Constraint Handling in Manufacturing Planning Systems

Heimo H. Adelsberger, Klaus-Peter A. Keilmann
Universität GH Essen, Wirtschaftsinformatik der Produktionsunternehmen
Altendorfer Str. 97-101, D–45143 Essen
e-mail: {heimo, keilmann}@wi-inf.uni-essen.de

Abstract

This paper deals with constraint handling in manufacturing. It starts with a basic definition of the constraint satisfaction problem, followed by a description of consistency enforcing and constraint guided search. Constraint handling in scheduling problems is the main topic of the paper also presenting different types of schedule construction methods and explaining the use of temporal constraints. Special attention is given to the task of mid-term production planning. First, production planning in ordinary MRP II-based systems is introduced, followed by the presentation of the master production scheduling leitstand (MPSL), a research and development activity of a "Forschungsschwerpunkt" at the University GH Essen.

1 Introduction

Constraint handling as a new concept of problem solving has emerged in the last decades. First investigations had been done in the area of picture processing in the late seventies [14] in order to make search more "intelligent". Constraints are used in many areas, e.g., diagnosis [20], examination planning [7], mission planning in astronautics [21], or even medicine [18].

In manufacturing, many applications in the area of graphics, design, engineering, and planning [10] make use of constraint solving techniques. Many of these problems involve search. A good example is Computer Aided Planning (CAP). It is purposed to define the technically necessary sequence of the various operations in manufacturing a part, considering machines and tools available [12]. This task, however, is only performed once for every part or product, therefore developing efficient techniques for this job is of less interest from the planning point of view. Production planning or scheduling as the task to lay down when to perform which activity on which resource is another typical constrained search problem. It confronts the constraint handling community with real world problems instead of "toy" problems like *'Who owns the zebra?'* or the *'n-queens'* problem [17].

Production planning is reflecting two different time horizons:

- short-term planning
- mid-term planning

Short-term production planning is regarded when talking about scheduling. Assignments of operations to machines have to be made. Operations are given in terms of ready time, duration (processing time), and due date. Resources are limited to be unary which means that only one operation can be assigned to a resource at a time [24].

Mid-term production planning in standard MRP II-based manufacturing planning and control systems consists of master production scheduling, material requirements planning, and capacity planning. In this case, multiple unit resources and more pliable data have to be considered.

In Section 2, a basic definition of the constraint satisfaction problem is given followed by a description of consistency enforcing and constraint guided search. Constraint handling in scheduling areas is dealt with in Section 3, focusing on different types of schedule construction and the use of temporal constraints. Topic of Section 4 is the task of mid-term production planning. First, production planning in ordinary MRP II-based systems is introduced, followed by the concept of the Master Production Scheduling Leitstand (MPSL). The paper ends with a discussion of further research possibilities.

2 Constraints and search

Constraints describe relationships between values of variables which are defined over (finite) domains, restricting the possible values which can be assigned to the variables participating in the constraint simultaneously [22]. The set of all given constraints for a certain problem is called a constraint net. A constraint net can be described by a set of variables $V = \{V_1, \ldots, V_n\}$ each defined over a finite domain $D_i = \{d_{i1}, \ldots, d_{ip_i}\}$, and a set of constraints C = $\{C_1, \ldots, C_m\}$. The arity of a constraint is defined by the number of variables involved—e.g., a binary constraint defines a relationship between two variables.

Finding a solution of a constraint net—the constraint satisfaction problem—is approached by selecting an appropriate assignment of a value $v_i \in D_i$ for each variable V_i. The simplest way is to use some "generate and test" algorithm which means to assign values to the variables and to test if all constraints hold. A more efficient way is to assign a value v_i to one variable V_i and then to assign values to all variables connected with V_i by some constraint so that this constraint holds. If it is not possible to assign valid values to all variables, backtracking occurs. Many algorithms have been explored to make this backtracking process more efficient (see [16] for a summery of backtracking and backjumping algorithms). Constraint solvers have been successfully implemented for boolean, linear, rational, and list domains.

2.1 Consistency enforcing and propagation

The task of finding a consistent variable assignment is supported by using propagation of information concerning valid variable domains in the constraint net. Given two variables V_1, V_2 with domains $D_1 = D_2 = \{1, 2, 3\}$ and given the constraint $C : V_1 \leq V_2 + 2$, it follows that $V_1 = 1$ and $V_2 = 3$. Removing values from the domain which are inconsistent in respect to just a single constraint is called consistency enforcing, resulting in so-called local consistency (in the case of binary constraints, the term arc consistency is also used). Changes of the domains are propagated, resulting in new consistency checks, until no more values can be removed from any domain in the constraint net. Achieving local consistency cuts down the search space but does not necessarily lead to a solution of the constraint satisfaction problem as can be easily seen by the following example: Given the variables V_1, V_2, V_3 with domains $D_1 = D_2 = D_3 = \{1, 2\}$ and given the constraints $C_1 : V_1 \neq V_2, C_2 : V_1 \neq V_3$, and $C_3 : V_2 \neq V_3$, consistency enforcing does not remove any values from the given domain. There exists a variable assignment for each constraint C_1, C_2, C_3, ensuring arc consistency. Nonetheless, it is evident that no solution for the whole constraint net exists.

Searching for a solution is equivalent to trying to achieve global consistency which means that there exists a set of variable assignments in such a way that all constraints hold simultaneously. This can be accomplished by labeling a variable with some value from its domain (which might turn out to be the wrong one). This new information is then again propagated through the constraint net—hopefully resulting in further removals of values in other domains. At this point backtracking could occur in the case that the propagation of a value assignment leads to inconsistency. A solution is not found until every variable can be labeled with a valid value. At that moment it becomes clear that the constraint solving process is determined by the chosen labeling strategy—which in return reflects the used search strategy.

Constraint reasoning as described above is determined by different aspects:

- The backtracking strategy of the reasoning engine: several backtracking algorithms have been developed reflecting different requirements and problem domain characteristics (see [16]).
- The search (labeling) strategy: different methods of variable ordering and value ordering have been designed. Variable ordering deals with the selection of the next variable for which a value is going to be assigned. Common heuristics are choosing the variable with the smallest domain (first fail principle) or selecting the most constrained variable (the variable which occurs in most constraints). Which value to choose next is a more difficult task. One heuristic is to select the value which maximizes the number of consistent values of the remaining variables.

2.2 Search

Search and constraints are closely related. In the context of constraints, search can be seen as the process of finding some leave in a search tree which represents a feasible solution whereby constraints allow some branches which do not contain solutions to be pruned off. In order to find an optimal solution, a measure (typically a cost function) has to be defined and the costs for all possible solutions have to be computed. Algorithms using this approach are called enumeration algorithms. To solve complex problems, to-

tal enumeration is impracticable since there are far too many nodes in the search tree.

Enumeration algorithms. Branch & Bound—probably the best known and most widely used search algorithm—fits well for constraint solving because of its incremental character. Constraints can be used to limit the search space by cutting branches due to consistency enforcing. Later on, the search can be guided via variable ordering and value selection.

Approximative algorithms. On the other hand, optimal solutions are not needed in most real life situations. Near-optimal solutions are sufficient in most cases: either the difference between the optimal and the near-optimal solution is marginal and therefore irrelevant for practical reasons or the cost function used cannot be determined thoroughly so that—given the circumstances—the theoretically optimal solution may not be the best solution of the given real problem. Finding good (near-optimal) solutions with a high probability in reasonable time can be achieved by using approximative algorithms which only explore a part of the solution space [24], thus they are not able to guarantee to find the optimum.

Approximative algorithms—such as simulated-annealing, hill-climbing, and genetic algorithms—work if the associated cost-function is in some way continuous: similar solutions should have nearly similar costs. If a description of a search problem includes constraints, one could consider to assign high costs to nodes in the search space which are violating constraints, thereby avoiding infeasible solutions or invalid variable assignments due to the high costs they produce. Unfortunately, this approach is not practicable since it leads to an irregular cost function, violating a necessary requirement for using approximative algorithms. Constraints, however, could be used to generate good initial starting points (so that such an approximative algorithm can start from a better point than the one found by random generation) and to constraint the operators generating new solutions. Genetic algorithms give a good example of how constraints can be used to guide mutation (e.g., enforcing that some variables take certain values), and recombination (e.g., avoiding that variables take certain values proposed by the cross-over operator—these variables are labeled under constraints instead) [24].

3 Constraint handling and scheduling

Scheduling, defined as sequencing and assigning resources over time, is known as being a NP-hard problem [11]. Following [8], scheduling can be seen as time-based planning where tasks compete for resources which they need to reach their goal.

From the constraint handling point of view, scheduling is of interest when dealing with the general job-shop problem; as for the cases of flow-shop and single machines, priority rules exist for many cases leading to results which can be proven to be optimal with respect to any regular measure (maximal and average flow time, completion time, earliness, tardiness, number of waiting jobs, and idle times)—see [9, 6] for an introduction to classical scheduling.

The general job-shop problem according to [9] is defined as follows: Given a set of machines $M = \{M_1, \ldots, M_m\}$, and a set of jobs $J = \{J_1, \ldots, J_j\}$ which have to be processed by these machines. It is assumed that each job has to be processed on each machine exactly once. Processing a job J_k on machine M_l is called an operation or task, the set of all operations is given by $O = \{o_{11}, \ldots, o_{jm}\}$. The processing time is defined as the time an operation needs to be performed. Set-up and transport times can be included in the processing time (avoiding extra variables) to simplify the model. Every job has its own ready time (the earliest possible time it can be accessed), and a due date (the time when it has to be finished). Constraints in the general job-shop problem occur as technological constraints, time constraints, or capacity limitations. Technological constraints exist between operations of the same job, reflecting the sequence of the operations. The general job-shop problem does not make any restrictions regarding the form of these constraints. Every job may have its own processing order. Jobs have to be performed in a specific amount of time given by ready time, due date, and processing times, thus limiting the potential starting and finishing dates for the operations. Since every job has to be processed by every machine and every machine can only process one operation at a time, resource competition occurs.

Even the general job-shop problem is rather complex, it very often does not reflect the real situation in manufacturing since it uses many simplifications. The standard definition of the job-shop problem ignores a lot of constraints found in real life systems. Examples are stochastic processing times, sequence dependent set-up times, secondary resources like workers, tools, or fixtures, alternative routings, and the like.

Fox [8] identifies five constraint categories, each

consisting of different types of constraints:

- Organizational goals: The performing of an organization has to be evaluated using different measures. These measures, e.g., meeting of due dates (late/tardiness), work-in-progress inventory, or shop stability, can be formulated as constraints. Profit also measures the performance of an organization, in this case, however, one has to be able to assign valid current and future costs to every influencing factor.

- Physical: Physical constraints specify given limitations in functionality. E.g., limitations in size, material which can be handled, and time.

- Causal: Tasks often depend on each other. E.g., drilling has to be finished before painting can start. There are also limitations in the usage of resources. Often tasks have to be assigned to specific resources, tools, and operators.

- Preference: Sometimes schedulers prefer the assigning of a specific task to a specific machine or give some tasks precedence over others.

- Availability: Resources can only be assigned to one task at a time, resulting in these resources not being available to other tasks during processing.

Trying to satisfy a constraint net including all these different types of constraints will in most cases not be successful—depending on the expectation about the quality of the plan: The constraints, although well-defined and individually correct (e.g., high inventory from a sales person's view, low inventory from a inventory manager's point of view), rival with each other. Another example is meeting due dates: this conflicts inherently with the goal (constraint) to achieve low work-in-process inventory. If the conflict cannot be solved, one of the constraints has to be relaxed.

In addition, there are more considerations effecting constraint satisfaction [8]:

- Importance. In manufacturing systems, orders with different priority exist. Single constraints can have different levels of importance, depending on the characteristics of the given orders.

- Interaction. Constraints influence each other due to constraint propagation. This influence can be positive or negative. The knowledge how two or more constraints influence each other is difficult to maintain and to make use of.

- Obligation. Often different values for the same constraint can be distinguished depending on the personal point of view and on time aspects. E.g., for the same order, different due dates are used by manufacturing and marketing. The constraint solver therefore has to determine which data to take.

3.1 Predictive and reactive scheduling

Predictive schedulers create schedules based on information about jobs, task durations, resources, etc. Reactive schedulers take a schedule and modify it when something unexpected happens (e.g., machine breakdown, job modifications). Predictive scheduling is aimed at constructing a (sub)optimal schedule for a rather long period. The assumption is that these schedules will never be executed according to plan because of the uncertainty of the data used and because of the simplified model of the manufacturing system in question.

Reactive schedulers have to react quickly to any change on the shop floor and have to make use of detailed information about the actual manufacturing situation. They also have to limit the changes necessary to repair the schedule and to come to a new schedule, restricting the changes to local interventions in space and time (machine breakdowns in one working area should not have an impact on other areas to avoid nervousness in the production area).

The expression turn-pike scheduling is used to denote the strategy to find the least possible change to the given schedule necessary to get back to the original schedule. Real-time schedulers combine predictive and reactive scheduling, permanently simulating the future development of the manufacturing system and comparing the results of this simulation with the predicative schedule in place, rescheduling whenever an immediate need turns up or an unfavorable development of the manufacturing system can be recognized.

Whereas traditionally most scientific work concentrated on predictive scheduling, in practice reactive and real-time scheduler become more and more important.

3.2 Temporal constraints

As can be recognized from above, reasoning about time is one of the central aspects in scheduling. This is caused by the sequencing constraints and the resource contention between different tasks which need the same machine for execution. Temporal reasoning using constraints deals with constraints over intervals

representing start and end points. Thirteen different relations can be identified between two intervals [22, 5] as shown in Fig. 1.

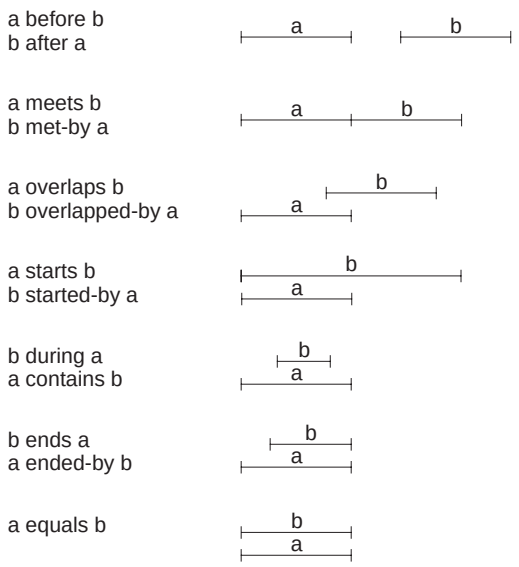

Figure 1: The 13 temporal relations as given in [5]

Constraints over intervals can be propagated and new constraints can therefore be established. Given, e.g., the intervals a, b, c, and d, and constraints C_1 : $a\ after\ b\ after\ c$ and C_2 : $d\ during\ c$. How can the information that d has to be performed after a and before c be derived? (Reflecting a schedule, one can imagine a situation where there has to be some critical heating up (operation o_2) between operation o_1 and o_3 and it is known that if there is more than two minutes break of power, some emergency activity has to be started.) Due to constraint reasoning, it is possible to establish a constraint C_3 : $a\ before\ d\ before\ c$ as shown in Fig. 2.

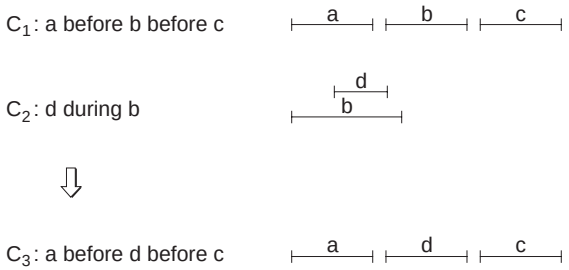

Figure 2: Deducing new temporal constraints

In [5], an algorithm is given for achieving arc-consistency in interval constraint networks. A constraint between two intervals is represented as a non-empty subset of the relations given above. The constraint C_1 : a $\{contains, overlaps\}$ b means that the interval a starts before b and ends before or after b ends. A constraint therefore simply connects the possible relations by "or". Reasoning over time is the process of establishing new constraints over yet unconstrained intervals. Given the intervals a, b, c and the constraints C_1 : a $\{starts\}$ b, C_2 : b $\{overlapped\_by\}$ c, it can be deduced that c starts before b starts and c ends either during a is still running, ends or has already ended resulting in the constraint $C3$: a $\{overlapped\_by, during, ends\}$ c. A transitivity table for all thirteen relations and an an interval propagation algorithm is given in [22] based on [5].

4 Constraints and manufacturing planning

In the previous section, the basics of constraint handling have been stipulated. Now, the application of constraint handling for manufacturing planning is discussed. The section starts with a short description of MRP II, followed by a presentation of a planning tool for master production scheduling, the Master Production Scheduling Leitstand (MPSL) [3]—developed at the University of Essen—which takes extensive advantage of constraint handling.

Current manufacturing planning and control systems (MPC) follow the manufacturing resource planning (MRP II) approach. The term MPC is more general than MRP [23] and includes strategies and techniques like, e.g., Just-in-Time or KANBAN which are, however, not of interest in this context. The basic components involved in planning and the flow of information in such systems are shown in Fig. 3.

The different planning tasks are processed straight forward (there are no **formal** feed-back loops; if, however, the result of the planning is unacceptable, previous planning decisions have to be corrected). After some rough-cut capacity planning based on highly aggregated data, master production scheduling (MPS) takes place. Master production scheduling results in a medium-term production plan focused on final products. This plan determines the production quantities and due dates of all final products. The plan is the basis for material requirements planning which translates the gross need (product requirements) into net need (part requirements), with regard to inventory and scheduled shipments. The result is the net requirement of parts per period, taking several aspects like, e.g., lot sizes or lead times into account. Capac-

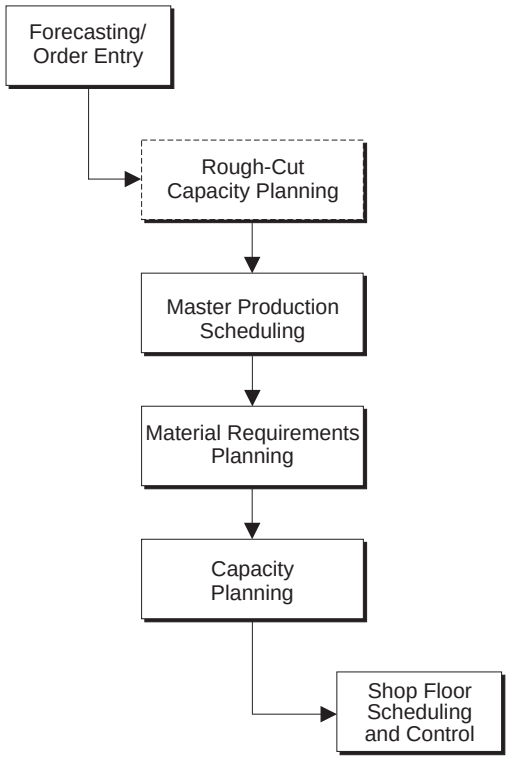

Figure 3: Planning in MRP II-based systems

ity planning is done in the end. Capacity planning uses techniques like forward and backward scheduling to compute earliest start and latest finishing dates for every manufacturing order. It also smoothers capacity loads over time (loads in periods already working to capacity are moved to periods where full capacity is not used). The resulting manufacturing orders are then released to shop-floor scheduling which is responsible for the short-term scheduling of the operations as mentioned in Section 3. Shop-floor scheduling depends on reliable key data provided by the previous planning stages since the data provided (like earliest start and latest finishing dates for operations) restrict the flexibility at this planning level considerably.

Master production scheduling nowadays is usually done every week with the planning period ranging from six weeks to eighteen months; the time raster is normally one or two weeks, sometimes a month [23]. In contrast to market forecasts, the MPS does not represent expected demands. It rather focuses on production quantities. Since the MPS has to consider capacity limitations, trying at the same time to make use of available capacities as efficiently as possible, this might result in items being produced in advance (make to stock) or in potential orders being refused. MPS takes the end-product requirements per week (fixed orders and the forecast for orders) as input for calculating a production plan based on end-products per week. It takes into account the current stock of semi-finished and end-products on hand (some MPS systems allow to decompose end-products to component level). Rough-cut capacity planning considers the known bottlenecks in the production process. For every end-product or component, some average consumption of such a bottleneck resource is assessed. Currently MPS, however, ignores that bottlenecks may change ("moving bottlenecks") subject to changes in the production program. A further disadvantage is the extensive use of lead times which only superficially reflect the complex dependencies on the shop-floor. As a result, MRP II-based medium-term manufacturing planning suffers from many problems caused by the lack of feedback and the lack of sufficient coupling and coordination among the different planning tasks:

- capacity restrictions are not fully taken into account
- the planning process is strictly deterministic
- the use of fixed lead times instead of a correct model of the production process

In the case of mass production, manufacturing most of the time is organized in accordance with the production flow, allowing to concentrate on the bottleneck resources. In unit production, project planning methods are used for quoting realistic offers. In the case of very complex planning situations, e.g., due to a great number of products variants, to a high demand for flexibility in production, or to great interdependence between manufacturing and other enterprise areas, medium-term planning using MRP II-principles is hardly used, as the ordinary manufacturing planning and control systems are not able to provide adequate methods to support this task [3].

4.1 The master production scheduling leitstand (MPSL)

To overcome the problems identified in traditional master production scheduling systems for variant and and small-size serial producers, the University GH Essen [3] is developing a new kind of production planning system. Centered around the master production scheduling function, it tries to transfer the ideas which made the so-called leitstand systems so successful to short-term production planning.

According to [2], a leitstand is a computer-aided graphical decision support system for interactive production scheduling and monitoring. Shop-floor leitstands have to deal with relatively short planning horizons (1–2 weeks) with a fine granularity at the resource level. MPS is based on a much longer planning horizon, and lower granularity at resource group or shop level. This allows the modification of several parameters while creating a plan, e.g., shift models and inventory levels can be altered. This kind of parameters are considered to be fixed in short-term production planning. Fig. 4 shows the structure of manufacturing planning using a MPSL.

The leitstand approach explicitly considers the finite capacity situation. Integration of capacity planning into MPS is very important, as traditional planning on the basis of average capacities and fixed lead-times often shows unrealistic results. Additionally, bottlenecks in manufacturing are caused by the production program which is not treated by the traditional planning approach.

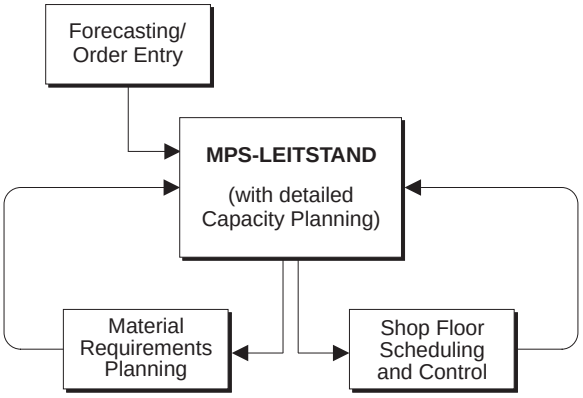

Figure 4: Planning in the context of the MPSL

Constraint handling in MPSL is done in principle like in short-term manufacturing planning, allowing, however, to take many more degrees of freedom into consideration. The project takes fully advantage of the results of the ESPRIT Project 5161: Design, Development, and Implementation of a Knowledge-Based Leitstand (KBL)[1] where utmost attention was given to constraint handling. The aim of the KBL project was to develop a prototype of a Knowledge-Based Leitstand. A major aspect was to support in a flexible and powerful way the formulation and integration of general and company specific constraints to allow a maximum of freedom in the formulation of the constraints concerning the manufacturing situation.

The experience acquired from constraint handling in short-term production planning in the KBL project was used for mid-term production planning in the MPSL. First of all there still exist technological constraints which are now reflecting the sequencing of machine groups despite of single machines. The handling of groups of machines leads to an additional level of complexity: dealing with multiple units of one resource. This means that different tasks could be assigned to the same resource at the same time. Secondly we have to deal with a greater number of constraints than in short-term manufacturing planning (e.g., manufacturing orders have to be generated based on given forecasts, orders, and manufacturing situations, compared to short-term production scheduling where the manufacturing orders are already released; i.e., they have fixed ready times and due dates, and the lot sizes are already determined). Most of the constraints identified by [8] are ignored in short-term planning—this is inconceivable in mid-term production planning, because the production plan reflects the overall goals of the company, not the goals of the production managers. (Theoretically these should be the same, but often manufacturing is interested in local aspects like high utilization of machines or low inventories. Sometimes, these goals can be in conflict with global goals like to maximize profit.)

Capacity planning. As lead times should not be used in the MPSL-approach, load profiles of all resources have to be kept. Capacity requirements of components and sub-parts will be calculated on the basis of bill-of-materials and routing plans. If capacity limitations have been violated, the master production schedule has to be adapted by means of shift modifications, quantity modifications (use more machines), or intensity modifications (increase of production speed). The use of multiple unit resources instead of only unary ones is of big importance from a constraint handling point of view, since it requires the use of cumulative constraints.

Even in the case of multiple unit resources, capacity limitations exist. E.g., the capacity of resource group R is given as the sum of the maximum capacity of every resource in R. Cumulative constraints over finite domains were first introduced in [4] to deal with scheduling and placement problems. Cumulative constraints are defined over a set of operations $O = \{O_1, \ldots, O_n\}$, defining a relation between the set of starting times $S = \{S_1, \ldots, S_n\}$, the set of processing times $P = \{P_1, \ldots, P_n\}$, and the set of machines $M = \{M_1, \ldots, M_n\}$ with a given capacity L. The constraint holds if and only if the required capacity is

lower than L during the interval given by the minimal starting time given by $min_{1 \leq i \leq n}\{S_i\}$ and the maximum ending time $max_{1 \leq i \leq n}\{S_i + P_i\}$ (see [4] and [15] for a more detailed description).

Soft constraints and search. Both, technological (hard) and non-technological (soft) constraints (in [8], the terms 'required' and 'preferential' constraints are used) have to be taken into consideration during planning. In principle, soft constraints cannot be used to limit the domains of the involved variables—soft constraints can be relaxed in case there is no solution which satisfies these constraints. In manufacturing—as well as in other areas—solutions which cause too much violation to soft constraints are not acceptable. E.g., if some order is finished excessively late, i.e., a soft constraint is violated since delivery cannot be effected before due-date, the price will have to be deducted considerably, the delivery will be rejected, or the customer might claim damages for non-fulfillment.

Two maneuvers allow soft constraints to be treated like hard ones:

- Restricting the domain of each variable to acceptable values. The acceptance is influenced by the constraint, the application, and the environment given.
- Defining a cost function over the by this time finite domain.

The concept of using cost-functions is quite similar to the use of utility functions used in [19]. Utility functions in form of probability or density functions reflect the probability that the chosen value assigned to some variable results in an acceptable schedule. Sadeh and Fox [19] restrict their preference propagation to unary resources in order to avoid the problem that these utility functions depend on resource utilization.

This is not possible in the context of the MPSL. MPSL uses cost functions available in real manufacturing to achieve some measure to value the goodness. E.g., labor costs depend on skill and the time of occurrence (work at night is more expensive than during normal working hours), costs of plant and machinery depend on intensity and tools needed. As such cost functions do not exist for variables describing operations, like starting and finishing time, or duration, some secondary measure has to be used. This is achieved on the one hand by assigning material costs needed to perform the operation and on the other hand by taking time into consideration due to discounting the net in- and outflows, thus transferring the concept of shareholder value analysis to production.

The modified soft constraints can be used in consistency enforcing and constraint propagation. The cost functions allow to perform a value and variable ordering to direct search in the same way as in short-term planning. Dealing with operations, variable ordering is accomplished by some "expensive operations first" rule. The underlying idea is that material needed for an operation has to be ordered at the latest convenience and that products have to be manufactured near their due date, ensuring low work-in-process inventory. This is also reflected during value selection. Processing should be done as late as possible. From the resource point of view, the rule "expensive resources first" is used for variable ordering. Expensive resources are defined based on their specific cost of usage (specialized resources). Value ordering is done by "cheapest time frame possible". The assigned cost function reflects the current utilization of the resource. Bottleneck resources which have to be run at their maximum intensity for some period become expensive during this time. Choosing the cheapest time frame possible especially reflects the aim of balancing the load if bottlenecks occur. As can be seen, search in MPSL makes extended use of heuristics.

5 Conclusion and further research

Constraint handling can be used in manufacturing scheduling, although many of the constraints turn out to be preference constraints like due dates or high resource utilization (for bottleneck machines). Short-term planning concentrates on assignment of operations to resources. As in the general job-shop problem, most systems imply unary resources. Constraints are able to improve the generated plans owing to their search centered on areas (constraint directed search) where good schedules could be expected. Since the scheduling problem is too complex to be solved by enumeration, more intelligent search strategies speed up the planning task which, in fact, results in better plans.

In the case of mid-term planning, the extensive use of constraints could result in a change of the planning paradigm. In this field, constraints can be used to allow to perform scheduling at an aggregated level. This leads to the use of cumulative constraints, adding an additional level of complexity. Preference constraints can be handled, even if they are in conflict with each other, provided it is possible to assign domains of acceptable values to each variable involved. Cost functions defined over these domains allow the specification of variable and value ordering in the constraint solving

task, serving as a basis for constraint guided search.

Further research has to be carried out to show that, in principle, the approach is reasonable to formulate a generic framework for master production planning. New trends in constraint satisfaction have to be followed and taken into consideration. A good example for this is fuzzy constraint satisfaction [26] which has been developed to handle imprecise constraint definitions. Conflicts between constraints are solved by reducing the so-called truth threshold (the truth threshold reflects the level of satisfaction for each constraint). Modifications of the truth threshold balance the conflicts and allow to find a compromise solution (see [27] as an introduction to the theory of fuzzy systems).

Acknowledgment

The starting point for the work on the Master Production Scheduling Leitstand (MPSL) was the ESPRIT-project application titled *"A Leitstand-Based Multi-Site Master Production Scheduling System"*, authored by H. H. Adelsberger, F. Jørgensen, J. J. Kanet, and U. Schreier. The current research and development is a team effort. We would like to mention Ch. Lenke and A. Tomalla for their contributions in designing the MPSL, as well as our programmers R. Prüß, J. Godau, and M. Rüschhoff for their implementation efforts. In the name of the readers, thanks to D. Meyer who turned many "Germanism" into proper English. Special thanks to W. Conen and M. Neubauer for fruitful discussions.

The research and development is performed as part of the "Forschungsschwerpunkt": *Verfahren und Werkzeuge für die Planung und Steuerung verteilter Produktions- und Logistiksysteme* at the University GH Essen.

References

[1] Adelsberger, H. H.; Gandenberger, S.; Hansen, T.M.; et al.: The Concept of a Knowledge-Based Leitstand — Summary of First Results and Achievements in Esprit Project 5161 (KBL). Computer Integrated Manufacturing, Proceedings of the Eighth CIM-Europe Annual Conference, 1992, pp. 346–356

[2] Adelsberger, H. H.; Kanet, J. J.: The Leitstand — A New Tool for Computer-Integrated Manufacturing. Production and Inventory Management Journal, First Quarter, 1991, pp. 43–48

[3] Adelsberger, H. H.; Keilmann, K.-P.; Lenke, Ch.: Master Production Scheduling Leitstand — Project Description, Wirtschaftsinformatik der Produktionsunternehmen, Universität GH Essen, Essen, 1994

[4] Aggoun, A.; Beldiceanu, N.: Extending CHIP in Order to Solve Complex Scheduling and Placement Problems, Journées Francophone de la Programmation en Logique, 1992, pp. 52–66

[5] Allen, J.: Maintaining Knowledge about Temporal Intervals, Communication of the ACM, 26, 1983, pp. 832–843

[6] Baker, K. R.: Introduction to Sequencing and Scheduling, John Wiley, New York, 1974

[7] Boizumault, P.; Delon, Y.; Péridy, L.: Solving a Real-Life Planning Exams Problem using Constraint Logic Programming, in [13], pp. 107–112

[8] Fox, M. S.: Constraint-Directed Search: A Case Study of Job-Shop Scheduling, Pitman, London, 1987

[9] French, S.: Sequencing and Scheduling, Ellis Horwood, Chichester, 1982

[10] Frühwirth, T.; Herold, A.; Küchenhoff, V.; et al.: Constraint Logic Programming — An Informal Introduction, Technical report ECRC-93-5, ECRC, München, 1993 (also appeared in: Comyn, G.; Fuchs, N. E.; Ratcliffe, M. J. (eds.): Logic Programming in Action, Lecture Notes in Computer Science 636, Springer, Berlin Heidelberg, 1992, pp. 3–35)

[11] Garey, M. R.; Johnson, D. S.: Computers and Intractability: A Guide to the Theory of NP-Completeness, Freeman, San Francisco, 1979

[12] Lažanský, J.: Practical Applications of Planning Tasks, in: Mařík, V.; Štěpánková, O.; Trappl, R. (eds.): Advanced Topics in Artificial Intelligence, Lecture Notes in Artificial Intelligence 617, Springer, Berlin Heidelberg, 1992

[13] Meyer, M. (ed.): Constraint Processing, Proceedings of the International Workshop at the Congress on Computer Systems and Applied Mathematics (CSAM) 1993, St. Petersburg, Research Report 93-39, DFKI, Saarbrücken, 1993

[14] Montanari, U.: Network of Constraints: Fundamental Properties and Applications to Picture Processing, Information Science, 7(2), 1974, pp. 95–132

[15] Petith, H.: Working with PROTOS-L, Technical Report, Institute for Logic and Linguistics, Scientific Centre, IBM Deutschland Informationssysteme GmbH, Heidelberg, 1995

[16] Prosser, P.: Hybrid Algorithms for the Constraint Satisfaction Problem, Computational Intelligence, 9(3), 1993 pp. 268–299

[17] Prosser, P.: Scheduling as a Constraint Satisfaction Problem: Theory and Praxis, in: Proceedings of the Workshop on Scheduling of Production Processes of the 10th European Conference on Artificial Intelligence, Vienna, 1992, pp. 7–15

[18] Rotterdam E.; van Denneheuvel S.; Hennis P.; et al.: Resolution of Constraint Inconsistency with the Aim to Provide Support in Anaesthesia, in: Lažanský, J.; Mařík, V.; Wagner, R. R. (eds.): Database and Expert Systems Applications, Lecture Notes in Computer Science 720, Springer, Berlin Heidelberg, 1993, pp. 541–552

[19] Sadeh, N.; Fox, M. S.: Preference Propagation in Temporal/Capacity Constraint Graphs, Technical Report CMU-RI-TR-89-2, The Robotics Institute, Carnegie Mellon University, Pittsburgh, 1989

[20] Santos, E. Jr.: On Modeling Time and Uncertainty for Diagnosis through Linear Constraint Satisfaction, in [13], pp. 93–105

[21] Schielow, N.: MARS — Mission Activities and Resource Scheduler, in: Klauck, Ch.; Müller, J. (eds.): Proceedings of the First Bremer KI-Pfingstworkshop, Report 5/95, Universität Bremen, Fachbereich Mathematik und Informatik, 1995

[22] Shoham, Y.: Artificial Intelligence Techniques in Prolog, Morgan Kaufmann, San Francisco, 1994

[23] Vollmann, T. E.; Berry, W. L.; Whybark, C. D.: Manufacturing Planning and Control Systems, 3rd ed., Irwin, Homewood Boston, 1992

[24] Wallace, M.: Applying Constraints for Scheduling, in: Mayoh, B.; Tyugu, E.; Penjaam, J. (eds.): Proceedings 1993 NATO ASI Parnu, Estonia, NATO Advanced Science Institute Series, vol. 23, Springer, Berlin Heidelberg New York, 1994, pp. 161–180

[25] Wallace, M.: Constraints in Planning, Scheduling and Placement Problems, Report CORE-93-6, ECRC, München, 1993

[26] Young, R. E.; Perrone, G.: A Fuzzy Constraint Modeling System for Concurrent Engineering, working paper, Laboratory for Machine Tools and Production Engineering, Technical University of Aachen, Aachen, 1995

[27] Zimmermann, H.-J.: Fuzzy Set Theory and its Applications, 2nd ed., Kluwer Academic, Boston Dordrecht London, 1991