USING CONSTRAINT PROGRAMMING AND SIMULATION FOR EXECUTION MONITORING AND PROGRESSIVE SCHEDULING

Julien Bidot [*,**,1] **Philippe Laborie** [*]
J. Christopher Beck [***] **Thierry Vidal** [**]

[*] *ILOG S.A.*
9, rue de Verdun–B.P. 85
94253 Gentilly Cedex, France
[**] *Laboratoire Génie de Production, ENI de Tarbes*
47, avenue d'Azereix–B.P. 1629
65016 Tarbes Cedex, France
[***] *Department of Mechanical and Industrial Engineering*
University of Toronto
5 King's College Road
Toronto, ON M5S 3G8, Canada

Abstract: The problem we tackle is progressive scheduling with temporal and resource uncertainty. Operation durations are imprecise and alternative resources may break down. Operation end times and resource breakdowns are observed during execution. In this paper, we assume we have a representation of uncertainty in the form of probability distributions which are used in the simulation of schedule execution. We generate the schedule piece by piece during execution and use simulation to monitor the execution of the partial schedule. This paper describes the basis on which the decision to select and schedule a new subset of operations is made.

Keywords: Scheduling, Constraint Satisfaction, Monte-Carlo Simulation, Uncertainty, Stability

1. INTRODUCTION

In practical applications, scheduling must take into account imprecise data, incomplete information, and/or potential changes in the environment. The central issue is to design robust scheduling techniques, aimed at guaranteeing the feasibility and the quality of the executed schedule; e.g., see Wu *et al.* (1999). One way among others is to generate a schedule piece by piece to execute. This approach is relevant as far as we do not want to change decisions and scheduling is fast enough w.r.t. the scheduling execution; i.e., when the dynamics of the system are low with respect to those of decision-making. For example, the time spent on machining a workpiece is much longer than the time required for generating a schedule in an automated production workshop. The problem we tackle is a scheduling problem with temporal and resource uncertainty. Operation durations are imprecise and operation end times are observed during execution. Resources may break down, and

[1] Partially supported by Convention Industrielle de Formation par la REcherche 274/2001.

resource breakdown start and end times are observed during execution. In this paper, we assume we have a representation of the uncertainty in the form of probability distributions which are used in the simulation of schedule execution. This paper addresses the basis on which the progressive scheduling is made by presenting simple measures of the data provided by simulation. We have chosen to use constraint programming and simulation to solve this kind of problem. The interest of these techniques is that they can solve large-scale combinatorial problems (a few thousands of operations) and they can easily be extended to solve more complex problems. Propagation algorithms can reduce the search space. Simulation allows to quickly obtain good approximations whatever type of uncertainties and probability distributions are considered. We illustrate our approach on job-shop scheduling problems with imprecise operation durations, alternative resources, and uncertain resources.

2. SCHEDULING PROBLEM

We are interested in scheduling problems with probabilistic operation durations; i.e., operation end times must be observed during execution. Moreover, resources may break down: for each resource, the duration between two consecutive breakdowns is probabilistic and breakdown durations are also probabilistic. In addition, there are alternative unary resources; i.e., an operation can require one of a number of resources. In other words, we address job-shop problems with general allocation, probabilistic operation durations, and uncertain resources.

The problem is a randomly generated $n \times m$ scheduling problem and consists in n process plans. Each process plan p_i is a job and consists in a sequence of m operations (o_{ij}). For each operation, a given number k of randomly picked alternative resources constitute the set R_{ij} of possible resources. There is no temporal constraint between process plans. There are $n \times m$ operations to be processed and the total number of resources equals q.

We assume the following: operations are not *preemptive*; an operation is halted when a resource breaks down and resumes execution when the resource is again available.

2.1 Costs

This scheduling problem is an optimization problem where the objective is to find the schedule that minimizes the average global cost. We distinguish two types of schedule costs: tardiness and allocation costs.

2.1.1. Tardiness Cost Each process plan $p_i \in P$ is associated with a due date due_i. If the last executed operation of p_i finishes later than due_i, then a cost $tardiCost_i^{\text{eff}} = tardi_i^{\text{eff}} \times \phi_i$ is incurred, called *tardiness cost*. $tardi_i^{\text{eff}}$ depends on how late p_i finishes: $tardi_i^{\text{eff}} = \max(endp_i^{\text{eff}} - due_i, 0)$, where $endp_i^{\text{eff}}$ is the effective end time of p_i, observed during execution. A weight, ϕ_i, is applied to each time unit after the due date that the process plan has not yet finished. A schedule is associated with a tardiness cost $K_{\text{tardiness}}$ defined as follows.

$$K_{\text{tardiness}} = \sum_{\forall p_i \in P} tardiCost_i^{\text{eff}}$$

2.1.2. Allocation Cost Another cost, called *allocation cost*, is associated with each resource allocation: when a given operation $o_{ij} \in O$ is executed with a given resource $r_l \in R_{ij}$, it incurs a cost $allocCost_{ijl}$. A schedule is defined by a set of allocations and associated with an allocation cost $K_{\text{allocation}}$: each operation o_{ij} is effectively allocated to a resource $r^{\text{eff}} \in R_{ij}$ and this allocation costs $allocCost_{ij}^{\text{eff}}$.

$$K_{\text{allocation}} = \sum_{\forall o_{ij} \in O} allocCost_{ij}^{\text{eff}}$$

2.1.3. Global Cost The global cost K represents the whole cost of a solution; this cost takes both allocation and tardiness costs into account. It is formally defined as:

$$K = K_{\text{allocation}} + K_{\text{tardiness}}.$$

The choice of this global cost is inspired by project-scheduling applications where the cost of a project depends on how late it is delivered with respect to due dates and what resources have been rented or bought to carry it out. Note that tardiness and allocation costs are antagonistic since the tardiness cost is high if we only want to reduce allocation cost during search by choosing systematically the cheapest resources when allocating operations, and conversely the allocation cost is high if we only take tardiness cost into account during search since this reduces the choice of possible allocations.

Each scheduling problem is characterized by a maximal global cost, $K^{\max}$, that defines an upper bound with respect to the schedule costs. We want to maximize the probability that the global cost is less than the maximal global cost; this can be formally expressed as follows.

$$\max \Pr(K \leq K^{\max})$$

$K^{\max}$ may represent the initial budget dedicated to a project.

3. MONTE-CARLO SIMULATION

For updating unknown probability distributions associated with non-controllable variables such as operation end times, we use Monte-Carlo simulation. A Monte-Carlo simulation run consists in randomly picking a value for each random variable such that the two sets of values that are randomly picked for two consecutive simulation runs may be completely different. For each operation o_{ij} that is allocated and sequenced but not yet executed, we randomly pick a set of possible durations and for each resource that is allocated to o_{ij}, we randomly pick a series of breakdowns.

For running simulation, we use a precedence graph that is generated from the constraint network: each node represents an operation and each arc represents a precedence constraint between two operations. [2] We then topologically sort the precedence graph and use this ordering in the simulations. The simulation horizon equals the sum of all operation durations. The complexity of a single simulation run is in $O(nbBk + nbOp + nbPCt)$, where $nbBk$ is the number of resource breakdowns, $nbOp$ is the number of operations and $nbPCt$ is the number of precedence constraints.

4. PROGRESSIVE APPROACH

A progressive approach for scheduling consists in generating a schedule piece by piece during execution. The first piece of schedule is solved off line. In such an approach, execution and scheduling are interleaved. For implementing such a scheduling approach, we need to define progression conditions that are used to decide when to select and schedule a new subset of operations to anticipate execution, what operations to select, and what data to use when reasoning. When using such an approach, decisions made are never changed later on. Vidal *et al.* (1996) tackled a scheduling problem by using a progressive approach but their problem comprises only imprecise temporal requirements; i.e., they assume resources do not break down.

We propose to decide operation start times just before execution because we want to minimize tardiness cost and we do not want to change decisions. In this way, solutions are flexible and we can be opportunistic with respect to observed operation end times and resource breakdowns. Operations are executed as soon as possible.

[2] There are precedence constraints between process plans if ordering decisions have been made.

The problem with such an 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 the uncertainty level around the decision is low enough so that the decision is well informed, but, on the other hand, we cannot wait too long because we do not just want to have a reactive and myopic decision process. Determining when to select, allocate, and order a new subset of operations will be done based on monitoring a progression criterion during execution. Determining what operations to select will be done using heuristics and Monte-Carlo simulation. Determining how to allocate and order the operations of the selected subset will be done using constraint-based search, possibly combined with Monte-Carlo simulation.

Our progressive approach is characterized by four parameters that can be set to choose indirectly the anticipation horizon and the size of each sub-problem, see Sections 4.1 and 4.2.

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 operations $O_{\text{allocOrder}}$ of the problem have already been allocated and ordered given all constraints at t and the rest of the operations O_{pending} of the problem are not yet allocated and only ordered given the precedence constraints of the original problem. A subset of operations $O_{\text{executed}} \subseteq O_{\text{allocOrder}}$ have already been executed, a second one $O_{\text{executing}} \subseteq O_{\text{allocOrder}}$ are executing, and a third one $O_{\text{toBeExe}} \subseteq O_{\text{allocOrder}}$ have to be scheduled and executed. An operation is scheduled when its start time is fixed at a date before, at, or after the current time. (i) Of course, we do not want to wait until the last operation of O_{toBeExe} finishes execution before allocating and ordering subsequent operations of O_{pending} because we would then have very little time to react and could not easily come up with a good decision, and (ii) 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 operations and perform a combinatorial search on this sub-problem in order to satisfy temporal, resource, and cost constraints. So there are three questions here: (1) how to design conditions that will be monitored during execution and say "now, we can try to extend the current partial flexible schedule by allocating and ordering a new part of the problem," (2) when these conditions are met, how to select the subset of operations to be allocated and ordered, and (3) when the subset of operations is selected, how to allocate and order the operations of this subset in such a way that we

maximize the probability that the constraints will be satisfied and the global cost will be minimal at the end of execution.

4.1 When to Try Extending the Current Partial Flexible Schedule?

To extend the current partial flexible schedule, we need to assess what time we still have before being forced to select, allocate, and order a new subset of operations of O_{pending} and how high the uncertainty level of the end times of the operations of O_{toBeExe} is; i.e., we need to monitor two conditions during schedule execution and when at least one of them is met, we can try to extend the current partial flexible schedule.

We propose to evaluate the trade-off between the fact that $\delta t^{\min}$, the minimal anticipation horizon, should be large enough to have time to perform a combinatorial search leading to a good solution and to ensure the stability of the schedule, and the fact that execution has advanced far enough to get reduced uncertainty on the mean end times of the operations of O_{toBeExe}.

4.1.1. Temporal Condition for Starting Selection Given O_{toBeExe}, there exists a time point t_{D} that is the last time point before which we have to make at least an allocation decision if we want to anticipate execution. t_{D} is equal to the earliest of the mean end times of the operations in $O_{\text{allocOrderLast}} \subseteq O_{\text{toBeExe}}$ that are ordered at last positions in process plans. t_{D} is maintained using Monte-Carlo simulation. We can try to extend the current partial flexible schedule from the date at which $t_{\text{D}} - t \leq \delta t^{\min}$, where $t_{\text{D}} = \min_{\forall p_i \in P}(\max_{\forall o_{ij} \in O_{\text{toBeExe}}}(end_{ij}^{\text{mean}}))$ and end_{ij}^{mean} is the mean end time of operation o_{ij}. We minimize on all process plans to guarantee that the start of selection anticipates execution when $\delta t^{\min} > 0$.

Figure 1 represents the execution of a partial flexible schedule of a 3×6 scheduling problem; $O_{\text{allocOrder}}$ is the subset of operations represented by nine shadowed rectangles; $O_{\text{allocOrderLast}}$ is composed of the three operations represented by the most shadowed rectangles. O_{pending} is composed of the operations represented by white rectangles.

4.1.2. Uncertainty Condition for Starting Selection When the highest standard deviation of the end times of the operations to be executed of a process plan is less than a given standard deviation $\sigma t^{\min}$, we can try to select a subset of pending operations. These standard deviations are maintained using Monte-Carlo simulation. In a more formal way, we extend the current partial flexible schedule from the date at which $\min_{\forall p_i \in P}(\max_{\forall o_{ij} \in O_{\text{toBeExe}}}(\sigma(end_{ij}^{\text{mean}}))) \leq \sigma t^{\min}$, where end_{ij}^{mean} is the mean end time of operation o_{ij}. We minimize on all process plans to guarantee that the start of selection anticipates execution when $\sigma t^{\min} > 0$.

4.2 How to Select the Subset of Operations to Be Allocated and Ordered?

As soon as one of the two conditions defined above is satisfied, we still need to select a subset of operations to allocate and order. We need to find a relevant order in which we iterate through a subset of pending operations and determine which of them are selected. Actually, we do not want to select a too large 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 the selection process. To select the subset of pending operations to be allocated and ordered, we proceed in two steps as follows: (i) we compute and associate priorities to a subset of pending operations called the eligible operations to determine the order in which we assess each of them and (ii) we assess the eligible operations by using a temporal condition and an uncertainty condition, see Section 4.2.2, to determine which of them are selected.

4.2.1. Assessment Order It is important to assess the eligible operations in a relevant order because we need to consider the eligible operations that have priority in terms of resource contention and tardiness costs. An eligible operation is the pending operation that is ordered at the first position of a process plan. There is thus one and only one eligible operation per process plan at the beginning of the selection process. We proceed in several steps. (1) We use a heuristic that gives the order in which we iterate through the current eligible operations and assess the current flexible schedule to determine which of them we select. (2) Once all eligible operations have been labeled by a priority value, we consider each eligible operation in the decreasing order of priority and assess the probability distribution of the end time of its preceding operation with respect to the process plan it belongs to:

- if this distribution does not meet at least one of the two conditions defined in Section 4.2.2, then the eligible operation is selected and no longer eligible (and no longer pending); this selected operation is ordered and allocated in

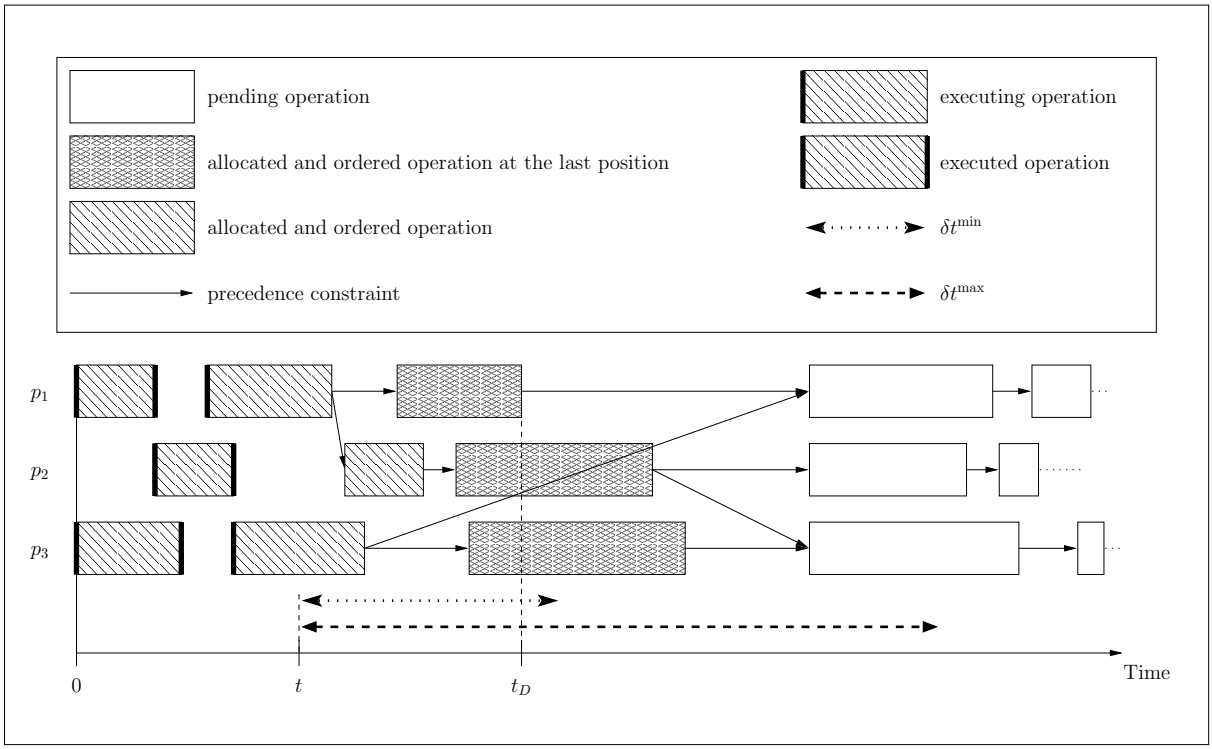

Fig. 1. Example of a schedule progressively generated.

such a way that its mean end time is minimized; the next operation of the same process plan, which is a pending operation, becomes eligible and the priorities of the current eligible operations are then (re)computed;

- if this distribution meets the temporal condition and the uncertainty condition defined in Section 4.2.2, then the eligible operation remains pending and is no longer eligible concerning the current selection process. Both conditions must be met to stop the selection because we want to be sure to select a minimal number of pending operations.

(3) If there are no more eligible operations, then the selection process is stopped, otherwise the selection process goes on by executing alternately (2) and (3).

We assume the schedule execution is stopped during the selection process since the dynamics of the underlying physical system are much lower than the dynamics of the decision-making system.

In fact, we are only interested in the probability distributions of the end times of the operations of $O_{\text{allocOrderLast}}$ and in the probability distributions of resource breakdown end times at time t when selecting and making decisions on a new subset of operations. When an eligible operation is selected, we run a set of simulations concerning this operation, its preceding operations, and the alternative resources it requires to decide what resource to allocate during the selection process. These allocation decisions are not definitive since a combinatorial search is done when the selection is finished, see Section 4.3.

4.2.2. Temporal and Uncertainty Conditions for Stopping Selection Given the assessment order, we have to make sure both that we will select enough pending operations to keep an anticipation with respect to execution and that the uncertainty level of the end time of each selected operation is higher than a given threshold: an eligible operation o_{ij} is not selected and is no longer eligible during the current selection process if the mean end time of its preceding operation of the same process plan o_{ij-1} is greater than $t + \delta t^{\max}$ and the standard deviation of o_{ij-1} is greater than $\sigma t^{\max}$. We assume $\delta t^{\max} \geq \delta t^{\min}$ and $\sigma t^{\max} \geq \sigma t^{\min}$, otherwise we could not select any pending operation.

- If $\delta t^{\max}$ is chosen close to $\delta t^{\min}$ and $\sigma t^{\max}$ is close to $\sigma t^{\max}$, then it amounts to selecting a small number of pending operations because both conditions are met very quickly.
- If $\delta t^{\max} \gg \delta t^{\min}$ and $\sigma t^{\max} \gg \sigma t^{\min}$, then it means that we select a large number of pending operations until we meet both conditions.
- If $\delta t^{\max}$ is chosen close to $\delta t^{\min}$ and $\sigma t^{\max} \gg \sigma t^{\min}$, then it amounts to selecting pending operations until the uncertainty condition is met because the temporal condition is met very quickly.
- If $\delta t^{\max} \gg \delta t^{\min}$ and $\sigma t^{\max}$ is chosen close to $\sigma t^{\min}$, then it means that we select pending

operations until the temporal condition is met because the uncertainty condition is met very quickly.

4.2.3. Heuristic The energy-based heuristic is combined with the Apparent Tardiness Cost rule based heuristic, see Aktürk and Yildirim (1999), to compute the priorities of eligible operations, and the estimations of the process plan queues and the allocation costs of pending operations. We take resource constraints into account by extending the mean durations of the pending operations by using an energy-based heuristic, see Erschler (1976); i.e., we compute the criticality of each operation that depends on the average loads and costs of the resources it requires, we then modify its mean duration accordingly.

4.3 How to Allocate and Order the Subset of Operations?

The set of all the selected operations on all the process plans constitutes the sub-problem to solve. After the selection process is finished, we need to approximate the contribution of each process plan in terms of cost; i.e., the allocation cost and the length of the pending operations of each process plan. This approximation is done by using the same heuristic as the one dedicated to the selection of operations. To make decisions, we generate a deterministic problem; i.e., we use the mean durations of the selected operations extended depending on resource breakdown distributions, and process plan queues estimated heuristically. We use standard constraint-programming techniques to explore and reduce the search space.

5. DISCUSSION AND FUTURE WORK

If $\delta t^{\max}$ and $\sigma t^{\max}$ are small, then it means we frequently extend the current partial flexible schedule with small subsets of operations: this is a reactive approach. If $\delta t^{\max}$ and/or $\sigma t^{\max}$ are large, then it amounts to selecting and scheduling all operations: this is a predictive approach.

We can change both $\delta t^{\min}$ and $\sigma t^{\min}$ to choose when we want to consider a subset of pending operations during execution of the current flexible schedule. If $\delta t^{\min}$ is small and $\sigma t^{\min}$ is large, then it amounts to extending the current flexible schedule when the uncertainty condition is met. If $\delta t^{\min}$ is large and $\sigma t^{\min}$ is small, then it means we extend the current flexible schedule when the temporal condition is met. If both $\delta t^{\min}$ and $\sigma t^{\min}$ are small, then it means we extend the current flexible schedule at the last time: the temporal anticipation is short. If both $\delta t^{\min}$ and $\sigma t^{\min}$ are large, then it amounts to extending the current flexible schedule very early: the temporal anticipation is long.

An experimental study of this approach has still to be conducted to understand the relationships between the different and numerous parameters, indicators, and problem characteristics. A preliminary work in this direction has consisted in generating problem instances.

An interesting future work will focus on mixing this progressive approach with a proactive approach and/or a revision approach, and in studying the relationships between the different parameters, indicators, and problem characteristics. The following approaches are potential candidates. Bidot *et al.* (2003) presented a revision approach for tackling job-shop scheduling problems with probabilistic, temporal data. Beck and Wilson (2005) were interested in studying proactive techniques for tackling the same scheduling problems. For example, we could set a mixed approach that generates solutions with a probability of exceeding the global cost less than 0.15, a middle progression horizon, and a small number of reschedulings.

REFERENCES

Aktürk, M.S. and M.B. Yildirim (1999). A new dominance rule for the total weighted tardiness problem. *Production Planning and Control* **10**(2), 138–149.

Beck, J.C. and N. Wilson (2005). Proactive algorithms for scheduling with probabilistic durations. In: *Proceedings of the 19th International Joint Conference on Artificial Intelligence (IJCAI'05)*. Edinburgh, Scotland. pp. 1201–1206.

Bidot, J., P. Laborie, J.C. Beck and T. Vidal (2003). Using simulation for execution monitoring and on-line rescheduling with uncertain durations. In: *Working Notes of the ICAPS'03 Workshop on Plan Execution*. Trento, Italy.

Erschler, J. (1976). Analyse sous contraintes et aide à la décision pour certains problèmes d'ordonnancement. Ph.D. dissertation. Université Paul Sabatier, Toulouse, France.

Vidal, T., M. Ghallab and R. Alami (1996). Incremental mission allocation to a large team of robots. In: *Proceedings of the 1996 IEEE International Conference on Robotics and Automation (ICRA'96)*. Minneapolis, Minnesota, United States of America. pp. 1620–1625.

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