

Explanation-based repair techniques for solving dynamic scheduling problems

Abdallah Elkhyari¹ and Christelle Guéret^{1,2} and Narendra Jussien¹

¹ École des Mines de Nantes – BP 20722 – F-44307 Nantes Cedex 3 – France

² IRCCyN – Institut de Recherche en Communications et Cybernétique de Nantes

email: {aelkhyar, gueret, jussien}@emn.fr

Keywords: Dynamic scheduling, repair-based techniques, explanation-based constraint programming, Resource-Constrained Project Scheduling Problem.

Introduction

Scheduling problems have been studied a lot over the last decade. Due to the complexity and the variety of such problems, most works consider static problems in which activities are known in advance and constraints are fixed. However, every scheduling problem is subject to unexpected events (consider for example a new activity to schedule, or a machine breakdown). In these cases, a new solution is needed in a preferably short time taking these events into account and as close as possible to the current solution.

In this paper, we present an exact approach for solving dynamic scheduling problems. This approach uses explanation-based constraint programming and operational research techniques. Our tools have been designed for a general scheduling problem: the Resource-Constrained Project Scheduling Problem (RCPSp).

Problem description

The RCPSp can be defined as follows: let $A = \{1, \dots, n\}$ be a set of activities, and $R = \{1, \dots, r\}$ a set of renewable resources. Each resource k is available in a constant amount R_k . Each activity i has a duration p_i and requires a constant amount r_{ik} of the resource k during its execution. Preemption is not allowed. Activities are related by precedence constraints, and resource constraints require that for each period of time and for each resource, the total demand of resource does not exceed the resource capacity. The objective considered here is to find a solution for which the end of the schedule is minimized. This problem, denoted by PS/prec/C_{max} (Brucker *et al.* 1999), is NP-hard (Blazewicz, Lenstra, & Rinnoy Kan 1983).

The static RCPSp has been extensively studied (Brucker *et al.* 1998). A major difficulty in this problem is to maintain the resource limitation over the horizon time. Several deduction rules exist: *core-times* (Klein & Scholl 1999), *energetic-reasoning* (Erschler & Lopez 1990), *task-interval* (Caseau & Laburthe 1996), etc.

The dynamic RCPSp is seldom studied. Two classical methods are used to solve it:

- recomputing a new schedule each time an event occurs. This is quite time consuming and may lead to a solution quite different from the previous one.

- building a partial schedule and completing it progressively as time goes by (like in on-line scheduling problems). In this case the schedule cannot be constructed in advance.

Recently, (Artigues & Roubellat 2000) introduced a formulation of the RCPSp based on a flow network model. They developed a polynomial algorithm based on this model in order to be able to insert an unexpected activity.

Explanation-based constraint programming

Constraint programming techniques have been widely used to solve scheduling problems (Klein 1999). Constraint programming is based upon the notion of *constraint satisfaction problems*.

A *constraint satisfaction problem* (CSP) consists in a set V of variables defined by a corresponding set of possible values (the domains D) and a set C of constraints. A solution for the CSP is an assignment of the variable such that all the constraints are satisfied.

Explanation-based constraint programming (*e-constraints*) has already proved its interest in many applications (Jussien 2001). This section recalls what is an explanation and how it can be used.

Explanations

In the following, we consider a CSP (V, D, C) . Decisions (variable assignments) made during the enumeration phase of the resolution of this problem correspond to adding or removing constraints from the current constraint system (*eg.*, upon backtracking).

A **conflict set** (*a.k.a. nogood*) is a subset of the current constraints system of the problem that, left alone, leads to a contradiction (no feasible solution contains a conflict set). A conflict set divides into two parts: a subset of the original set of constraints ($C' \subset C$) and a subset of decision constraints introduced so far in the search: $\neg(C' \wedge v_1 = a_1 \wedge \dots \wedge v_k = a_k)$.

In a conflict set composed of at least one decision constraint, a variable v_j is selected and the previous formula can be rewritten as¹: $C' \wedge \bigwedge_{i \in [1..k] \setminus j} (v_i = a_i) \implies v_j \neq a_j$

¹A conflict set that does not contain such a constraint denotes an over-constrained problem.

The left hand side of the implication constitutes an **eliminating explanation** for the removal of value a_j from the domain of variable v_j and is noted $\text{expl}(v_j \neq a_j)$.

Classical CSP solvers use domain-reduction techniques (removal of values). Recording eliminating explanations is sufficient to compute conflict sets. Indeed, a contradiction is identified when the domain of a variable v_j is emptied. A conflict set can easily be computed from the eliminating explanations associated with each removed value:

$$\neg \left(\bigwedge_{a \in D_{v_j}} \text{expl}(v_j \neq a) \right)$$

There exist generally several eliminating explanations for the removal of a given value. Recording all of them leads to an exponential space complexity. Another technique relies on *forgetting* (erasing) eliminating explanations that are no longer relevant² in the current variable assignment. By doing so, the space complexity remains polynomial. We keep only **one** explanation at a time for a value removal.

Using explanations

Explanations can be used in several ways (Jussien 2001). For example, when debugging, explanations can be used to: **clearly** explain failures, explain differences between intended and observed behavior for a given problem (why is value n not assigned to variable x ?).

Explanations can also be used to determine direct or indirect effects of a given constraint on the domains of the variables of the problem, and for dynamic constraint removal. This is the case with the justification system used in (Bessière 1991) for solving dynamic CSP. This justification system is actually a partial explanation system. Moreover, being able to explain failure and to dynamically remove a constraint facilitates the building of dynamic over-constrained problem solvers.

We added explanation handling within a branch and bound algorithm in order to provide a dynamic RCPSPP problem solver.

Solving dynamic RCPSPP

We developed an environment for solving dynamic RCPSPP that is based upon:

- a branch and bound algorithm (inspired from (Brucker *et al.* 1998)) within a constraint programming solver: at each node, deduction rules are applied in order to determine redundant information. In the following, we call *constraint* each initial constraint of the problem (precedence and resource constraints), but also each decision taken by the branching scheme, and each deduction made (thanks to propagation rules) during the search.
- an extensive use of *explanations*. Explanations are recorded during the search, and improved thanks to propagation rules (namely *core-times* and *task-interval*) which have been upgraded in order to provide a precise explanation for every deduction made.

²An explanation is said to be relevant if all the decision constraints in it are still valid in the current search state.

Our environment can efficiently handle dynamic events. Indeed, an unexpected event leads to add, modify or remove a constraint in the system. In the first two cases, if the current solution is no more valid, then the explanations tell us which are the constraints responsible of the contradiction. Repairing is done by removing at least one constraint from the explanation. The resulting solution is generally quite similar to the previous one, and is found faster than if we have had to solve the problem from scratch.

Notice that, as we saw before, if the constraints of the explanation are only initial constraints of the problem, then the problem is over-constrained. In this case, our system explains to the user why the problem has no solution. Responsible constraints can therefore be relaxed.

Conclusion

We developed a dynamic environment for solving general scheduling problems which can let the user interact with his/her problem: define a new time-window for an activity, add/remove precedence constraints, add/remove *must-overlap* constraints (*i.e.* stating that two activities must overlap), define/reduce/remove a required time-lag between two activities, etc.

We are currently experimenting our tools in order to show the interest of a real dynamic approach compared to re-execution from scratch. We do expect promising results.

References

- Artigues, C., and Roubellat, F. 2000. A polynomial activity insertion algorithm in a multi-resource schedule with cumulative constraints and multiple modes. *European Journal of Operational Research* 127(2):297–316.
- Bessière, C. 1991. Arc consistency in dynamic constraint satisfaction problems. In *Proceedings AAAI'91*.
- Blazewicz, J.; Lenstra, J.; and Rinnoy Kan, A. 1983. Scheduling projects subject to resource constraints: classification and complexity. *Discrete Applied Mathematics* 5:11–24.
- Brucker, P.; Knust, S.; Schoo, A.; and Thiele, O. 1998. A branch and bound algorithm for the Resource-Constrained Project Scheduling Problem. *European Journal of Operational Research* 107:272–288.
- Brucker, P.; Drexl, A.; Möring, R.; Neumann; and Pesch, E. 1999. Resource-constrained project scheduling: notation, classification, models and methods. *European Journal of Operational Research* 112:3–41.
- Caseau, Y., and Laburthe, F. 1996. Cumulative scheduling with task-intervals. In *Joint International Conference and Symposium on Logic Programming (JICSLP)*.
- Erschler, J., and Lopez, P. 1990. Energy-based approach for task scheduling under time and resource-constraints. In *Second international workshop on Project Management and Scheduling (PMS)*, 115–121.
- Jussien, N. 2001. e-constraints: explanation-based constraint programming. In *CP01 Workshop on User-Interaction in Constraint Satisfaction*.
- Klein, R., and Scholl, A. 1999. Computing lower bounds by destructive improvement: an application to Resource-Constrained Project Scheduling Problem. *European Journal of Operational Research* 112:322–345.
- Klein, R. 1999. *Scheduling of Resource Constraints Projects*. Boston: Kluwer Academic.