$$\sum_{\forall r_j \in R, \forall a_l \in A_{\text{collected}}} alloc_{jl} < K_{\text{allocation}}^{\text{max}}$$

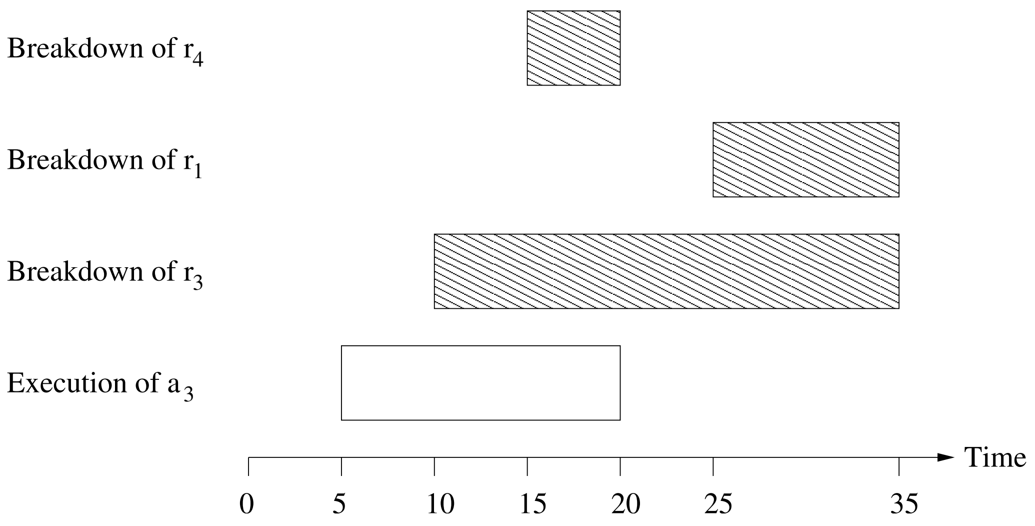

Figure 4: Example of simulation of an unallocated activity and its three alternative resources

where $K^{\max}_{\text{allocation}} = K_{\text{allocation}} - \sum_{\forall r_j \in R, \forall a_l \in A_{\text{pending}}} alloc^{\min}_{jl}$.

4 Comments

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.