REAL-LIFE SCHEDULING USING CONSTRAINT PROGRAMMING AND SIMULATION

András Kovács[*], József Váncza,
László Monostori, Botond Kádár, András Pfeiffer

Computer and Automation Research Institute, Hungarian Academy of Sciences
Kende u. 13-17,H-1111 Budapest, Hungary

[*]Faculty of Electrical Engineering and Informatics,
Budapest University of Technology and Economics, Budapest, Hungary

Abstract: The paper presents an integrated production planner and job shop scheduler system with flexible modeling capabilities and powerful, scalable solution methods. The system generates close-to-optimal production and capacity plans on the medium term, and detailed production schedules on the short-term. However, the constraint-based, deterministic scheduling model can hardly account for all the uncertainties on the shop floor. Hence, we included such factors into a discrete-event simulation model that is applied to evaluate the robustness of schedules in face of various uncertainties.

Keywords: production planning, constraint-based scheduling, simulation

1. INTRODUCTION

In this paper we suggest a novel approach to modeling and solving production planning and scheduling (PPS) problems. Focus is set to *make-to-order* production environments. We depart from the fact that in the era of supply chain management, virtual enterprises and production networks, the traditional organizational boundaries are getting dissolved. Decisions on the use of resources should concern both internal and external capacities; the internal flow of materials should be synchronized with the incoming and outgoing flows (Váncza, Márkus, 2000; Wiendahl, Lutz, 2002). Medium-term production planning and short-term scheduling are strongly coupled, since production planning sets the goals and constraints for scheduling. On the other hand, scheduling is responsible for keeping due dates and the efficient use of production resources. There is no scheduling strategy that could improve much on an inadequate plan, whereas bad scheduling strategy that wastes resources may inhibit the fulfilment of a good plan. All this makes PPS problem extremely *complex* and hard to solve. Conversely, the complex situations call for efficient, robust *decision support* methods. There is a need of intuitive and flexible models *and* fast, reliable solution techniques that scale-up well also to large, real-life problem instances.

Production plans and schedules, let they be generated by the most sophisticated methods make not much sense if they cannot be executed. However, assumptions (e.g., concerning resource availability, production technology) taken by planning time are often violated at execution time. The closer we are to the realization of plans and schedules, the higher is the chance of unexpected events that can render plans and schedules inadequate. That is why practical scheduling is driven by *uncertainty*, and the methods applied in dynamic job shops rarely utilize theoretical results (McKay, Wiers, 1999).

We do hope, however, that the claim of theoretical modelling and analysis, the quest for the best possible solutions can go hand in hand with practical relevance. To handle the complexity of the PPS problem, we apply *aggregation*. For managing uncertainties, we suggest the application of both

proactive and *reactive* methods. Our goal is to generate robust schedules that resist shop floor uncertainties as far as possible. Contrary to most proactive methods of scheduling (Pisch et al., 2002), we do not build buffers (extra slacks) into the solution, but take a selectionist approach and apply simulation to find the best solution from among a set of alternative candidates.

2. ARCHITECTURE OF THE SYSTEM

After studying the problem domain and having conducted basic research and exploratory experiments in the field of production planning and scheduling, we defined a multi-tiered system structure (see Fig. 1.):

- The solution of medium-term, integrated capacity and production planning problem by integer-linear programming.
- The solution of short-term, detailed finite scheduling problem by constraint programming.
- The evaluation and analysis of detailed, short-term schedules by discrete-event simulation.

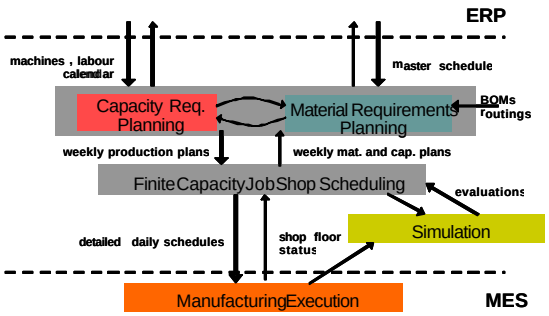

Fig. 1. Main system components: medium-term capacity- and production planner, short-term scheduler and the simulator.

Simulation captures such relevant aspects of the PPS problem that cannot be represented in a deterministic, constraint-based optimisation model. The most important issues are uncertain resource availability, uncertain processing times, uncertain raw material quality, and insertion of conditional operations into the technological routings.

An important practical requirement is that the system components should be able to work with the data stored in existing – so-called legacy – production information systems. In this respect, our proposed architecture connects the Enterprise Resources Planning (ERP) and the Manufacturing Execution (MES) systems of a factory.

3. MEDIUM-TERM CAPACITY AND PRODUCTION PLANNER

Our medium-term planner integrates two planning functions: Material Requirements Planning (MRP) and Capacity Requirements Planning (CRP). While MRP produces detailed, time-phased material plans by using engineering information about how different items are built up (BOM) and produced (routings), CRP matches available capacity to the requirements. It determines the amount of capacities per time units required for producing all elements of the material plan. Decisions of CRP concern overtime, labor hiring or layoff, subcontracting, etc. The integrated problem can be stated as follows:

- given (1) the orders to be completed within the horizon (2) the BOMs of products, (3) routings that specify the sequence, processing time and the resource requirements of operations, and (4) the available resource (machine and worker) capacities;
- determine for each unit of the planning horizon (1) the operations that should be performed that time, and (2) the eventual extra capacities needed.

The planning horizon is a quarter of year, and the time unit is one week. The horizon is rolling, hence part of the order set are open shop orders.

The method we apply for medium-term capacity and production planning is based on a generalized version of *Resource Constrained Project Scheduling Problem* (RCPSP) (Demeulemeester, Herroelen, 2002; Weglarz, 1998). Homogenous groups of machines and workers are considered as *discrete resources*. Machine and worker capacities may vary over time but are known in advance throughout the planning horizon: this *resource calendar* is given by the higher-level capacity planning. Resources may be *internal* or *external*, with are different *prices* for the use of external resources. For more details, see (Márkus et al., 2003).

Each product order is considered as a *project*. Projects have *time windows* given by their earliest start and a latest finish dates. A project consists of activities that are linked by *precedence* constraints. Each *activity* may require several resources and the execution of a given *amount of work*. However, the *intensity* of executing an activity may vary over time; the activity can even be pre-empted. Activities here are *aggregates*: they represent a logical group of atomic manufacturing operations, some of which are executed simultaneously, while others sequentially, and still others independently of each other. This leads to a model in which neither processing times of activities are fixed, nor their intensity is constant over time, however, the amount of work needed to process them is fixed. The solution of a problem instance is a project schedule which specifies what portions of which activities have to be done in each time unit,

such that all precedence and capacity constraints are respected, and the schedule is optimal according to some optimization criterion.

The description of activities is based on static data about the detailed structure and production technology of the products. Hence, the activity model is built on the BOMs and routings. The one-to-one mapping of operations to activities is not convenient, because the problem instances would contain too many activities. Instead, we *merge operations* into activities and – respecting the ordering in the BOMs and routings – define precedence relations between some of the activities. Principles of this transformation are as follows:

- The total resource demand of an activity should not exceed the internal capacity limit per time unit.
- The number of activities in a project should be as small as possible (so that to reduce the problem size).
- The depth of the project's precedence tree should be minimal (so that it contain as many parallel branches as possible).

Since the BOMs and the sequential routings define a tree-shaped production process, the above transformation was formalized as a tree-partitioning problem. Since the last two requirements are in conflict, a trade-off must be found. Pareto optimal solutions are generated by a polynomial-time, dynamic programming method (Kovács, Kis, 2003).

The project scheduling problem can be represented as a mixed integer-linear program, and is solved by a branch-and-cut algorithm proposed by (Kis, 2002). As a basic setting, we consider project deadlines strict and minimize the cost of external resource usage. However, all classical optimization criteria (minimal project duration, maximal tardiness, weighted tardiness, etc.) fit in the suggested framework.

4. SHORT-TERM SCHEDULER

4.1. Schedule generation

The short-term scheduler performs finite capacity scheduling that unifies the resource and temporal aspects at the most detailed level of aggregation. Scheduling should guarantee that all shop orders can be executed in time and that load on resources never exceeds available capacity.

The scheduling horizon of this problem is as long as the time unit of the medium-term planner (i.e., one week). The operation set to be scheduled is given by the aggregate activities that fall into the first week of the medium-term production plan. If an activity extends beyond several weeks, its operations that have to be scheduled during this period are determined proportional to the activity's intensities.

Resources form homogeneous groups, and from each group, a certain number of instances are available. These are so-called *discrete* resources. Resource availability – that may vary shift-by-shift – is given by a calendar. Each operation requires a given combination of resources, each of them either at the beginning of, at the end of, or, (typically) throughout its execution. E.g., a turning operation might require a turning centre and a machinist during the entire length of its processing, and also a quality checker at the end.

The solution of a problem instance is an assignment of starting times to operations such that all precedence and resource capacity constraints are observed.

We obtain such a solution using a *constraint-based scheduling* approach (Baptiste et al., 2001). The model uses variables (e.g., start time of operations), domains of variables, constraints and an optimization objective. The actual criterion is the minimal makespan of the schedules. We are looking for those values of all the variables (in their corresponding domains) that satisfy all the constraints and are the best according to the given criteria.

This technique is considered as the most efficient approach to solve detailed scheduling problems. Constraint-based scheduling enables an efficient combination of general purpose reasoning and search techniques as well as special heuristics. However, scheduling discrete resources is an especially hard problem, which has only a few viable solution techniques for real-life problems (Baptiste, LePape, 2000). Our solution method generates a series of improving solutions, with a more and more narrowing gap between the lower and upper bound of the schedules. The algorithm can be stopped at any time.

4.2. Schedule execution

In a realistic production environment, the rapidly changing conditions by necessity preclude the execution of any rigid schedule, such as one e.g., with fixed operation starting times. Dynamic, in part unpredictable environments require *reactive* scheduling techniques and a mixed-initiative support for schedule execution.

Most reactive scheduling systems first solve the scheduling problem addressing the global optimum. This solution is then considered as a *predictive schedule* and is adapted to the actual execution environment by *iterative repair techniques* and periodical *rescheduling*. While repair techniques maintain a reasonable schedule by applying simple and fast modifications on the predictive schedule, causing minimal perturbation to it as a reaction to smaller changes in the environment, greater breakdowns may require global rescheduling to keep in a small range of the theoretical optimum.

Hence, we *relax* the solution that has been generated by the constraint-based scheduler. We remove the fixed start times of operations and keep only the *sequence* of the operations on the various resources. Hence, based on the start times, *queues* are formed for each homogeneous resource group, let it be a machine or worker. Note that an operation stands in as many queues as many resources it needs.

By default, an operation can be processed if it is in the front of all of its queues. However, since there are not only unary, but also discrete resources, we can apply a more liberal execution policy while keeping all the time the consistency of the overall job-shop schedule. Accordingly, an operation can be processed any time if it does not cause lateness of the operations that are before it in the its respective queues. This so-called *overtaking* rule is illustrated at Fig. 2.

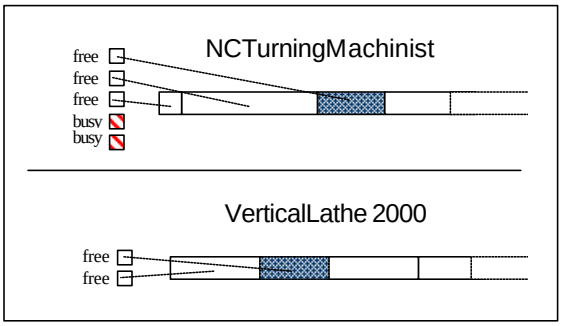

Fig. 2. Application of a simple overtaking rule.

The selected operation is queuing for two resources: a machine (VerticalLathe2000) and a worker (NCTurningMachinist), where the overall capacity of the machine and worker resources are 2 and 5, respectively. Suppose two workers are bound to make other operations. Although in the queues there are some operations before the selected one, there are enough free resources to execute it without causing the eventual delay of the others in fronts of the queues. Hence, even if the first operations in the queues cannot be executed for some reasons, the selected one can.

Generally, resources are reserved for all the operations that cannot yet be executed but precede the selected operation. If free resources remain, the operation can jump forward and be executed. Of course, the overtaking rule must not violate the explicit precedence relations between the operations. Note that in case of unary resources an operation can be processed only if its is the first in all of its queues.

However, shop foreman have the final word in deciding on schedule execution. In order to prevent blocking operations, the foreman can postpone or remove some operations from the queues. Such decisions are supported by the MES and scheduler systems.

Upon unexpected events, such as a machine breakdown, or when discrepancy from the predictive schedule exceeds a certain threshold, the foreman can ask also for a global rescheduling.

5. THE SIMULATOR

In the followings a multi-purpose application of simulation is depicted. On the one hand, the simulation modelling follows the traditional methodology and results in a complex decision support tool. On the other, the developed simulator is utilised as a component of a higher level system taking the role of the real production system (see Fig. 1) and acting as a *"quasi-emulator"*. Quasi-emulation means that, contrary to traditional simulation models which integrate the control logic of a real system – generally utilising dispatching rules – the model and the control of the physical system are separated. Accordingly, emulation is a replication of a real system without any control function.

In our case, the simulator will playback the schedules calculated by the constraint-based scheduler. Control will be performed according to the schedule execution strategy describe above (see Sect. 4.2).

However, the deterministic constraint-based factory model can hardly account for all the *uncertain events* especially for those that may happen on the shop floor. Hence, we include such uncertain factors into our *simulation* model which will be used to evaluate the results of the constraint-based scheduler in face of uncertainties.

5.1. Uncertainties

The basic types of uncertainties that are modelled in the simulation model are as follows:

- *Start time uncertainty*: Delivery of purchased items (raw materials) is not always reliable. Hence, it is not sure whether some operations can really be started at the planned earliest start dates.
- *Downtimes*: Due to failures and/or unexpected absence machines and/or workers may not be available as planned.
- *Processing time*: The actual processing time of some operations may depend on the proficiency and skill of the worker. Processing times can be shorter or longer than planned.
- *Re-work and adjustment*: The execution of specific operations depend on the result of quality check operations. Based on the result of the check, they may be repeated or some adjustment operations are performed.
- *Quality uncertainty*: Whether some adjustment operations should be performed or not may depend also on the quality of incoming material. Due to a major quality problem, even the progress of a product may be blocked.

5.2. Evaluation of schedules

Our intention to connect our scheduler to a discrete event simulator was dual: On the one hand, it serves as a benchmarking system to evaluate the schedules on a richer model, one also covering the non-determinisms of the real-life production environment. Furthermore, we expected that the statistical analysis of schedules could help improving their behaviour under such circumstances.

Despite the latter topic has already been investigated by several researchers, (see e.g. Kempf et al., 2000) rather initiatives than solutions are known to it. This is due to the many — and hard to understand — interdependencies of atomic decisions in the scheduling problem. Moreover, the speed of solving either the scheduling or the simulation problem does not allow performing a larger number of iterative improvement cycles. Possibilities here include a balanced loading of resources throughout several weeks, or generating short-term schedules where the most uncertain operations are not on the critical path.

Hence, in the current implementation, we confined ourselves to generating a contingent of schedules by introducing a small perturbance into the search strategy of the constraint-based scheduler, and provide statistical data about each schedule instance. This data is measured over several runs of the discrete event scheduler, and includes

- the average external tardiness (average amount of work that could not be executed within its due date);
- the average internal tardiness (average amount of work that could not be completed until its internal deadline, posted by the medium-term scheduler);
- schedule stability (average number of re-schedulings performed).

The most robust schedule is then selected by a human decision maker.

5.3. Functions of the simulator

All in all, the main functions of the discrete-event simulator are as follows:

- It supports the auditing of factory databases (like routing, machines, worker etc. tables).
- It evaluates the robustness of daily schedules against the above-mentioned uncertainties.
- Simulation helps in visualizing and verifying the results of the PPS system.
- Simulation supports the systematic test the pilot PPS system.
- The simulator offers a benchmark platform for the calculated schedules.

5.4. Implementation

The simulator model has been implemented by using the eM-Plant object-oriented discrete-event simulation tool. eM-Plant is a standard software for integrated, graphic and object oriented modelling, simulation and animation of systems and business processes.

The simulator has at the moment off-line, file exchange-based interfaces to the ERP database and the constraint-based scheduler. Besides, on-line database interfaces are also available to ease the set-up of automatic simulation runs.

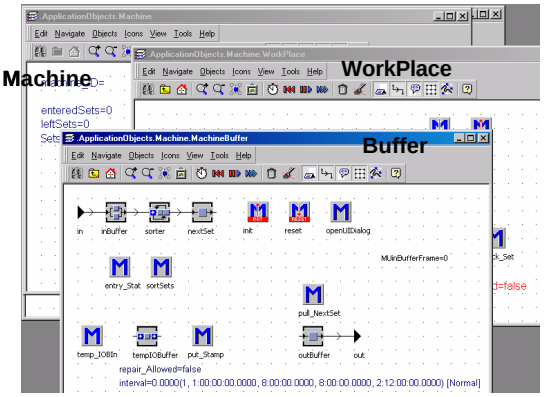

Fig. 3. The Machine application object in the template library of the simulator

6. CASE STUDY

We elaborated a case study in a factory that produces mechanical products by using machining and welding resources, assembly and inspection stations and some highly specialized machines. Though products of different families differ also in their production technology, the problem cannot be decomposed: the sets of resources used for producing different families are not disjoint. Production is performed in a make-to-order manner, where deadline observance is an absolute must, even for unpredicted orders. Since quality assurance is a key issue, tests may result in extra adjustment operations. Hence, production technology contains a factor of uncertainty in itself.

The planning and scheduling method was validated and tested with the above real-life data. First, projects were generated from existing routing tables and BOMs, then, using the resource calendars, the planning problem was solved on a 15 weeks horizon, with a time unit of one week. Then, the first week's production plan was passed to the constraint-based finite job-shop scheduler that worked with a 10 min. time unit.

Typical problem instances had up to several hundred orders (projects). The projects consisted at most ten activities that were merged from ten to hundreds of

operations. Altogether resources considered were in the order of 10^2. Generating the project model with the tree-partitioning algorithm was a matter of seconds, and the solution of the project model took never more than one minute. Constraint-based scheduling had to solve problem instances for in the order of 10^2 discrete resources and 10^3 operations. It produced an improving series of solution, with the first one within seconds, and really good ones (approaching 5-10% to the optimum) within minutes.

7. CONCLUSIONS

Above we have presented a PPS system whose components – a medium-term aggregate capacity and production planner, a short-term job shop scheduler, and a discrete-event simulator – work on more and more detailed models of a given production environment. While the basic models of the planner and scheduler are deterministic, the simulator can capture non-deterministic events – especially those that may happen on the shop floor.

Now, having all the main system components operating efficiently, we can make detailed experiments. One key issue is how to set the parameters of the simulation model so that we get statistically reliable evaluations of the schedules. Secondly, since the activity model aggregates operations, there is no guarantee that a solution of the project model leads to an executable schedule on the finer level. Finally, the quality of a realized schedule depends both on the predictive schedule and the repair techniques applied on it during its execution. One question, which is still open to us, is how to make a smart compromise between the optimality and flexibility of the schedules. Factors considered here are the even distribution of load throughout the scheduling horizon, the assurance that the minimal number of tasks will have to be adjourned beyond the horizon, and the stability of the schedules.

ACKNOWLEDGEMENTS

This work has been supported by the "Digital Factory" NRDP grant No. 2/040/2001. The authors are particularly indebted to F. Erdélyi, Z. Mihályi and A. Szántai for their help.

REFERENCES

Baptiste, Ph., Le Pape, C. (2000). Constraint Propagation and Decomposition Techniques for Highly Disjunctive and Highly Cumulative Project Scheduling Problems. *Constraints*, **5**:1-21.

Baptiste, Ph., Le Pape, C., Nuijten, W. (2001). *Constraint-Based Scheduling*. Kluwer Academic Publishers.

Demeulemeester, E.L., Herroelen, W.S., (2002). *Project Scheduling: A Research Handbook*. Kluwer Academic Publishers.

Kempf, K., Uzsoy, R., Smith, S., Gary, K. (2000). Evaluation and Comparison of Production Schedules. *Computers in Industry*, **42**:203-220.

Kis, T., (2002). A Branch-and-Cut Algorithm for Scheduling Projects with Variable-Intensity Activities. Submitted to *Mathematical Programming*.

Kovács, A., Kis, T. (2003). Partitioning of Trees with Respect to New Criteria. Submitted to *Information Processing Letters*.

Márkus, A., Váncza, J., Kis, T., Kovács, A. (2003). Project Scheduling Approach to Production Planning. *Annals of the CIRP*, **52**(1).

McKay, K.N., Wiers, V.C.S. (1999). Unifying the Theory and Practice of Production Scheduling. *Journal of Manufacturing Systems*, **18**(4): 241-255.

Nuijten, W., Le Pape, C. (1998). Constraint-Based Scheduling with ILOG Scheduler. *Journal of Heuristics*, **3**:271-286.

Pisch, M.T., Loch, Ch.H., De Meyer, A. (2002). On Uncertainty, Ambiguity, and Complexity in Project Management. *Management Science*, **48**(8):1008-1023.

Váncza, J., Márkus, A. (2000). An Agent Model for Incentive-Based Production Scheduling. *Computers in Industry*, **43**:173-187.

Weglarz, J. (ed.) (1998). *Project Scheduling. Recent Models, Algorithms and Applications*, Kluwer Academic Publishers.

Wiendahl, H.-P., Lutz, S., (2002). Production in Networks. *Annals of the CIRP*, **51**(2):1-14.