

Stable solutions for dynamic project scheduling problems

A. Elkyari¹, C. Guret^{1,2}, and N. Jussien¹

¹ École des Mines de Nantes – 4 rue Alfred Kastler – BP 20722
F-44307 Nantes Cedex 3 – France

e-mail: Abdallah.Elkhyari, Christelle.Gueret, Narendra.Jussien@emn.fr

² IRCCyN: Institut de Recherche en Communications et Cyberntique de Nantes
France

Keywords. T1.1: project scheduling , T4.1: flexible and robust scheduling, T3.6.: constraint programming and scheduling

The Resource Constrained Project Scheduling Problem (RCPSP) is a general scheduling problem which consists in scheduling a set of activities taking into account temporal and resource constraints (Demeulemeester and Herroelen, 2002). Preemption is not allowed. The objective considered here is the minimization of the makespan (total duration) of the project. This problem is *NP-hard* (Blazewicz et al., 1983).

Most work about RCPSP consider static problems in which activities are known in advance and constraints are fixed. However, every schedule is subject to unexpected events (consider for example a new activity to schedule, or a resource failure – *eg.* machine breakdown). When such a situation arises, a new solution taking these events into account is needed generally in a short time. Furthermore, this new solution must preferably be not too far from the previous one.

Several works concern dynamic scheduling problems. But generally, they deal with very specific problems like one-machine problems (Mehta et Uzsoy, 1999;, Aloulou et Portmann, 2002) or m-processors (Moukrim et al. 1999), and the number of types of events taken into account are limited. Furthermore, to our knowledge, no optimal approach has been proposed for any dynamic scheduling problems. The only approach concerning the dynamic RCPSP has been recently proposed by (Artigues et al., 2000). The authors developed a heuristic based on a flow network model to update an initial static schedule when considering the insertion of an unexpected activity.

Conversely, constraint programming is increasingly used for solving scheduling problems as its flexibility is well suited for real-life scheduling problem. Moreover, solving dynamic constraint satisfaction problems (DCSP) is a vivid research topic in the constraint programming community. A DCSP (Dechter and Dechter, 1988) is a constraint satisfaction problem whose set of variables or/and constraints evolves throughout computation leading to a series of problems differing one from the other by the addition/retraction of single variable/constraint. However, no technique, as far as we know, has been proposed yet to handle dynamic scheduling problems.

We introduce here a constraint-based technique to handle dynamic RCPSP instances. This technique which provides optimal solutions is able to handle a large number of different unexpected events and computes a new solution quicker than when solving the new problem from scratch. Moreover, the successive computed solutions are much more stable than solutions computed from scratch.

1 An explanation-based constraint programming technique

Regarding dynamic constraint satisfaction problems, several techniques exist: *pro-active* techniques that use a model of the potential perturbations of the initial problem and *re-active* methods that handle the perturbation as it appears. As far as the latter set of techniques is concerned, a recent extension of constraint programming can be used: explanation-based constraint programming (Debruyne et al., 2003).

Efficiently solving dynamic constraint satisfaction problems requires incremental addition and retraction of constraints. Even though incremental constraint addition is naturally handled by modern constraint solvers, incremental retraction of constraints is often performed with recording trace/undo information. Such an information is used to determine past effects of removed constraint that need to be undone. *Explanations* for constraint programming are a generalization of that information. An explanation is a set of constraints that justifies an action of the solver (classically value removals) *i.e.* as long as each constraint that appears in the explanation remains active (not removed) the value removal is valid; thus, no valid solution can be build from this partial assignment.

Explanations are used to determine the past effects of a given constraint in order to be able to retract it incrementally. Explanation-based constraint programming techniques need two prerequisites: all handled constraints need to be enhanced with explanation capabilities; the branching scheme used during the enumeration phase need to make decisions represented as a single constraint and whose negation is also a single constraint. To meet those two requisites for solving dynamic RCPSP problems, we enhanced existing resource management constraints (*core-times* (Klein and Scholl, 1999), *task-intervals* (Caseau and Laburthe, 1996)) with explanation capabilities (Elkhyari et al., 2002). Moreover, we introduced a notion of distance between activities that is used to define branching decisions with simple constraints whose negations are also simple.

2 An environment for solving dynamic RCPSP

Our dynamic RCPSP solving system is based on a branch and bound algorithm inspired from (Brucker et al., 1998). This branch and bound algorithm has been enhanced with explanation capabilities as in (Guret et al., 2000). We are capable of dynamically handling a various set of events: (a) **temporal events**: addition, retraction, or modification of classical and generalized precedence, disjunction or overlapping constraints between two activities which is useful to handle, for example, new constraints arising from the arrival of a new urgent order, or, new technical constraints, etc. (b) **activity related events**: addition, retraction, or modification of an activity which is useful to handle new orders, order removals, duration or resource needs modifications, etc. (c) **resource related events**: addition, retraction, or modification of a resource which is useful to handle machine breakdowns, newly repaired machines, etc.

The process used to solved dynamic RCPSP instances is the following: an optimal solution for the original problem is computed; unexpected events are handled incrementally from this first solution: (a) upon the addition of a new information: if a conflict with the current solution is identified, a re-optimization is asked (the

conflict is analyzed using recorded explanations, some past decisions are undone, some new decisions are made until a new optimal solution is obtained); otherwise, nothing happens; (b) upon the removal of an existing information: the related constraints are incrementally removed and re-optimization is performed (starting from the current solution and stored information about past search). Modifications are considered as: first, an information removal and, second, an information addition.

Notice that adding, removing, and modifying constraints can quite quickly lead to an unfeasible instance. Thanks to the explanations, our system is able to provide some information to the user in order to let him/her know what were the subset of modifications that leads to this situation letting him/her modify things in order to get back to a feasible instance (Debruyne et al., 2003).

3 Experiments

We give here the results obtained in terms of computation time and stability of successive solutions when comparing our approach with a re-execution from scratch after each modification. Stability can be measured by several parameters. We report here the number of relative positions of activities that change between consecutive solutions (the POSI measure). As no benchmarks exists regarding dynamic RCPSP, we used the SMCP (*Single-Mode Project Scheduling*) series (Kolish et al, 1995): 200 problems broken into 20 series of 10 problems. For each instance, we randomly chose a set of activities to be removed and named it P_0 . P_i the problem obtained after i perturbations. Therefore, P_n (n corresponding to the last perturbation depends on the size of the problem) corresponds to the original problem of SMCP series.

Table 1 reports improvement results (in percent compared to the from scratch rescheduling) obtained when considering activities additions on problems with 12, 22, and 32 activities. These results are given for the successive problems and the last column reports the overall improvement (or loss – negative values).

As far as computation time is concerned average improvement lies between 10.7% and 78.2%, and average general improvement $GAVG$ is really important: between 43.7% and 90.5%. The larger the instance, the more improvement is brought by our technique. The average improvement of stability lies between -51.8% and 81% . The table show a great improvement for problems with 32 activities. Once more, large instances get more benefits than small instances from our techniques.

	$P_0 \rightarrow P_1$	$P_1 \rightarrow P_2$	$P_2 \rightarrow P_3$	$P_3 \rightarrow P_4$	$GAVG$		$P_0 \rightarrow P_1$	$P_1 \rightarrow P_2$	$P_2 \rightarrow P_3$	$P_3 \rightarrow P_4$	$P_0 \rightarrow P_{end}$
12 ×4	37.7%	32.2%	–	–	43.7%	12 ×4	-24%	-51.8%	–	–	-58.3%
22 ×4	29.7%	5.8%	34.4%	–	78.9%	22 ×4	18.2%	12.5%	-14.6%	–	-19%
32 ×1	58.3%	68.5%	77%	46.7%	83.8%	32 ×1	73.1%	79%	81%	-27.6%	11.2%
32 ×2	10.7%	57.8%	68.7%	62.8%	84.9%	32 ×2	53.7%	37.4%	38.2%	63.5%	37.5%
32 ×3	61.4%	66%	74.6%	78.2%	90.5%	32 ×3	54.3%	45.8%	76.5%	33.3%	21.1%
32 ×4	57%	63.5%	60.7%	78.2%	90%	32 ×4	36.7%	62.1%	34.1%	49.7%	27.3%

Table 1. Computation time (left) and stability (using the POSI measure – right) improvements when considering activity additions.

4 Conclusion

We introduced here a new approach for solving dynamic RCPSP instances. Our proposal is based on new constraint programming techniques and provides a complete system able to handle a wide set of perturbations. Moreover, our experimental results clearly show the interest of this approach which provides in a short time very stable solutions when incrementally handling unexpected events for RCPSP problems. We are currently working on improvements of our system to handle other problems (*eg.* timetabling problems).

References

- ALLOULOU, M.A. and PORTMANN, M.C. (2002): A genetic algorithm to achieve scheduling flexibility for a single machine problem. submitted to RAIRO-OR.
- ARTIGUES, C. and ROUBELLAT, F. (2000): A polynomial activity insertion algorithm in a multi-resource schedule with cumulative constraints and multiple modes. In: *European Journal of Operational Research*, 127(2):179–198.
- BESSIÈRE, C. (1991): Arc consistency in dynamic constraint satisfaction problems. In *Proceedings AAAI'91*.
- BLAZEWICZ, J., LENSTRA, J.K. and RINNOY KAN, A.H.G. (1983): Scheduling projects subject to resource constraints: classification and complexity. In: *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. In: *European Journal of Operational Research*, 107:272–288.
- CASEAU, Y. and LABURTHE, F. (1996): Cumulative scheduling with task-intervals. In: *Joint International Conference and Symposium on Logic Programming (JICSLP)*.
- DEBRUYNE, R., FERRAND, G., JUSSIEN, N., LESAIN, W., OUIS, S. and TESSIER, A. (2003): Correctness of Constraint Retraction Algorithms. In: *Proceedings FLAIRS'03*, AAAI press, pages 172–176.
- DECHTER, R. and DECHTER, A. (1988): Belief maintenance in dynamic constraint networks. In: *Proceedings AAAI'88*, pages 37–42, St Paul, MN.
- DEMEULEMEESTER, E.L. and HERROELEN W.S. (2002): Project Scheduling – A Research Handbook, chapters 9–10, Kluwer Academic Publishers, Boston.
- ELKHYARI, A., GUÉRET, C. and JUSSIEN, N. (2002): Conflict-based repair techniques for solving dynamic scheduling problems. In: *Proceedings CP 2002*, LNCS 2470, Springer-Verlag, pages 702–707.
- GUÉRET, C., JUSSIEN, N., and PRINS, C. (2000): Using intelligent backtracking to improve branch and bound methods: an application to open-shop problems. In: *European Journal of Operational Research*, 127(2):344–354.
- KLEIN, R. and SCHOLL, A. (1999): Computing lower bounds by destructive improvement: an application to Resource-Constrained Project Scheduling Problem. In: *European Journal of Operational Research*, 112:322–345.
- KOLISCH, R. SPRECHER, A. and DREXL, A. (1995): Characterization and generation of a general class of resource-constrained project scheduling problems. In: *Management Science*, volume 41, pages 1693–1703.
- METHA, S.V. and UZSOY R. (1999): Predictable scheduling of a single machine subject to breakdowns. In: *International Journal of Computer Integrated Manufacturing*, 12:15–38.
- MOUKRIM, A. and SANLAVILLE, E. and GUINAND, F. (1999): Scheduling with Communication Delays and On-Line Disturbances. In: *Proceedings of Euro-Par'99 - Parallel Processing: 5th International Euro-Par Conference*, LNCS ISSN: 0302-9743.