Replanning in Predictive-reactive Scheduling

Marek Vlk and Roman Barták (supervisor)
Charles University in Prague, Faculty of Mathematics and Physics
Malostranské nám. 25, 118 00 Praha 1, Czech Republic
{vlk, bartak}@ktiml.mff.cuni.cz

Abstract

Achieving optimal results in real-life production scheduling is precluded by a number of problems. One such problem is dynamics of environments with unavailable resources (such as machine breakdowns and ill workers) and new demands (e.g. new orders) coming during the schedule execution. Traditional approach to react to unexpected events occurring on the shop floor is generating a new schedule from scratch. Complete rescheduling, however, may require excessive computation time. Moreover, the recovered schedule may deviate a lot from the ongoing schedule. Some work has focused on tackling these shortcomings, but none of the existing approaches tries to substitute jobs that cannot be executed with a set of alternative jobs. This paper reviews techniques related to predictive-reactive scheduling and suggests the future goal, which is to propose algorithms for dealing with unexpected events using the possibility of alternative processes.

Introduction

Scheduling as a decision-making process, of which the aim is to allocate limited resources to activities so as to optimize certain objectives, has been paid a lot of attention. In manufacturing environment, developing a detailed schedule of the activities to be performed helps maintain efficiency and control of operations.

In the real world, however, manufacturing systems face uncertainty due to unexpected events occurring on the shop floor. Machines break down, operations take longer than anticipated, personnel do not perform as expected, urgent orders arrive, others are cancelled, etc. These disturbances render the ongoing schedule infeasible. In such case, a simple approach is to collect the data from the shop floor when the disruption occurs and to generate a new schedule from scratch. Since scheduling problems are usually NP-hard, complete rescheduling may require excessive computation time, and, moreover, the recovered schedule may deviate prohibitively from the ongoing schedule.

For these reasons, reactive scheduling, which may be understood as the continuous correction of precomputed predictive schedules, is becoming more and more important. On the one hand, reactive scheduling has certain things in common with some predictive scheduling approaches, such as iterative improvement of some initial schedule. On the other hand, the major difference between reactive and predictive scheduling is the on-line nature and associated real-time execution requirements. The schedule update must be accomplished before the running schedule becomes invalid, and this time window may be very short in a complex manufacturing environment.

Several novel sophisticated methods attempt to cope with the shortcomings of complete rescheduling, e.g., by rescheduling only the activities somehow affected by the disturbance. To the best of our knowledge, none of the existing approaches, however, tries to replace activities, which cannot be processed or must be reallocated to other resources, by a set of alternative activities using other (available) resources. Our goal is to propose, implement and experimentally compare algorithms for modifying a schedule to accommodate disturbances, such as a resource failure, using the possibility of alternative processes, i.e., to replan the influenced part of the schedule.

Terminology

The *rescheduling framework*, presented in (Vieira, Herrmann, and Lin 2003), introduces *rescheduling environments*, identifying the sets of jobs to schedule (static vs. dynamic), *rescheduling strategies*, describing whether or not schedules are generated, *rescheduling policies*, specifying when to reschedule (periodic, event-driven, or hybrid), and *rescheduling methods*, describing how to generate and update schedules (complete rescheduling, and schedule repair).

Rescheduling Strategies

There are two basic strategies for controlling production in dynamic environments with uncertain job arrivals. The first strategy does not create production schedules, but decentralized production control methods dispatch jobs when necessary, using information available at the moment of dispatching. Jobs waiting for processing at a resource are chosen by using a dispatching rule or other heuristics. This strategy is often referred to as *completely reactive scheduling*, on-line scheduling, or dynamic scheduling. It is useful for very dynamic environments, where it is not known in advance which activities are to be processed. On the other

hand, by this strategy it is obviously hard to obtain a schedule close to the optimal one (the quality can be guaranteed under certain conditions, but these conditions are in practise hard to enforce).

The second strategy is usually called *predictive-reactive scheduling*. It has two primary steps. The first one generates a production schedule; the second one corrects the schedule in response to disturbances and other changes within the environment.

According to (Ouelhadj and Petrovic 2009), predictive-reactive scheduling is based on schedule modifications considering only shop efficiency. In such case the new schedule may be fundamentally deviated from the original one, which may cause poor performance owing to affecting other planning activities based on the original schedule. For these reasons, the following two strategies (approaches) may be distinguished.

Robust predictive-reactive scheduling takes a focus on generating schedules to minimize the impacts of disruptions on the performance. A usual solution is to consider not only schedule efficiency, but mainly deviation from the original schedule (termed *stability*). Similarity of two schedules may be formally defined for example as a *minimal perturbation problem* (Barták, Müller, and Rudová 2003).

In some cases the disruptions are predictable or even expected, which may be exploited when computing a predictive (original) schedule. This is the aim of *robust pro-active scheduling*[1]. Such methods usually add some temporal slack among operations of a job in order to absorb some level of uncertainty without rescheduling.

Related Methods

When dealing with disturbances, reactive scheduling systems usually attempt to revise only necessary parts of the ongoing schedule to avoid rescheduling from scratch, because schedule repair methods have the advantage in terms of computation time and stability. Such methods are reviewed in this section, based on the classification in (Ouelhadj and Petrovic 2009) and (Raheja and Subramaniam 2002).

The predictive schedules are usually based on optimisation principles. It is obvious that any correction will cause a deviation from the predictive schedule and thus performance measures will no longer be optimal. Therefore, the focus has been to find a technique that handles the schedule recovery without deterioration in the quality.

On the other hand, the schedule must be recovered and ready to replace the ongoing schedule while it is still active, i.e., before it becomes infeasible or obsolete. Otherwise the manufacturing process fails. For these reasons, the time span needed by the algorithm to react to unexpected events must also be considered when evaluating the schedule repair performance.

[1]To make matters more confusing, robust pro-active scheduling is also referred to as robust predictive-reactive scheduling and then robust predictive-reactive scheduling amalgamates with predictive-reactive scheduling.

Heuristics

Heuristic-based approaches do not guarantee to find an optimal solution, but respond in a short time. The simplest schedule repair technique is the *right shift rescheduling* (Abumaizar and Svestka 1997). This algorithm shifts the operations globally to the right on the time axis in order to cope with disruptions. When it arises from machine breakdown, the method introduces gaps in the schedule, during which the machines are idle. It is obvious that this approach results in schedules of bad quality, and can be used only for environments involving minor disruptions.

The shortcomings of total rescheduling and right shift rescheduling gave rise to another approach: *affected operation rescheduling* (Smith 1995), also referred to as partial schedule repair. The idea of this algorithm is to reschedule only the operations directly and indirectly affected by the disruption in order to minimize the deviation from the initial schedule.

The *Prec-rep* algorithm proposed in (Barták and Skalický 2009) is worth mentioning although it is designed not for the recovery of schedules being executed, but for repairing violated precedence and resource constraints in manually altered schedules. The algorithm sweeps over the violated constraints and repairs them in such a way that the activity that is to precede another one is shifted to the left and the succeeding one to the right. This gives significantly better results when compared to right shift rescheduling.

Meta-heuristics

Meta-heuristics such as simulated annealing or genetic algorithms are high level heuristics guiding local search methods to escape from local optima. Local search methods are based on probing into neighbourhoods, i.e., the algorithms start from some given solution and iteratively improve it using move operators. Thus, when such algorithm reaches a solution that cannot be directly improved, which means that the algorithm gets stuck in a local optimum, it terminates. Meta-heuristics help these techniques to jump from local optima by (occasional) accepting worse solutions or by generating better initial solutions for local probing in some sophisticated way.

Since the local search heuristics may be trapped in a poor local optimum, using meta-heuristics is a good way to their enhancement. On the other hand, meta-heuristics require higher computational effort, which holds especially for genetic algorithms with increasing size of a problem.

An example of integrating local search and heuristic procedures is the *iterative flattening search* (Cesta, Oddi, and Smith 2000), which has been designed for finding predictive schedules minimizing makespan. The algorithm iterates two steps. First, in the relaxation step, some precedence constraints, which have been added into the model so as to resolve resource restrictions, are retracted. Second, in the flattening step, some precedence constraints are added into the model in order to make the schedule feasible again. The iterative flattening search is reported to have been enhanced by tabu search meta-heuristic technique to achieve fine-grained exploration, and by introducing partial order schedules aimed

at increasing temporal flexibility in the temporary solutions (Oddi et al. 2007).

Artificial Intelligence Approaches

Some techniques from the field of artificial intelligence and knowledge-based systems are also applied in rescheduling. *Case-based reasoning* (Cunningham and Smyth 1997) is applicable to domain specific problems and allows continuous learning from past cases, but requires an extensive experience database, which involves time-consuming search through. *Fuzzy logic* (Ramkumar, Tamilarasi, and Devi 2011) requires the knowledge of the domain to be built into the algorithm and learning of the algorithm is impossible. *Neural networks* (Jain and Meeran 1998) provide very fast responses and predict the repair strategy according to past experience, which, however, may require excessive re-training time.

Another approach, which is rather an independent branch, is *multi-agent based architectures* (Zhang et al. 2011). In multi-agent systems independent agents cooperate in order to achieve a common goal. Albeit this is one of the most promising approaches to building complex, robust, and effective scheduling systems owing to their distributed, autonomous and dynamic nature, but the coordination among the agents is hard to achieve.

Predictive-reactive Targeted

Some works focused directly on predictive-reactive scheduling. A predictive-reactive approach for the single machine problem with minimization of both makespan and total weighted tardiness was proposed in (Aloulou, Portmann, and Vignier 2002). In predictive phase, a set of schedules with partially ordered activities is constructed, and every time a decision should be taken, a decision-maker can pick the most suitable solution from a set of alternatives following from the temporal and job sequencing flexibility. In order to construct flexible predictive schedules, a Multi-Objective Genetic Algorithm (MOGA) is used.

Next, (Asudegi and Haghani 2013) introduced the problem of construction equipment scheduling for a company with several ongoing projects in different regions, which is modelled as linear optimization problem and experimentally evaluated by Xpress-IVE 7.0 solver.

Further, (Dulai, Werner- Stark, and Hangos 2013) presented a model and an algorithm generating a predictive schedule of production workflows that is (proactively) robust with regard to so called immediate events, which include breakdown of a workstation and faulty termination of a workflow execution. The robustness is attained by suitably shifting activities (introducing or enlarging gaps on resources) based on the probabilities of resource failures, which are estimated according to previous experiences. One of the main shortcomings of the algorithm is the assumption that every resource failure is only temporary, and the time for how long the resource is unavailable in case of its failure is known in advance. The recent state of the work of the authors, aimed at database that supports data mining from logged executions for improving the schedule, is presented in (Dulai and Werner- Stark 2015).

New automated approach for the Aircraft Assignment Problem and the Aircraft Recovering Problem with responsiveness to unforeseen events is proposed in (Sousa et al. 2015). The algorithm employs Ant Colony Optimization and is designed to schedule and reschedule flights dynamically by using a sliding window.

Current Work

Most of the aforementioned literature suggests generally useable technique regardless of the particular scheduling model. Nevertheless, it is not always possible to straightforwardly use the presented methods for a selected class of scheduling problems, hence it requires significant adjustments to make it applicable. In addition, if it is desired to achieve better results in terms of the speed of procedures and the modification distance of a schedule, it is suitable to tailor an algorithm for the particular class of problems.

This motivated our recent work that was presented in (Barták and Vlk 2015). The paper proposes two methods to handle a resource failure occurring on the shop floor during the schedule execution. The first method, *Right Shift Affected*, takes the activities that were to be processed on a broken machine, reallocates them, and then it keeps repairing violated constraints until it gets a feasible schedule. This approach is suitable when it is desired to move as few activities as possible; however, the question whether the algorithm always ends is still open.

The second method, which is aimed at shifting activities by a short time distance regardless of the number of moved activities, is called *STN-Recovery*. The routine deallocates a subset of activities and then it allocates the activities again through integrating techniques from the field of Constraint Programming (Brailsford, Potts, and Smith 1999), namely *Conflict-Directed Backjumping with Backmarking* (Kondrak and Van Beek 1997). Before the allocation process, the search space is suitably pruned based on the values from the original schedule, which is another thing that seems to be neglected in the mentioned literature.

We have been working with the scheduling model taken from the FlowOpt system (Barták et al. 2012), which is based on Nested Temporal Networks with Alternatives (Barták and Čepek 2007) and involves simple temporal constraints. Hence, our main inspiration came from the *Repair-DTP* algorithm proposed in (Skalický 2011). We also employed *Simple Temporal Networks* (Dechter, Meiri, and Pearl 1991) and the *Incremental Full Path Consistency* algorithm (Planken 2008), which incrementally maintains the *All Pairs Shortest Path* property.

Future Plans

Our current plan is to continue working with the FlowOpt model, where, in response to unexpected events, the intention will be not only to modify the allocation of already scheduled activities, but to replace some activities in the original schedule by a set of other (not yet scheduled) activities by searching through alternative branches, i.e., to replan some (ideally the smallest necessary) subset of the schedule.

In further future, the target will be to enhance the model of Nested Temporal Networks with Alternatives by recursion and to suggest algorithms for this model. The recursion will bring the full power of planning, i.e., the possibility to generate activities according to a given target.

Acknowledgments

This research is partially supported by SVV project number 260 224 and by the Czech Science Foundation under the project P103-15-19877S.

References

Abumaizar, R. J., and Svestka, J. A. 1997. Rescheduling job shops under random disruptions. *International Journal of Production Research* 35(7):2065–2082.

Aloulou, M. A.; Portmann, M.-C.; and Vignier, A. 2002. Predictive-reactive scheduling for the single machine problem. In *8th Workshop on Project Management and Scheduling, Valencia*, 3–5.

Asudegi, M., and Haghani, A. 2013. A predictive-reactive dynamic scheduling under projects' resource constraints for construction equipment. In *Proceedings of the International Conference on Operations Research and Enterprise Systems*, 334–337. SciTePress.

Barták, R., and Čepek, O. 2007. Nested temporal networks with alternatives. In *AAAI Workshop on Spatial and Temporal Reasoning, Technical Report WS-07-12, AAAI Press*, 1–8.

Barták, R., and Skalický, T. 2009. A local approach to automated correction of violated precedence and resource constraints in manually altered schedules. In *Proceedings of MISTA 2009: Fourth Multidisciplinary International Scheduling Conference: Theory and Applications*, 507–517.

Barták, R., and Vlk, M. 2015. Reactive recovery from machine breakdown in production scheduling with temporal distance and resource constraints. In *Proceedings of the International Conference on Agents and Artificial Intelligence*, 119–130. SciTePress.

Barták, R.; Jaška, M.; Novák, L.; Rovenský, V.; Skalický, T.; Cully, M.; Sheahan, C.; and Thanh-Tung, D. 2012. Flowopt: Bridging the gap between optimization technology and manufacturing planners. In *Luc De Raedt et al. (Eds.): Proceedings of 20th European Conference on Artificial Intelligence (ECAI 2012)*, 1003–1004. IOS Press.

Barták, R.; Müller, T.; and Rudová, H. 2003. Minimal perturbation problem – a formal view. *Neural Network World* 13(5):501–511.

Brailsford, S. C.; Potts, C. N.; and Smith, B. M. 1999. Constraint satisfaction problems: Algorithms and applications. *European Journal of Operational Research* 119(3):557–581.

Cesta, A.; Oddi, A.; and Smith, S. F. 2000. Iterative flattening: A scalable method for solving multi-capacity scheduling problems. In *AAAI/IAAI*, 742–747.

Cunningham, P., and Smyth, B. 1997. Case-based reasoning in scheduling: reusing solution components. *International Journal of Production Research* 35(11):2947–2962.

Dechter, R.; Meiri, I.; and Pearl, J. 1991. Temporal constraint networks. *Artificial intelligence* 49(1):61–95.

Dulai, T., and Werner-Stark, Á. 2015. A database-oriented workflow scheduler with historical data and resource substitution possibilities. In *Proceedings of the International Conference on Operations Research and Enterprise Systems*, 325–330. SciTePress.

Dulai, T.; Werner-Stark, Á.; and Hangos, K. M. 2013. Immediate event-aware model and algorithm of a general scheduler. *Hungarian Journal of Industry and Chemistry* 41(1):27–34.

Jain, A. S., and Meeran, S. 1998. Job-shop scheduling using neural networks. *International Journal of Production Research* 36(5):1249–1272.

Kondrak, G., and Van Beek, P. 1997. A theoretical evaluation of selected backtracking algorithms. *Artificial Intelligence* 89(1):365–387.

Oddi, A.; Policella, N.; Cesta, A.; and Smith, S. F. 2007. Boosting the performance of iterative flattening search. In *AI* IA 2007: Artificial Intelligence and Human-Oriented Computing, LNCS 4733*. Springer Verlag. 447–458.

Ouelhadj, D., and Petrovic, S. 2009. A survey of dynamic scheduling in manufacturing systems. *Journal of Scheduling* 12(4):417–431.

Planken, L. R. 2008. *New algorithms for the simple temporal problem*. Ph.D. Dissertation, TU Delft, Delft University of Technology.

Raheja, A. S., and Subramaniam, V. 2002. Reactive recovery of job shop schedules – a review. *International Journal of Advanced Manufacturing Technology* 19:756–763.

Ramkumar, R.; Tamilarasi, A.; and Devi, T. 2011. Multi criteria job shop schedule using fuzzy logic control for multiple machines multiple jobs. *International Journal of Computer Theory and Engineering* 3(2):282–286.

Skalický, T. 2011. Interactive scheduling and visualisation. Master's thesis, Charles University in Prague.

Smith, S. F. 1995. Reactive scheduling systems. In *D. Brown and W. Scherer (eds.), Intelligent scheduling systems*, 155–192. Springer US.

Sousa, H.; Teixeira, R.; Lopes, H. C.; and Oliveira, E. 2015. Airline disruption management - dynamic aircraft scheduling with ant colony optimization. In *Proceedings of the International Conference on Agents and Artificial Intelligence*, 398–405. SciTePress.

Vieira, G.; Herrmann, J.; and Lin, E. 2003. Rescheduling manufacturing systems: a framework of strategies, policies, and methods. *Journal of Scheduling* 6:39–62.

Zhang, L.; Wong, T.; Zhang, S.; and Wan, S. 2011. A multi-agent system architecture for integrated process planning and scheduling with meta-heuristics. In *Proceedings of the 41st International Conference on Computers & Industrial Engineering*.