Digital Enterprise Solution for Integrated Production Planning and Control

L. Monostori[1,2], G. Erdős[1],B. Kádár[1], T. Kis[1], A. Kovács[1], A. Pfeiffer[1] and J. Váncza[1,2]

[1]*Computer and Automation Research Institute, Hungarian Academy of Sciences, Budapest, Hungary*

[2]*Department of Manufacturing Science and Technology, Budapest University of Technology and Economics, Budapest, Hungary*

Abstract: Digital enterprise technologies combined with sophisticated optimization algorithms can significantly contribute to the efficiency of production. The paper introduces a novel approach for integrated production planning and control, with the description of the mathematical models and solution algorithms. The deterministic optimization algorithms are complemented by a discrete-event simulation system to assess solution robustness in case of disturbances. The methods are illustrated by describing two prototype systems and by some experimental results obtained in an industry initiated project.

Keywords: Production planning and control, discrete-event simulation

1 Introduction

The concept of *digital enterprise* – the mapping of the key processes of an enterprise to digital structures by means of information and communication technologies – gives a unique opportunity for planning and controlling the operation of enterprises [22]. Digitalized solutions are capable to connect customer order management with production planning, scheduling and control (PPC in short). On the other hand, the importance of optimization in decision making at various levels of an enterprise has greatly increased over the past decades, as companies invest in complex advanced planning and scheduling systems to replace their out-dated material requirement planning software. Digitalized data and sophisticated algorithms together can provide additional competitive advantage which cannot be achieved by applying solely the latest production technology.

The *scope* of our work is set to complex *engineer-to-order* and *make-to-order* production, where products (like turbines, assembly lines, etc.) are traditionally associated with high quality, advanced, cutting-edge technology. These industries usually require skilled and expensive human workforce. Recently, business focus has shifted from selling products to supplying a combination of products and services (like

engineering design, installation, on-site customization, maintenance). Such “extended products” are highly customized and their value is sensitive to the time of the delivery. Human resource intensive repair and recycling activities have to be planned together with normal production. Due to complex production processes and long lead times, production of components often starts before the overall design has been completed, or customization is executed in parallel with some production activities.

In our target sector, a *project-oriented* approach is taken in general for planning and controlling operations. Relying on the conventional wisdom that it can never be wrong to get work done early, existing project planners typically try to sequence activities as early as possible, subject to technological constraints and resource availability profiles. Such methods are only capable of finding a particular solution, without the ability to explore and evaluate alternatives. Hence, they cannot be used for optimization. Further on, manual intervention is typically required to deal with overloaded resources and violated deadlines. According to our experience, medium-term production plans are *re-adjusted manually* time and again, and less than 50% of the original plan is executed finally.

Our *first objective* was to develop intuitive and flexible models and fast, reliable solution techniques that scale-up well also to large production planning and control problems. Hence, we tackled both the medium-term production planning and the short-term detailed scheduling problems as *resource-constrained project scheduling* problems (RCPSP) [7]. The solution methods had to respect all the main temporal, capacity and material availability constraints and find an optimal trade-off between various costs and due date performance criteria.

Our *second objective* was to find novel, *aggregate formulations* of the production planning problem which ensure the *integrity of results* that are generated on two different hierarchical levels, on various horizons, by using distinct models and solution algorithms. Following the usual planning hierarchy, production planning determines what to do on the medium-term so as to achieve high level business objectives. On the other hand, scheduling is responsible for refining a segment of the production plan into a detailed and *executable schedule*. However, since the two levels use different models, it is open whether production plans can really be unfolded into feasible, executable detailed schedules. An essential practical concern – especially in make-to-order production – is also that the representation of the planning problem should be generated automatically, from data readily available in *de facto* standard databases of production information systems.

Both planning and scheduling problems are burdened by various *uncertainties* like estimated resource needs, uncertain capacity availability, unspecified activities due to evolving problem definition, uncertain orders, hypothetical projects, uncertain processing times, as well as unreliable delivery dates of necessary components and materials. Any method that neglects these issues is prone to generate *fragile* solutions. Disruptions hardly stop at the boundaries of the particular shop floor of an enterprise; they spread upwards in the decision hierarchy and even to other members of a production network. Due to reasons of complexity, the direct inclusion of any main uncertainty factor into our RCPSP models was out of question. Hence, our *third objective* was to assess the

sensitivity of deterministic solutions and improve the robustness of production schedules by using *discrete-event simulation* techniques.

In the sequel first we present our integrated approach to production planning and scheduling (Section 2). While we capture problems at both levels of the planning hierarchy as RCPSPs, the details of the models and the solutions techniques are fairly different. Section 3 gives an account of how we built simulation models automatically from common master data and applied simulation techniques for assessing the sensitivity of production schedules. This work had a strong industrial motivation: in Section 4 we present two prototype systems developed for two enterprises operating in the engineer-to-order and make-to-order sectors, respectively. Finally, we conclude the paper in Section 5 and give an outlook for related research activities.

2 Integrated production planning and scheduling

In make-to-order manufacturing environments, *production planning* is responsible for allocating scarce resources to customer orders over time on a medium-term time horizon (3-12 months). Usually the resources and operations are aggregated to obtain manageable problems, and to get rid of unnecessary details [27]. The objective of planning can be either

- to maximize customer satisfaction by minimizing the tardiness of customer orders, or
- to minimize the cost of completing all the customer orders on time, when delayed shipment is unacceptable.

In contrast, *detailed scheduling* deals with the allocation of operations to machines (when more machines are available to perform the same operation) and the sequencing of operations on the allocated resources. The resource capacities and capacity extensions, as well as the time window of every operation is determined by production planning and the scheduler has to take them into account as hard constraints. Usually, production planning determines the calendar week in which an operation has to be completed. The time horizon of scheduling is usually a couple of days or a few weeks. In fact, it makes no sense to make a detailed operation schedule far in the future as this would be inevitably changed due to disturbances and unforeseen events. The objective of scheduling can be to minimize the makespan of scheduled operations with the aim of finding a schedule such that all the operations planned to complete in a calendar week would finish on time.

In project-oriented planning and scheduling each production order becomes a project that has to be carried out from the ordering of materials through manufacturing, assembly and delivery to the customer. The projects compete for the various resources. If there are several long projects that take several months to complete, then the planning of capacity allocation and expansion has to be done at the production planning level, while the daily work requires a detailed schedule of the operations. Both of these problems can be modeled and solved as resource constrained project scheduling problems, where planning provides constraints for scheduling, and conversely, the planning problem is defined by aggregating the resources and the detailed operation plans of the projects. Below we

present in detail the planning and the scheduling problem, and the necessary aggregation procedures. We also discuss qualitative results.

2.1 Project-oriented production planning

In project-oriented production planning, each project is a collection of activities that are aggregated from the detailed operation plans of the projects. The aggregation procedure will be discussed in Subsection 2.2. The time horizon of production planning consists of several weeks, and the plans specify the progress of each activity of each project in every calendar week. In case of long projects, aggregated activities usually take several weeks, and the progress or intensity of the activities varies over time. Such a variation of activity-intensities can be planned, provided the chosen planning method enables the modeling of variable-intensity activities [9]. Below we introduce such a mathematical programming approach and describe some results obtained with it.

There are various results in the literature for project scheduling with variable-intensity activities. Weglarz [32] considers the model with continuous time and provides analytical results for minimizing the makespan with respect to finite-capacity resources. Leachman et al. [21] provide practical heuristics. The discrete time model with the makespan objective has been studied by Tavares [30]. The problem with activity deadlines, and the objective of minimizing the above-capacity usage of resources is studied by Hans [10], who proposes a column-generation based approach. Wullink [33] suggests a fast constructive heuristic.

2.1.1 Modeling with variable-intensity activities and feeding precedence constraints

The basic building block of our approach is the variable-intensity activity. In practice, an activity typically starts with low-intensity preparatory work and its intensity gradually increases to a maximal level. The fraction of an activity done in time period t is called the *intensity of the activity in time period t.* For an illustration, consider the two variable intensity activities depicted in Figure 1 (the horizontal axes correspond to time periods and the vertical axes to intensities). The intensity of an activity is at most one, and the sum of intensities of an activity sums up to one. Usually, there is a physical limit on which fraction of the activity can be completed during a single time period. This bound is the *maximum intensity* of the activity.

Each activity may require one or more resources and the demand is proportional to the intensity of the activity. Namely, if the total resource requirement of some activity is q_k from some resource k, and the intensity of the activity is x_t in time period t, then it requires an amount of $q_k x_t$ from resource k in time period t. Of course, the intensity may vary over time as there can be other, more urgent activities that require the same finite-capacity resources.

The output of an activity can be fed into a downstream activity. Such a relation can be captured by a *feeding precedence* constraint, which models the material and information flow between activities. More formally, a feeding precedence constraint is specified by a triple (A,B,f), where A and B are activities and f is a number between 0 and 1. It specifies that f fraction of activity A has to be completed before B may start, and the total fraction

of activity *A* done up to some time period *t* cannot be less than that of activity *B*. In Figure 1 a pair of variable-intensity activities is depicted connected by a feeding precedence constraint. In this example, 20% of activity *A* has to be completed before *B* may be started. We note that the notion of overlapping activities is well-known in project scheduling, and it means that a minimal and a maximal difference between the start times of a pair of activities are constrained, cf. [7]. But, in the case of overlapping activities there is no restriction on the progress of the two activities with respect to each other.

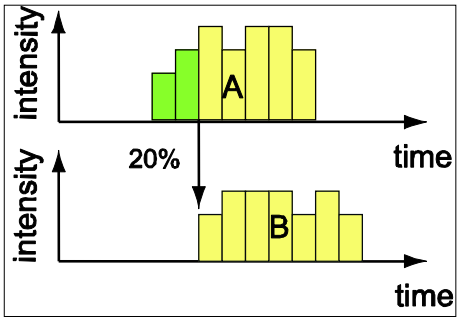

Figure 1: Feeding precedence constraint from *A* to *B*.

2.1.2 The mathematical program

We assume that the planning horizon is divided into a finite number of time periods *H*, indexed by *t*. There is a finite set of activities, *N* , and each activity *A* in *N* has a release date r_A, deadline d_A, and maximum intensity m_A, which is a number between 0 and 1. That is, each activity *A* has to be entirely processed between r_A and d_A, and in any time period *t* at most m_A fraction may be completed. There is a finite set *R* of continuously divisible resources, each resource *k* in *R* has a capacity $b_{k,t}$, but the overuse of resources is permitted. For each activity *A*, the proportion between the activity-intensity and the resource requirement from each resource *k* is specified by the constants q_k^A. The intensity x_t^A of each activity *A* has to be determined for each time period between r_A and d_A. The objective is to minimize the overuse of resources, i.e., if y_{kt} denotes the demand of resource *k* above $b_{k,t}$ in period *t*, then the objective is

$$\min \sum_{k \in R} \sum_{t \in H} y_{kt} .$$

The constraints are

$$\sum_{A \in N_{kt}} q_k^A x_t^A - y_{kt} \le b_{kt}, \text{ for each resource } k \text{ and time period } t, \tag{1}$$

$$\sum_{t=r_A}^{d_A} x_t^A = 1, \text{ for each activity } A, \tag{2}$$

$$0 \le x_t^A \le m_A, \text{ for each activity } A, \tag{3}$$

$$0 \le y_{kt}, \text{ for each resource } k \text{ and time period } t, \tag{4}$$

where N_{kt} is the set of those activities that may require resource k in time period t, i.e., $N_{kt} = \{A \in N \mid q_k^A > 0 \text{ and } r_A \le t \le d_A\}$. Constraints (1) and (4) provide lower bounds for y_{kt}. Since we minimize the sum of the y_{kt}, in any optimal solution, the value of y_{kt} will be $\max(0, \sum_{A \in N_{kt}} q_k^A x_t^A - b_{kt})$. Equation (2) ensures that activity A is entirely processed between r_A and d_A. Finally, the intensity of activity A can never be more than m_A, by constraint (3).

In addition, there can be feeding precedence constraints between pairs of activities. Suppose (A,B,f) is such a relation. Let $p_{A,f}$ denote the minimum number of time periods needed to complete the f fraction of activity A, i.e., it is the least integer with $p_{A,f} m_A \ge f$. Without loss of generality we assume that $r_A + p_{A,f} \le r_B$ and $d_A + p_{B,1} \le d_B$. We introduce new binary variables $z_t^{A,f}$ for $t \in \{r_A + p_{A,f}, ..., d_A\}$, and add the following constraints to the model:

$$\sum_{t=r_A}^{\ell-1} x_t^A \ge f(1 - z_\ell^{A,f}), \text{ for each time period } \ell \in \{r_A + p_{A,f}, ..., d_A\}, \quad (5)$$

$$x_t^B \le m_B (1 - z_t^{A,f}), \text{ for each time period } t \in \{r_B, ..., d_A\}, \quad (6)$$

$$z_t^{A,f} \ge z_{t+1}^{A,f}, \text{ for each time period } t \in \{r_A + p_{A,f}, ..., d_A - 1\}. \quad (7)$$

$$\sum_{t=r_A}^{\ell} x_t^A \ge \sum_{t=r_B}^{\ell} x_t^B, \text{ for each time period } \ell \in \{r_B, ..., d_A\}, \quad (8)$$

The binary variable $z_t^{A,f}$ takes value 1 if less than the f fraction of activity A is completed by the end of time period t-1. Constraint (5) ensures that $z_t^{A,f}$ takes value 0 only if the f fraction of activity A is completed by the end of time period $\ell - 1$. By inequality (6), the intensity of activity B can be positive in time period t only if the f fraction of activity A is completed by the end of time period t-1. The values of the binary variables $z_t^{A,f}$ are non-increasing, by (7). The feeding nature of the precedence constraint is captured by inequality (8). There is a new set of constraints for each feeding precedence constraint (A,B,f), but the variables $z_t^{A,f}$ are defined only for distinct (A,f) values. If f=1, then the constraint (8) is superfluous, as there can be no overlap between A and B.

The above model could be amended with a different objective function. For instance, we can minimize flow time related criteria subject to fixed capacity constraints. Suppose the activities are partitioned into jobs, each job being a partial order of a distinct subset of activities. For the last activity A of each job we introduce a set of binary variables $z_t^{A,1}$ and add the constraints (5)-(8) to the model. Then, the *weighted flow time* can be expressed as

$$\min \sum_{A} \sum_{t \in [r_A + p_A, d_A - 1]} c_A t (z_t^{A,1} - z_{t+1}^{A,1}),$$

where c_A is the weight associated with A, and the first summation is over the last activities of all jobs. Weighted flow time is an appropriate measure to minimize *work-in-process* (WIP) inventories.

2.1.3 Solution with Branch-and-Cut

Branch-and-cut is one of the most successful techniques for solving mathematical programs with integer (binary) and continuous variables, with linear constraints and linear objective function. Similarly to branch and bound, the linear relaxation is solved in each node of a search tree, but before branching, new valid inequalities are added to the problem that cut off the fractional solution available at the node [13].

We have also applied this method for solving the mathematical program modeling the project scheduling problem. The crux of the method is the automatic generation of cutting planes that cut off the fractional solutions at the nodes, and strengthening the relaxation in this way. Standard solvers, like Fair-Isaac's XpressMP, or ILOG's CPLEX provide a host of subroutines to generate cuts, but the user can also generate problem specific cuts. We briefly discuss a problem specific cut that we used in our computational experiments.

Our problem specific cuts are generated with respect to feeding precedence constraints. That is, we consider the convex hull of feasible solutions of the linear system consisting of the inequalities (2), (3), (5) and (7). Let K_f^A denote the convex hull of those $(x^A, z^{A,f})$ vectors with binary $z^{A,f}$ coordinates that satisfy the inequalities (2), (3), (5) and (7). To simplify the presentation, let m denote the maximum intensity of activity A, and p the least integer with $p \cdot m \geq f$. Moreover, let r and d be the short-hand notations for r_A and d_A, respectively. If $f = 1$, then K_f^A can be fully characterized by the following result:

Theorem 1. (Kis [14]) K_1^A consists of the vectors (x, z) in $[0, m]^{d-r+1} \times [0,1]^{d-r-p+1}$ that satisfy the following linear constraints:

$$\sum_{t=r}^{d} x_t = 1,$$

$$x_t \leq m z_t, \quad t \in \{r + p + 1, \ldots, d\},$$

$$x_t \leq m, \quad t \in \{r, \ldots, d\},$$

$$z_t \geq z_{t+1}, \quad t \in \{r + p + 1, \ldots, d - 1\},$$

$$m_r z_{t_1} + m \sum_{t \in S_1 \setminus \{t_1\}} z_t + \sum_{t \in \{r, \ldots, t_1\} \setminus (S_1 \cup S_2)} x_t \geq 1 - m \,|\, S_2 \,|,$$

where $m_r = 1 - (p-1)m$, $\varnothing \neq S_1 \subseteq \{r + p + 1, \ldots, d\}$ and $S_2 \subseteq \{r, \ldots, r + p\}$ are such that $|\, S_1 \,| + |\, S_2 \,| = p$ and t_1 is the greatest element of S_1.

If $f < 1$, then the characterization of K_f^A is more involved:

Theorem 2. (Kis [15]) Let $m_r = f - (p-1)\cdot m$. K_f^A equals the set of vectors (x,z) in $[0,m]^{d-r+1} \times [0,1]^{d-r-p+1}$ that satisfy the following linear constraints:

$$\sum_{t=r}^{d} x_t = 1,$$

$$x_t \le m z_t, \quad t \in \{r+p+1,\dots,d\},$$

$$x_t \le m, \quad t \in \{r,\dots,d\},$$

$$z_t \ge z_{t+1}, \quad t \in \{r+p+1,\dots,d-1\},$$

$$m_r z_{t_1} + m \sum_{t \in S_1 \setminus \{t_1\}} z_t + \sum_{t \in \{1,\dots,t_1\} \setminus (S_1 \cup S_2)} x_t \ge f - m\,|S_2|,$$

for every $\varnothing \ne S_1 \subseteq \{r+p+1,\dots,d\}$, $S_2 \subset \{r,\dots,r+p\}$ and $|S_1|+|S_2| = p$, t_1 being the greatest element of S_1;

$$(f-1)z_\ell + \sum_{t \in U_\ell^1} m z_t + \sum_{t \in U_\ell^2} m z_\ell + \sum_{t \in \{r,\dots,d\} \setminus (U_\ell^1 \cup U_\ell^2)} x_t \ge f,$$

for $\ell \in \{r+p+1,\dots,d\}$, $U_\ell^1 \subseteq \{p+1,\dots,\ell\}$, $U_\ell^2 \subseteq \{\ell+1,\dots,d\}$, $m\,|U_\ell^1 \cup U_\ell^2| \le 1$, and $m\,|U_\ell^2| \le 1-f$;

$$\sum_{t \in U} (x_t - m z_t) \ge 1 - f, \text{ for } U \subseteq \{r+p+1,\dots,d\} \text{ with } m\,|U| \le 1.$$

The above characterizations directly lead to polynomial time separation algorithms.

The methods have been extensively tested on benchmark instances from the literature, for details, see Kis [14], [15]. Qualitatively, generating problem-specific cuts really pays-off for strict precedence constraints, i.e., f =1 for all precedence constraints. By adding our cuts we have obtained better results than without them. However, for small values of *f*, adding our cuts has not improved solution quality that much. At extremity, when *f*=0, our cuts are meaningless, since the mathematical problem becomes linear without any binary variables.

2.2 Automated aggregation of production data into planning models

The first step towards the integration of medium-term production planning and short-term scheduling is ensuring that the two levels *operate on common data*. In make-to-order production, customer orders, technology and resource related data are typically available in the enterprise resource planning (ERP) systems of the company in the form of

production orders, bill of materials (BOM), technological routings, and resource calendars. These data can be mapped almost directly to detailed scheduling models, whereas production planning requires a less fine-grained representation. Therefore, the *aggregation* of detailed production data into high level planning models is of key importance [31].

According to the model above, we consider each production order as a separate project. Projects have strict release dates and deadlines, defined by the customer's deadline. Each project consists of a set of manufacturing operations, which is determined by the BOMs and routings. Our model assumes that there are no BOM or routing alternatives. Each individual *operation* of the manufacturing process is described by a *task* in the detailed scheduling model. Task i is characterized by its processing time p_i, sequence-independent setup time s_i, and the number of units q_k^i it requires of each resource $k \in R$. The prescribed amount of all required resources is occupied both during the setup and the processing of the task i. Pairs of tasks can be connected by precedence constraints $i \to j$, each of which describes that task i must be completed before the start of processing task j (though, task i and the setup before task j can proceed in parallel). Note that the tasks together with the precedence relations between them define a graph representation of the project, in which labeled vertices correspond to tasks and edges to precedence constraints. We briefly call this representation the *project graph*. We assumed that every project graph forms an in-tree, which usually holds when the manufacturing process involves machining and assembling operations only.

The above data serve as the input of the aggregation procedure to construct the planning level representation. Aggregation can happen in terms of time, resources, and activities. We fixed the aggregate time unit to one week – and therefore in the sequel we briefly use the word *week* to refer to the planning level time unit. Our aggregation techniques would allow the aggregation of resources, i.e., replacing several related physical resources by one aggregate resource in the planning model. However, since resource aggregation does not contribute significantly to the reduction of the running time of the planner, but leads to less precise planning models, we do not use resource aggregation. Activity aggregation corresponds to partitioning the project graphs, and merging the tasks in each partition into one aggregate activity. Planning is performed on these larger activities. The way the aggregate activities are built is crucial for the feasibility and the quality of the production plans. For instance, Figure 2 presents two distinct aggregations of the same project graph. Here, model B is preferable to A because it is less constrained by precedence relations and its minimal lead time is thus shorter by one week.

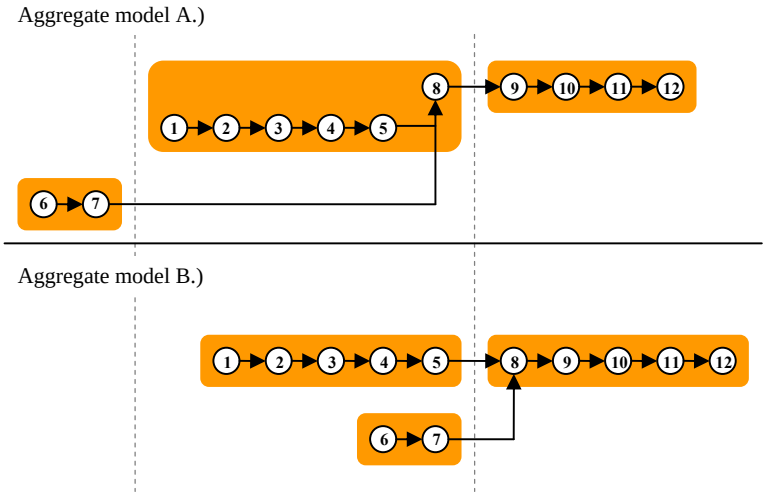

Figure 2. Two alternative aggregate models built from the same project graph.

In order to ensure that the temporal interdependencies of aggregate activities can be captured by simple precedence constraints, we look for connected components of the project graph. Finding the most appropriate partitioning is an optimization problem with the following criteria:

- The set of tasks contained in each aggregate activity must be, in itself, schedulable in one week.
- The longest directed chain of precedence relations between aggregate activities should be minimal.
- The number of aggregate activities should be as small as possible, in order to ensure the compactness of the production planning model.

We have presented details of the aggregation in [31], while in [16] proposed polynomial-time algorithms to find such a partitioning of the project graph. The resource requirements of an activity A are then computed as the sum of resource requirements of its contained tasks $i \in A$: $q_k^A = \sum_{i \in A} q_k^i (p_i + s_i)$. The deadline of activity d_A is set to the project deadline, and we allow the complete activity to be performed in one time unit, $m_A = 1$. Two aggregate activities A_1 and A_2 are connected by a precedence constraint $A_1 \to A_2$ if there exist two tasks $i_1 \in A_1$ and $i_2 \in A_2$ connected by a precedence constraint $i_1 \to i_2$. The generated planning problem is then submitted to the planner presented in Section 2.1.

We note that the extension of the model to the case when precedence relations define an arbitrary directed acyclic graph (DAG) is straightforward, but some of the implied graph problems become more complicated. For instance, the implied partitioning problem is tractable in polynomial time for trees, while it is NP-complete for DAGs.

2.3 The detailed scheduling problem

Short-term scheduling unfolds the (first units of the) production plan into executable task sequences on a horizon of a couple of days or weeks. Particularly, we set the scheduling horizon to one week, i.e., to the time unit of the production planner. If a detailed schedule is required for more than one week, then the scheduling problems of different weeks can be considered separately, and hence, the following steps must be repeated as many times as the number of independent detailed schedules. Detailed scheduling considers all relevant technological and economical constraints present in the given application, as well as the available resource capacities defined by the production plan. The objective of the scheduler is (1) to construct a detailed schedule without violating the specified time and capacity limits, and (2) to hedge against unforeseen disturbances. We mapped these requirements to the optimization criterion of *minimal makespan*.

The set of tasks to be scheduled in different weeks is received by disaggregating the activities in the production plan. Each task $i \in A$ is assigned to exactly one week, based on the intensity of the aggregate activity A over time. If the entire activity A is planned for week t, then all of its constituent tasks are assigned to t. If A spans over various aggregate time units with intensities x_t^A, then tasks $i \in A$ are ordered by decreasing distance from the root task of A, with path lengths calculated as the sum of processing times p_j along the path. The tasks are then assigned to weeks in this order, proportionally to intensities x_t^A, with appropriate rounding where necessary. Two tasks i and j are interconnected by a precedence if they are also interconnected in the routings. Also, each task i requires q_k^i units of resource k as specified in the routings. Resource capacities may vary over time, taking into account the capacity extensions defined in the production plan. The solution of the scheduling problem is an instantiation of the start times variables S_i of the tasks.

2.3.1 Constraint-based solution techniques

We apply *constraint-based scheduling* (CBS) techniques [3] for solving the above detailed scheduling problem. CBS takes a declarative model of the scheduling problem, expressed in the form of variables (the start time of tasks), variable domains (the time horizon), constraints (resource and precedence) and an optimization objective (makespan, in our case). Then, the constraint solver looks for a binding of these variables to a value from their domain that satisfies all constraints and minimizes the objective value.

Solution techniques in CBS rely on an effective combination of search and inference. The search procedure – usually some kind of a backtracking search – explores the space of potential solutions, while inference restricts search to promising regions of that space. A fundamental inference method is *constraint propagation*. A propagator is an algorithm attached to an individual constraint that removes values from the variables' domains that are inconsistent with the given constraint. Constraints that can be handled efficiently by CBS are those for which there exists an efficient propagator, i.e., one that finds nearly all inconsistent values sufficiently fast.

Typical CBS models, such as ours, differ from other, more widely known mathematical programming representations of planning and scheduling problems in a number of ways. First, in CBS the most widely used variables are the start and end times of tasks. The domain of these variables can be arbitrarily fine-grained without particular degradation of computational efficiency. This model is principally suitable for non-preemptive scheduling problems. While theoretically both constraint and mixed-integer programming can tackle any problem in NP, the set of constraints that can be handled efficiently by the two techniques differ essentially. In CBS, the propagation of precedence constraints $i \rightarrow j$ is straightforward even when delays are specified: the earliest start time of j, as well as the latest start time of i can be adjusted. Very efficient propagation algorithms are available for resource constraints, which state that tasks can use at most a given number of units of a certain resource at any point in time: edge-finding [5] deduces which tasks must (or must not) start before (or after) a set of tasks that require the same resource. The handling of sequence-independent setup times is straightforward using the above techniques, and also, significant results have been published on propagating sequence-dependent setup times [8]. On the other hand, the performance of CBS techniques is sensible to the choice of optimization criteria. Generally, maximum-type criteria, such as makespan or peak resource usage, can be propagated efficiently, whereas sum-type criteria, such as total flow time or tardiness, are more challenging. Various constraint-based solvers are available for solving such scheduling problems, among which we have used ILOG Solver and Scheduler.

2.3.2 Exploiting the structure in industrial problem instances

Generic models of scheduling problems, such as e.g., RCPSP, have had key importance in the development of solution approaches such as constraint-based techniques, and in general, of scheduling theory. Nevertheless, these models hide some actual properties of industrial problem instances, the exploration and exploitation of which would be of key importance for efficient solvability. Hence, we have developed two novel different methods that reveal the hidden structure and exploit it to improve solution efficiency.

A typical structural property of industrial scheduling problem instances is that there are several, in a sense, similar projects: there might by many orders for the same end product, or the same component can be used to build different end products. In [17] we have defined a notion of similarity on a graph theoretical basis, as a special kind of sub-graph isomorphism on two project graphs. This definition of similarity is exploited in so-called *progressive solutions*, in which the sequencing decisions among corresponding tasks of similar projects are made by simple inference methods in a pre-processing step. It has also been demonstrated that restricting the search space to progressive solutions can reduce considerably the solution times of practical detailed scheduling problems.

Figure 3 presents two pairs of similar projects. It is assumed that tasks with identical indices have equal durations and resource requirements. *P* and *Q* are completely identical project graphs, while *R* and *S* have isomorphic sub-graphs. The latter pair can represent the manufacturing of two different products from the same family: *R* has an additional component and *S* has an additional finishing operation, e.g., painting. Now, it is easy to see that there exists an optimal detailed schedule in which corresponding tasks of *P* and *Q*, as well as *R* and *S* are executed in the order shown in the figure (since *P* and *Q* are

identical, the direction of all precedence constraints between these two projects could be reversed). We note that this technique is applicable if there are no BOM and routing alternatives, or the actual alternative to use is already known at the time of detailed scheduling. For more details, including computational experiments on the efficiency of progressive solutions, the reader is referred to [17]

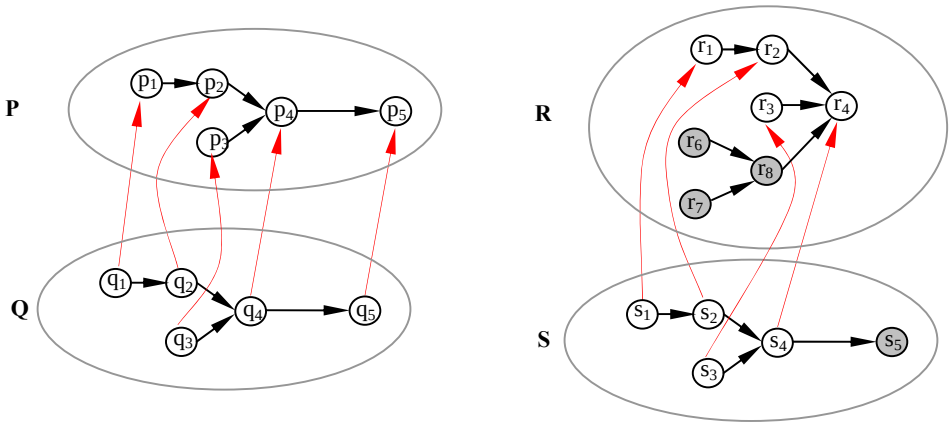

Figure 3: Precedence constraints deduced in a progressive solution when two identical projects are to be scheduled (left). Precedence constraints deduced when two end products are build from partly identical components (right).

Another typical structural property is that the scheduling problem consists of a number of loosely connected sub-problems. This might hold due to plants composed of several departments or different product families requiring different sets of resources. *Freely completable partial solutions* (FCPS) address the exploitation of this structural property [18]. An FCPS is a feasible binding of a subset of the variables that is compatible with any instantiation of the remaining variables. If an FCPS is found in any search node, then the contained variables can be bound to the values they take in the FCPS, which reduces the size of the problem yet to be solved.

3 Simulation support for PPC

In the following part the role of simulation-based schedule evaluation as well a new approach will be described including its structure, model building and shop-floor emulating features.

3.1 Production control and simulation

The discrete-event simulation (hereafter referred to as simulation) approach has been applied to decisions in planning and scheduling, related to production applications (see e.g., [2], [20], [24]). The simulation models that are used for making or evaluating these decisions generally represent the flow of materials to and from processing machines and the operations of machines themselves [28]. Potential problems can be identified and can be corrected using a simulation model. By far the most common use of simulation models is for *operational decisions* such as planning and scheduling [20].

Simulation captures those relevant aspects of the PPC problem which cannot be represented in a deterministic, constraint-based optimization model. The most important issues in this respect are uncertain availability of resources, uncertain processing times, uncertain quality of raw materials, and insertion of conditional operations into the technological routings. Aytug et al. [1] give a broad overview in their study on production schedule execution in the face of uncertainties. Categorization of uncertainty is formulated for a better understanding of the meaning of uncertainty during the calculation or execution of a production plan or schedule.

In simulation supported schedule evaluation, simulation is often used for evaluating the *robustness* of the schedule. A production schedule is termed robust in case it performs well after a disruption. The usefulness of simulations lies in detecting and preventing these problems concerning robustness before the detailed, short-term schedule reaches the shop floor. Thus, the key benefit of a simulation-based schedule evaluation system is the *feedback* about schedule performance (e.g., number of late tasks) which, in turn, can be used for improving subsequent solutions. Recently applied schedule evaluation methods of (predictive) production schedules during simulation-based assessment are classified in [19] regarding the environment of the evaluation (static/dynamic) and the evaluation criteria of the schedules (absolute/relative).

A number of authors present simulation-based experimental studies with the aim at analyzing scheduling problems and schedule evaluation techniques in a dynamic and stochastic environment. The categorization of the selected papers is also highlighted in [25]. The analytical solutions proposed in [25] are able to estimate important performance measures for schedule evaluation methods in a dynamic, stochastic manufacturing system, and are evaluated in simulation testbeds. In [4] and [6] the simulation-based execution of the calculated schedules is introduced considering uncertain activity durations in the form of probability distributions. Regarding the robustness and flexibility of tardiness and total flow-time in job-shops, several schedule repair methods are investigated in [12], and an experiment is performed on a set of benchmark problems by executing schedules against simulated machine breakdowns. Sabuncuoglu et al. [29] propose a simulation-based approach for testing the rescheduling

methods in a dynamic and stochastic manufacturing system, applying uncertain processing times and machine breakdowns. In their approach the system consists of three components: simulation model, controller and scheduler. An interesting combination of deterministic and stochastic simulation is given by Honkomp et al. [11]. They describe a simulator for semi-continuous and batch processing manufacturing environments that can accept deterministic schedules and simulate both a deterministic and a stochastic realizations of the schedule. Running two versions of the simulation the authors compare the performance and robustness of the schedules.

3.2 Model building and schedule evaluation

3.2.1 Requirements and functions

In our case, the proposed simulation module is utilized as a component of a higher level system taking the role of the real production system. The reasons of connecting the scheduler to a discrete-event simulator are twofold. On the one hand, it serves as a benchmarking system for evaluating the schedules on a richer model. On the other hand, it covers the non-deterministic character of the real-life production environment. Additionally, in the scheduling phase it is expected that the statistical analysis of schedules should help to improve the robustness of execution and support the scheduler during the calculation of further schedules. The main functions of the discrete-event simulator are as follows: it

- evaluates the robustness of weekly schedules against the uncertainties, performs sensitivity analysis of the schedules,
- helps in visualizing and verifying the results of a PPC system,
- supports the systematic test of a pilot PPC system, and
- supports rescheduling decisions.

The evaluation of schedules is measured over several runs of the discrete-event simulation where the number of replications (independent simulation runs, with different random numbers) depends on the setting of confidence intervals. The main requirements towards the simulation module are as follows:

- common data, bi-directional connection to the scheduler,
- support for input/output inspections,
- support for different playback strategies,
- from 1 day to 1 week playback time horizon, and
- short response time, making multiple model runs possible.

3.2.2 Main phases of the simulation

In order to meet all the requirements and achieve the desired functionality for a flexible simulation system, a so-called *component-based simulation* method has been developed [25]. The simulation module and the finite capacity job-shop scheduler have connections

to the same production database (see Figure 4). Resources, products, process plans, production information, i.e., *directly* and *indirectly* usable data are transformed exactly to the same form for all system components. Note that *simulation relevant data* (e.g. resource model, execution policies, process flow model) are stored locally in the simulation model.

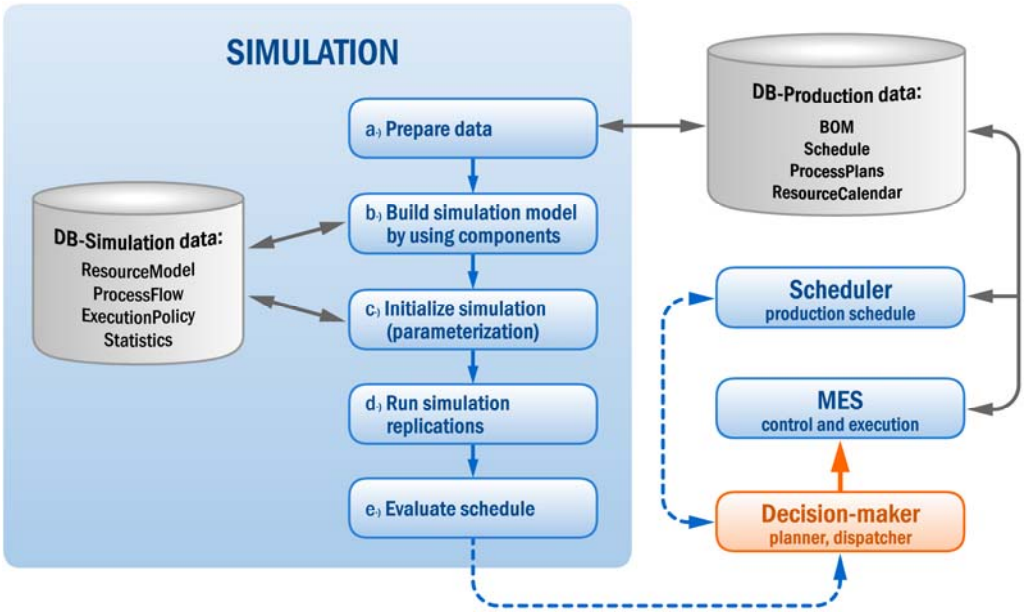

Figure 4: Architecture and the main process flow in the simulation module.

Hereby, the complexity of integrating the simulation module into the system is significantly reduced. None the less, the common data tables ensure data integrity during the creation of the simulation model; moreover, the data-model serves as a basis for the more detailed shop-floor model. Running the simulation by applying the basic data tables results in a waste number of queries during the model run, reducing the simulation speed significantly. However, in order to ensure sufficient number of simulation replications for the evaluation of a short-term production schedule, the total response time should be minimized. In order to resolve the above two contradictory objectives an exhaustive data pre-processing phase is included in the simulation process.

Data preparation is carried out before the overall simulation (see Figure 4). The redundant data storage in the simulation model is compensated by the advantage of the shorter response time. Modelling real production systems frequently brings up the problem of handling hundreds of resources in a simulation model. Having the modelling objects in hand, which were created on the base of the conceptual model, in our architecture the *simulation model* is created *automatically* based on the pre-processed data (phase b). The automatic generation of the model is followed by initialization (phase c). There, besides classical parameter settings, the procedure involves the generation of input parameter specific model components (entities such as products, operators). Contrary to the previous phase, this one is carried out for each replication. The simulation runs are repeated until the required number of replications is obtained (phase d). Each

replication is a terminating, non-transient simulation run. In the last phase, the schedule is evaluated by using the evaluation criteria and the results of the evaluation process are interpreted by the *Decision-maker* (e.g., planner) who is responsible for taking the necessary actions.

3.2.3 Novel model of the production execution process

Regarding resource modelling of the designated production system (e.g. flow/job-shop model), a machine model class library was developed and applied for all the machine resource instances [26]. These classes are pre-programmed component objects in the simulation, consisting of a generalized model of the resource, a built-in execution policy as well as the process flow. Tasks represent the operations on particular *parts*, together with the assigned machine and human operator resources. As a main principle, the simulator should play back the schedule without changing the optimized sequence of the tasks, but considering the calculated start times of the processes. Therefore, as a new solution, an ordered *queue* of the tasks is built up in front of each scheduled machine (*TaskObject* in Figure 5), and the tasks to be processed are forwarded into these queues.

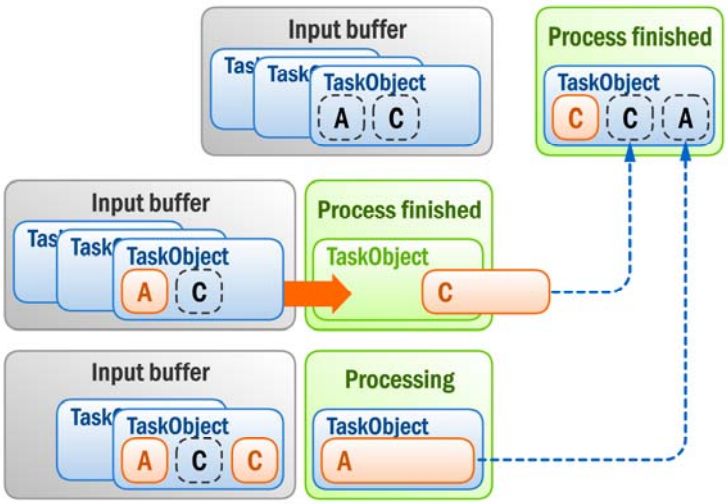

Figure 5: Object-oriented model of the execution of the pre-calculated production schedule in the simulator, by applying the *TaskObject* structure.

Each task has a list of the *TaskObjects* to be visited during the manufacturing process, according to its routing and the production schedule. Arriving tasks at the machines are processed in the simulation as follows (see also Figure 5):

- Tasks waiting for processing are stored in the input buffers of the machines, always sorted by their starting times. Each task in the schedule contains one of these *TaskObjects*, which, at the initialisation phase of the simulation, are distributed to the input buffer of the machines.
- The first *TaskObject* in the input buffer queue reserves the first position of the input buffer on the machine. This ensures that the designated machine is reserved for the designated task.

- If there is a *TaskObject* in the input buffer of the machine, which becomes ready at the moment, the setup process will be immediately started (regarding the task represented by the *TaskObject*). Setup has to be started also when the part itself has not arrived yet at the *TaskObject*. In this case, there is no event generated by the arrival of the part for the simulation, however, because of the first criterion enlisted above, it is not allowed to start the setup process based on the calculated starting times. The proposed solution is to start the setup process, but freeze it immediately, before requesting the operator. It will be restarted only if the simulation time equals the planned starting time of the setup process.
- In order to start the process at least one operator is needed with the designated service skills. One setup operation is executed by one operator.
- If the setup process has finished and – in case it is an assembly process – all the required parts (component type) are already in the input buffer, then the main process can be initiated.
- Before the parts are reallocated to the machines, the processing times of the tasks are set, according to the specified attributes of the part objects. The processing of the parts is realized on two machine objects. For the first one no operator (with service) is necessary, while, for the second, it is mandatory to have at least one operator (see fourth criteria). In this case, the processing time of the first machine is calculated as the total machine time minus the operator time. Of course, for the second machine the processing time equals the operators' time.
- If no free operator is available then the process cannot be started.
- After processing, parts are sent to a virtual transportation unit (for a predefined transportation time interval), before being reallocated to the input buffer of the next machine.

4 Applications

The models and solution methods presented in the previous sections have been implemented and embedded in two prototype planning and scheduling systems. The PROTERV *production planning* system addresses engineer-to-order and make-to-order enterprises, and operates on the high level description of planning level activities. The PROTERV-II system performs *integrated production planning and scheduling*, and incorporates the presented aggregation procedures to generate the planning level representation automatically from detailed scheduling data. The PROTERV-II system was further coupled with a simulation model of a real factory. This simulation model emulates the real execution environment and supports the analysis of the generated schedules.

4.1 PROTERV: Prototype planning system

The PROTERV prototype planning system has been developed to solve project-oriented, integrated production and capacity planning problems that are typical in engineer-to-

order and make-to-order type production. It directly builds on the project model and customized mathematical solver presented in Section 2.1.

A *projects* includes various activities needed to complete an order, like engineering design, technological process planning, components manufacturing, assembly, programming, documentation, installation and deployment. Although some ordering of the activities is to be followed, many of them may overlap in time, especially in case of large, complex projects. These temporal relations between activities are captured by feeding precedence constraints. Each activity may call for the use of a number of different resources. The *resources* are typically either machine or human resources (or both, in a coupled way) that will be shared by the activities of different projects. The system handles both *internal* and *external* resources, the use of which incurs different costs. The resources may be distributed, geographically dispersed and may even belong to different organizations. Resources are typically aggregated and their amount available is given in working hours per each week. Each project has a *time window* set by its negotiated earliest starting time and deadline. The intensity of executing an activity may vary over time; the activity can even be pre-empted. Hence, processing times of activities are not fixed a priori. The *decisions variables* are just the intensities of activities throughout the planning horizon; once these values are determined, it is straightforward to calculate the actual amounts of work to be performed on the activities in each week, as well as to summarize demand for the various capacities.

Hence temporal – both time window and precedence – constraints are hard, *projects compete for limited resources*. Excessive use of external resource capacities is possible only if there is no way to avoid this but violating some customer order deadlines. The primary *objective* is to minimize the total cost of this extra capacity usage, while the secondary objective is to reduce WIP as far as possible. All decisions are made with regard to the actual set of orders and production load, instead of using data collected in the past.

PROTERV obtains all basic project and resource calendar data from the ERP system of the enterprise through file exchange. The uploaded planning problem can be interactively modified in order to build different *scenarios* from the initial problem. The system's graphical user interface helps human planners develop a number of plan variants, organized in a *tree hierarchy* of scenarios. The scenarios are created by inheriting a predecessor scenario. The project related parameters (like starting time, deadline, precedence relations of tasks, probability of acceptance etc.) can be modified through a project editor (see Figure 6). The scenario variants may differ in the time windows of the projects, the set of projects selected for planning and the actual resource capacities in each time unit. Through this editor, it is also possible to investigate the internal structure of a project: its activities, the time windows of the activities, as well as the feeding precedence relations between them.

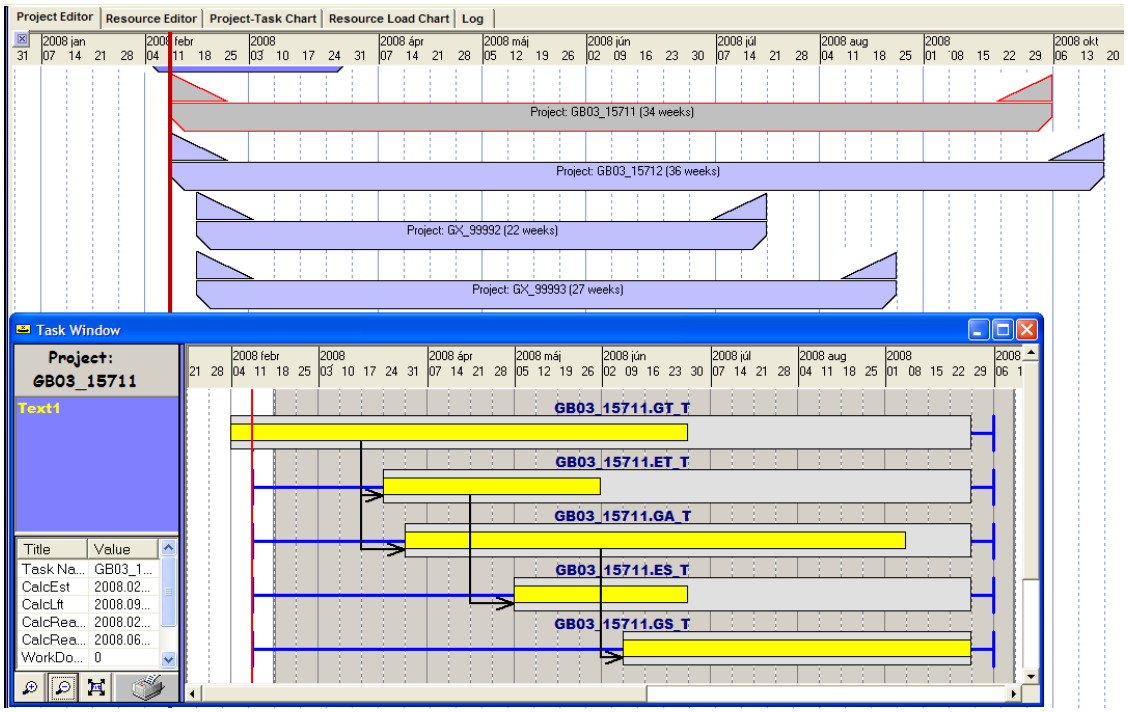

Figure 6: Screenshot of the project editor's graphical user interface.

For a given scenario, PROTERV calculates the *production plan* based on the actual production load, resource capacity limits and accepted production orders for several months or even years ahead. For instance, Figure 7 shows the plan of a particular project, i.e., the calculated work amounts for all of its three component activities, week by week.

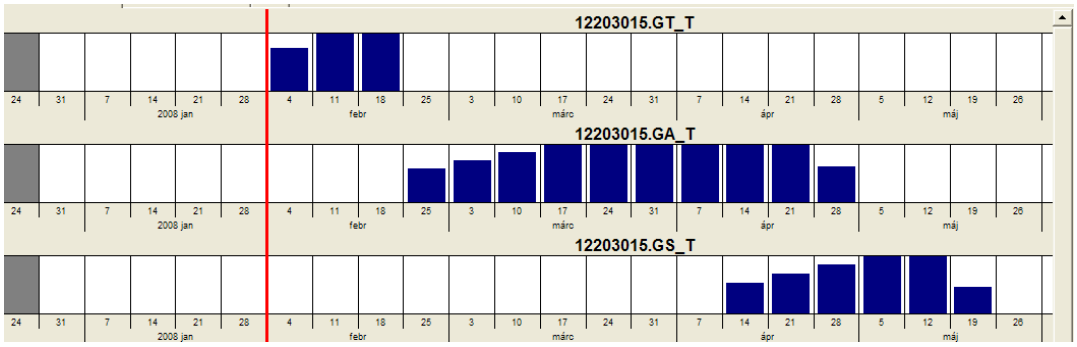

Figure 7: Plan of a particular project, with three variable-intensity activities.

The other main result of the system is the *capacity plan* which is given in terms of the aggregated weekly load of resources, with eventual external capacity requirements. As an example, consider Figure 8. Here, for two resources, against the pale skyline of available capacities, one can see aggregate planned resource demand on a weekly basis. Demand that exceeds the actual capacity limit is highlighted. Even though PROTERV strives to minimize extra capacity usage, there are periods when some extra capacity is needed so as to complete projects by the given deadline. As mentioned before, the user has an

opportunity to change the time window and status of projects in the project editor; future load of projects in hypothetical status is distinguished from the load of running projects.

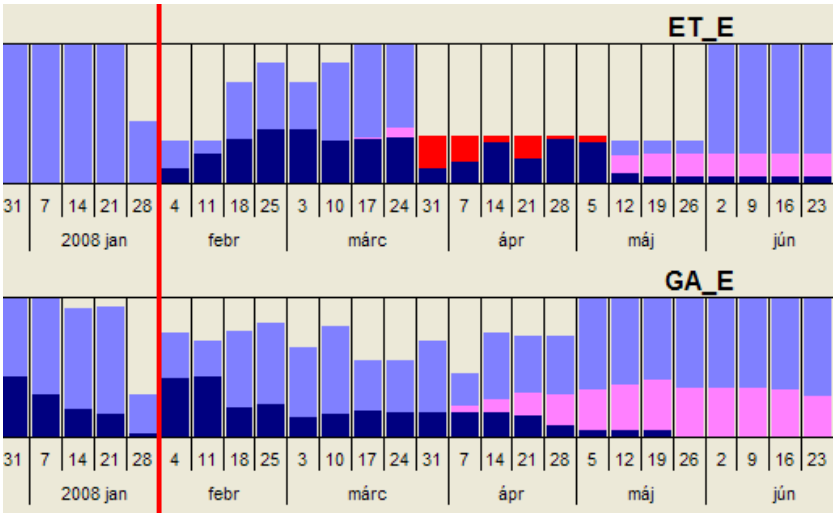

Figure 8: Resource load chart of two aggregated resources.

The different scenarios are solved by a generic planning engine that uses the solution method presented in Subsection 2.1.3. It was essential to keep the response time short, thereby allowing the users to work interactively with the system. Thanks to the efficient customized solver, the PROTERV system can be applied in a dynamic setting when re-planning is initiated by unexpected changes, as well as for creating "*what if*" scenarios with hypothetical projects (projects in proposal phases) and with possible resource extensions in the future. This way *available to promise* and *capacity reservation* decisions can be made on a more reliable estimate of the timing and resource needs of new projects. The system has been successfully tested on production planning problems whose main features are summed up in Table 1.

Table 1: Characteristics of the aggregate planning problems solved.

Input	**Size**
Aggregate resources	5 - 10
Running projects	50 - 80
Activities per project	3 - 8
Planning horizon (medium-term)	50 - 100 weeks
Solution time (worst case)	5 - 10 min

4.2 PROTERV-II: Integrated planning and scheduling system

The PROTERV-II prototype system performs *integrated production planning and scheduling* using the models and algorithms presented in Section 2. The system addresses make-to-order project-oriented production problems. It has a graphical user interface that facilitates its use as a *decision support system* at both levels of the decision hierarchy.

PROTERV-II takes as *input* the list of production orders, BOMs, routings, resource capacities and calendars, as well as shop-floor status information. Instances of production planning problems are automatically generated from this detailed data by using the aggregation methods presented before (see Subsection 2.2). On this level, the first optimization criterion is to *minimize the use of capacity extension*, while the secondary criterion is to *minimize WIP*, both subject to strict deadlines. The planning-level horizon is 15-30 weeks long, while the time unit is one week. The objective of the scheduler is to unfold the production plan into feasible detailed schedules. The time horizon of the detailed scheduler is one week, and it determines task start times with a precision of 0.1 hours.

For solving *production planning* problems on the medium term, the same planner engine is used as in the PROTERV system. Figure 9 presents the resource view of a production plan for a given worker group, displaying for each week the number of working hours planned. This worker group is one of the bottleneck resources of the factory: the normal capacities are completely exploited and in several weeks overtime is also required. Note that there are two weeks where the available capacity of the worker group is smaller than in other weeks: this is due to national holidays.

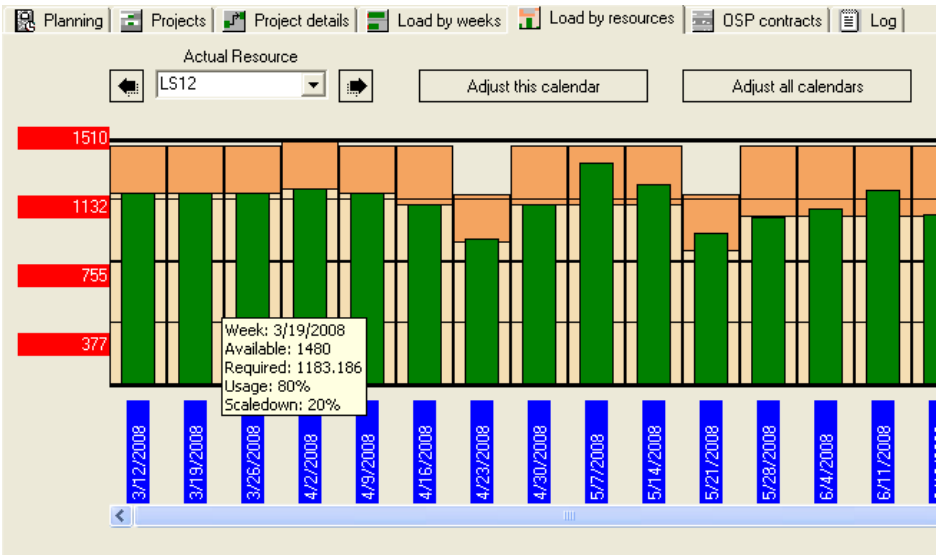

Figure 9: Medium-term production plan – load of a particular resource.

On the level of detailed scheduling, our special solution algorithms have been realized on the top of the commercial mathematical programming and constraint programming software of ILOG (see also Subsection 2.3).

In Figure 10 a segment of a detailed schedule is presented over a horizon of one week. Each row of the diagram contains a sequence of tasks to be performed on one component of an end product. The green boxes correspond to tasks, while the arrows connecting them represent the precedence constraints. Some task sequences are joined together when the corresponding components are assembled. In this example one can also notice that

detailed scheduling indicated the need for one extra weekend shift, which was not visible on the planning level.

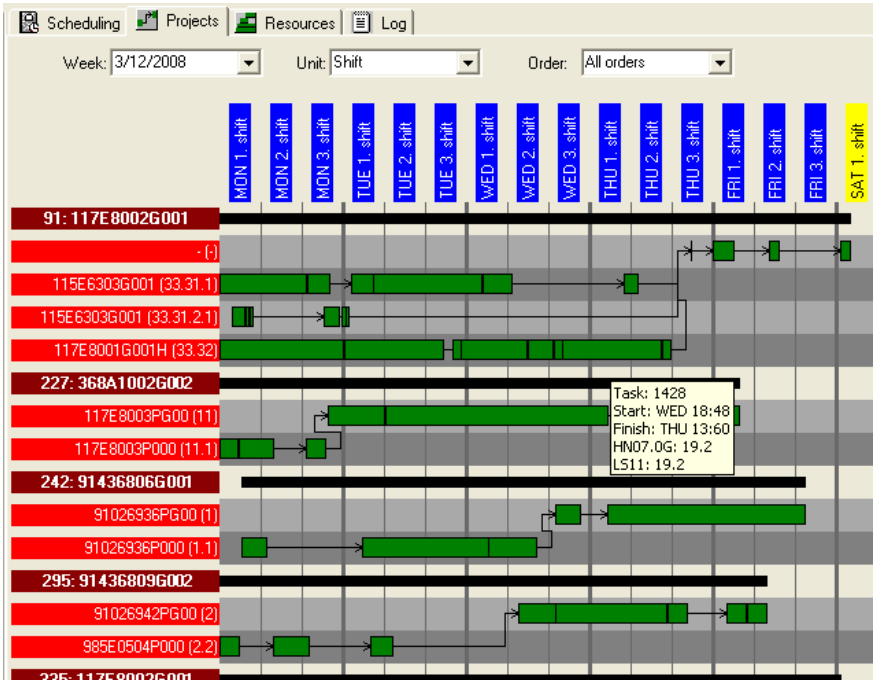

Figure 10: Short-term detailed schedule – details of a particular project.

The features of typical problem instances tractable by PROTERV-II are given in Table 2.

Table 2: Characteristics of the integrated planning and scheduling problems solved.

Input	**Size**
Resources (machines and operator groups)	80-120
Running projects	600 - 1200
Tasks per project	20 - 500
Duration of tasks	1 - 120 hours
Planning horizon (medium-term)	15 - 30 weeks
Scheduling horizon (short-term)	1 week

Table 3 displays computational results that characterize the integration of the two decision levels in Proterv-II. Each row of the table corresponds to one problem instance, i.e., one production planning problem on a horizon of 15 weeks and the induced 15 weekly detailed scheduling problems. The overall number of tasks in the problem instance is indicated in the first column, and the number of aggregate activities formed from them in the second. An activity contained ca. 20 tasks on the average. However, only a part of the activities had to be performed within the planning horizon, since the requirement of minimal WIP level implied that activities had to be planned as near to the customer deadline as possible. The number of activities planned within the horizon of 15 weeks is shown in column *Planned*.

The results of detailed scheduling are presented in two parts, first for the entire set of tasks (*Total*), and then for the critical set of tasks, i.e., those that are executed on the week just before the customer deadline (*Critical*). The portion of delayed tasks was between 0.46% and 1.16% (0.15% and 2.22% for critical tasks). The average delay over the tasks with positive delay was 5-9 hours (3-8 hours for critical tasks), although the delay of some tasks reached up to 50 hours (18.6 hours for critical tasks).

Table 3: Planning and scheduling: characteristics of the integrated planning and scheduling problems solved.

Test case	Production planning			Detailed scheduling							
				Total				Critical			
	Tasks	Activities	Planned	Tasks	Delays	Average delay (h)	Max. delay (h)	Tasks	Delays	Average delay (h)	Max. delay (h)
#1	43213	2413	621	9277	0.72%	5.72	16.7	6035	0.31%	3.85	11.7
#2	42850	2358	753	12127	0.68%	6.28	25.3	7713	0.30%	3.99	12.1
#3	42140	2247	833	13196	0.67%	6.46	19.7	8554	0.32%	5.29	17.2
#4	38255	2074	826	13824	0.54%	6.11	17.6	5530	0.25%	3.89	13.6
#5	34637	1901	801	13120	0.46%	8.09	29.2	3294	0.15%	4.50	11.5
#6	32549	1756	855	14536	0.72%	8.37	45.4	4344	0.99%	5.77	12.8
#7	30249	1584	847	16024	0.52%	7.99	50.4	3930	0.76%	3.71	14.1
#8	27892	1397	770	15308	0.91%	9.00	45.4	3917	2.22%	8.23	18.6
#9	24271	1244	769	15346	0.48%	7.32	41.8	5871	0.27%	2.95	15.6
#10	21068	1073	720	14490	1.14%	7.64	35.0	6144	1.01%	5.67	14.8
#11	16412	871	702	14001	1.08%	5.33	22.5	5492	1.11%	4.36	10.3
#12	13956	726	722	13804	1.16%	5.04	20.9	7051	1.16%	3.95	8.5

Although the method we propose for integrating the two levels of the planning hierarchy does not theoretically guarantee that productions plans can always be unfolded directly into feasible, deadline respecting production schedules, results of our large-scale experiments suggest that the integrity of the two levels can be ensured. Hence, we are convinced that our results are competitive with the performance of current production planning methods. This holds especially in the light of the criterion of minimal WIP, which implies that tasks have to be performed close to the project deadline. Furthermore, some of the above delays can be eliminated by simple heuristics, e.g., by swapping tasks between neighboring weeks. We are planning to extend our system with such local improvement heuristics in the future.

4.3 Simulation results

4.3.1 Implementation of the simulation

The above enterprise produces mechanical products by using machining and welding resources, assembly and inspection stations and some highly specialized machines. Production is performed in a make-to-order manner where deadline observance is an absolute must, even regarding unpredicted orders. Since quality assurance is a key issue, tests may result in extra adjustment operations.

By taking the component-based approach, we have built a stochastic evaluation environment for testing the schedules generated by PROTERV-II (see Figure 11). This environment has been implemented by using the eM-Plant simulation framework. The *Decision-maker* initializes the simulation experiment manually (*Simulation Input*), by selecting the time horizon, the uncertainties, the confidence level as well as the main goals of the simulation study from a predefined set. The simulation system generates the required scenarios automatically and executes the simulation replications. Finally, the results of the experiment (*Simulation Output*) are interpreted to the *Decision-maker* either by using the GUI of the simulation software or by exporting to other common formats.

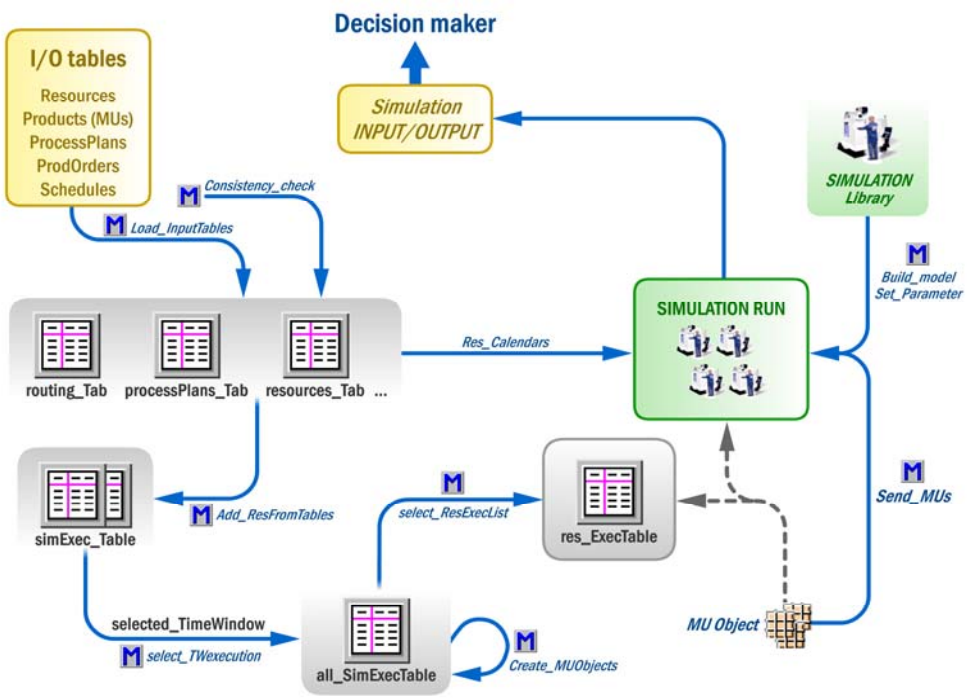

Figure 11: The simplified description of the process flow of data preparation and component-based model building realised in eM-Plant (phase *a* and phase *b*).

The four main criteria the simulation model has to cope with are as follows:

- During the simulation, only the *sequences* of the scheduled operations are considered, while, the calculated starting times are neglected because of the discrete-event-driven execution.
- All of the *operators* should return to the operators' pool when the shift ends. Operations not finished within the current shift should request new operator in the following shift. The reordering process of the operators to the unfinished operations is sequential.
- Each processing activity of a task requires at least one operator of a given type.
- It is possible that the processing time of the operator is shorter than the processing time of the machine, for the same operation.

The object-oriented hierarchical simulation model of the plant to be modelled is based on the functional decomposition approach. The simulation includes the modelled elements of the real plant and each unit of a production set is identified uniquely and traced during its lifecycle. In order to reduce the rigidity of the schedule during execution, the fixed start times of tasks are removed and only the sequence of the tasks on the various resources are kept. As a main principle, the simulator should play back the schedule only without changing the optimized sequence of the tasks.

4.3.2 Experimental results

The planning and scheduling methods, described in Section 2 were validated and tested with the real-life data. In the experiments, first projects were generated from existing routing tables and BOMs, then, using the resource calendars, the planning problem was solved on a 15-week horizon, with a time unit of one week. Next, the production plan was passed to the constraint-based finite job-shop scheduler. The shop-floor of the case-study includes more than 100 resources, all of which are modelled in the simulation module. The short-term schedule table contains approximately 1000 (max. 2000) tasks to be executed in one replication and the time frame of one simulation replication is one week. Statistical data were collected both on the resource and product sides.

In the case-study considered, the most important objective regarding the factory was the minimization of tardy jobs during the schedule execution. We considered the following performance measures: T_{mean} mean tardiness, T_{max} maximum tardiness, and n_{pt}, the number of tasks postponed to the next week. The basic types of *uncertainties* modelled in the simulation model are as follows:

- *downtimes*: due to failures or the unexpected absence of machines and/or operators;
- *processing time*: the actual processing time of some tasks may depend on the proficiency and skill of the operator; processing times may be shorter or longer than planned.

Table 4 demonstrates the results of the experiments after the execution of a predictive schedule in the simulation model including different uncertainty levels.

Table 4: Illustrative results of deterministic and stochastic schedule execution regarding one week (average values in hours, calculated from 10 simulation replications).

Applied *play-back* strategy	**Efficiency of new employees (75%)**			**Efficiency of employees (100%)**		
	T_{mean} (h)	T_{max} (h)	N_{pt}	T_{mean} (h)	T_{max} (h)	N_{pt}
Deterministic processes	3.25	16.39	15	0	0	0
95% machine availability and stoch. process times	5.73	27.20	45	5.24	18.65	27

Deterministic execution means that no uncertainty was set in the simulation. As expected, in this case the executed schedule is exactly the same as the planned one, and this way it serves as a benchmark for the stochastic scenarios. In the stochastic scenarios (row 2 in Table 4) the processing times of the tasks were set to a stochastic variable, where the lower bound is 90%, while the upper one is 130% of the planned process time. Correspondingly, machine availability was set to 95% in this scenario.

Major tasks on the shop floor consist of welding operations that depend highly on the skills of the operators. Experiments were carried out for evaluating different operator groups including operators with different skills. It is supposed that new operators are employed and their efficiency is lower during the “learning period”. Their overall effect on the planned schedule were investigated so that the total number of the operators in

each group (10 different operator groups were considered) were combined with the number of supposed new employees with skills under the standard skill level (for more details regarding the experimental design please refer to [25]). The two main columns in Table 4 highlight the variation of processing times deriving from the difference of the skills of operators (efficiency of the new employees were set to 75% of the regular operators' skills). Results were calculated from 20 different parameter settings, each with 10 replications.

Figure 12 shows the simulation results with the two pre-selected ratios (75% and 85%). In the x-axis (highlighted as 'Completed task') the tardiness is calculated for each task completed its processing. When evaluating the results, the Decision-maker is interested in the steepness, the breaks and steps of the curves (formulated by the plotted dots), as well as the distances between the curves. This information might be important also for the fine tuning of the scheduling parameters, but even more for the exact allocation of the operators (considering the skills) during the weekly execution of the predefined production schedule.

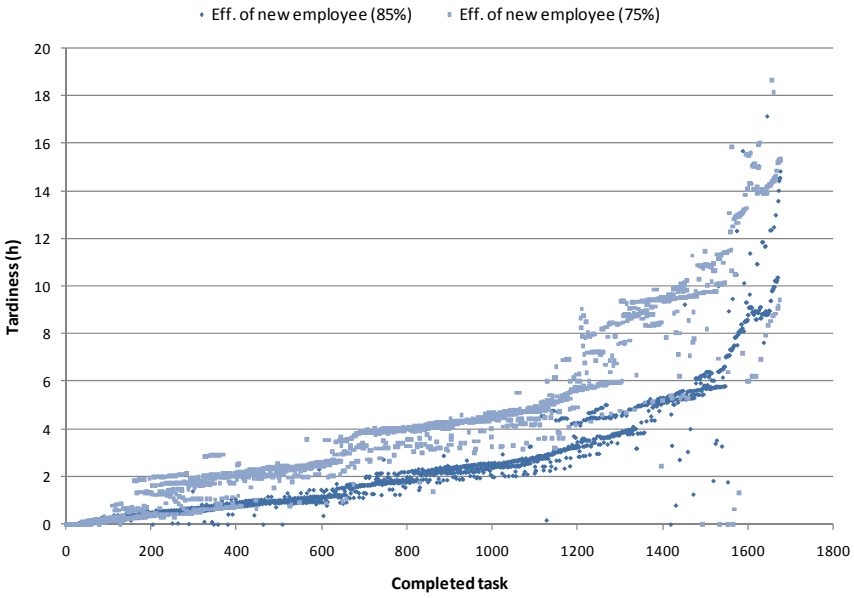

Figure 12: The possible effects of new operators with lower skills (operators' efficiency, %) on the tardiness of the completed tasks (schedule containing 1675 tasks, time horizon is one week).

Figure 13 shows the effects of both machine availability and processing time variance on average tardiness. Apart from the fact that the chart reinforces the prior expectations about the average tardiness, effect of input values from different interval sets can also be analysed together. The Decision-maker has the opportunity either to compare the maps of different scheduling settings (as a feedback, e.g. inserting slack time during scheduling), or to compare the evaluation results of different weeks with different system loads (e.g. number of tasks to be processed).

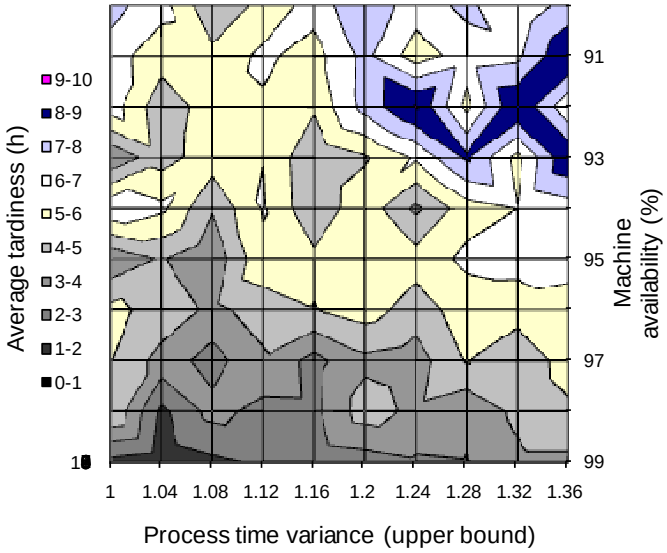

Figure 13: Response map: the dual effect of machine availability and processing time variance on average tardiness, for a selected production schedule (schedule containing 1675 tasks, time horizon is one week).

5 Conclusions

The main goal of the paper was to demonstrate how digital enterprise technologies combined with sophisticated optimization algorithms can contribute to planning and scheduling the operation of complex production systems in an efficient and consistent way. In an industry-initiated and involved project, novel models and algorithms were developed for integrated production planning and scheduling. As the results of computational experiments with two systems have shown, our original objectives could be attained. The main contributions are as follows:

- We have developed a novel project-based model for production planning in make-to-order and engineer-to-order industries. Planning with variable intensity activities and feeding precedence constraints enables to combine decisions about the timing of production activities and the utilization of resources. The model that can be amended with various objective functions results in more accurate production plans.

- We put special emphasis on developing efficient solution algorithms. On the level of planning, a new custom-tailored branch-and-cut algorithm solves the planning problem with the automatic generation of cutting planes that cut off the fractional solutions at the nodes of the search tree. The constraint-based scheduler was augmented with two methods that reveal – and exploit – typical but hidden structural properties of our industrial problem instances. Thanks to the efficient solvers, large-scale, real-life problem instances can also be solved, and decision makers are able to explore and evaluate a number of alternative future scenarios, too.

- In case of hierarchical planning and scheduling, the integrity of solutions can be ensured by an appropriate aggregation method that builds up the high-level planning model from detailed, *de facto* standard master data. The integration of planning with detailed scheduling and execution leads to a better due date observance and to more efficient use of resources. As a result, planners are capable of accepting more customer orders and reducing production costs.
- Our component-based simulation technology proved to be appropriate for assessing the sensitivity of deterministic solutions in face of typical uncertainties. Hence, results of the integrated production planning and scheduling system can be tested and further improved by a discrete-event simulation module which is able to analyze the effects of various types of changes and uncertainties related to operator skills and availability, machine breakdowns, and processing time variations. We emphasize that all the three models are built up automatically, from a common data store.

Finally, two possible connections of the integrated production planning and control approach introduced in the paper are to be outlined here. One is the *execution control* level [23] and the other is planning on the level of *production networks*. However, in any case, local powerful PPC methods that produce cost-efficient, executable and robust plans and schedules both on the medium and the short-term are prerequisites of these extensions.

Acknowledgements

We dedicate this paper to the memory of the late András Márkus who contributed many ideas presented here. Further on, we are also indebted to Ferenc Erdélyi, Tibor Tóth and Péter Egri for their advice and contributions to this work. This research has been supported by the Hungarian Scientific Research Fund (OTKA) grants “New mathematical models and methods for integrated production planning and scheduling” K-76810, and “Production Structures as Complex Adaptive Systems” T-73376.

References

[1] Aytug, H., Lawley, M.A., McKay, K., Mohan, S., Uzsoy, R.: Executing production schedules in the face of uncertainties: A review and some future directions, *European Journal of Operational Research* **161** (2005), pp. 86-117.

[2] Banks, J.: *Handbook of Simulation, Principles, Methodology, Advances, Application and Practice*. John Wiley & Sons Inc., 1998.

[3] Baptiste, Ph., Le Pape, C., Nuijten, W.: *Constraint-Based Scheduling*. Kluwer Academic Publishers, 2001.

[4] Bidot, J., Laborie, P., Beck, J.C., Vidal, T.: Using simulation for execution monitoring and on-line rescheduling with uncertain durations. In: *Proc. of the ICAPS'03 Workshop on Plan Execution*, Trento, Italy, (2003).

[5] Carlier, J., Pinson, E.: A practical use of Jackson's pre-emptive schedule for solving the job-shop problem. *Annals of Operations Research*, **26** (1990), pp. 269-287.

[6] Cowling, P., Johansson, M.: Using real time information for effective dynamic scheduling. *European Journal of Operational Research* **139** (2002), pp. 230-244.

[7] Demeulemeester E.L., Herroelen, W.S.: *Project Scheduling: A Research Handbook*. Kluwer Academic Publishers, 2002.

[8] Focacci, F., Lodi, A., and Milano, M.: Embedding relaxations in global constraints for solving TSP and TSPTW. *Annals of Mathematics and Artificial Intelligence* **34**(4) (2002) pp. 291-311.

[9] Hackman, S.T., Leachman, R.C.: An aggregate model of project-oriented production. *IEEE Transactions on Systems, Man, and Cybernetics*, **19**(2) (1989), pp. 220-231.

[10] Hans, E. W.: *Resource Loading by Branch-and-price Techniques*. Ph.D. thesis, Twente University Press, The Netherlands, 2001.

[11] Honkomp, S.J., Mockus, L., Reklaitis, G.V.: A framework for schedule evaluation with processing uncertainty. *Computers and Chemical Engineering* **23** (1999), pp. 595–609.

[12] Jensen, T.M.: Improving robustness and flexibility of tardiness and total flow-time job shops using robustness measures. *Applied Soft Computing* **1** (2001), pp. 35-32.

[13] Jünger, M., Reinelt, G., and Thienel, S.: Practical problem solving with cutting plane algorithms in combinatorial optimization. *DIMACS Series. in Discr. Math. and Theor. Comput. Sci.* **20** (1995), pp. 111-152.

[14] Kis, T.: A branch-and-cut algorithm for scheduling of projects with variable intensity activities. *Math. Prog.*, Ser. A **103** (2005), pp. 515–539.

[15] Kis, T.: RCPS with variable intensity activities and feeding precedence constraints, In: Józefowska, J., and Weglarz J (eds), *Perspectives in Modern Project Scheduling*, *International Series in Operations Research & Management Science*, Springer US, Vol. 92 (2006), pp. 105-129.

[16] Kovács, A., Kis, T.: Partitioning of trees for minimizing height and cardinality. *Information Processing Letters* **89**(4) (2004) pp. 181-185.

[17] Kovács, A., Váncza, J.: Progressive solutions: a simple but efficient dominance rule for practical RCPSP. In: *Proc. of CPAIOR 2006, the 3rd Int. Conf. on Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems* (Springer LNCS 3990), (2006), Cork, pp. 139-151.

[18] Kovács, A., Váncza, J.: Completable partial solutions in constraint programming and constraint-based scheduling. In: *Proc. of the 10th International Conference on Principles and Practice of Constraint Programming* (Springer LNCS 3258), (2004), pp. 332-346.

[19] Kempf, K., Uzsoy, R., Smith, S., Gary, K.: Evaluation and comparison of production schedules, *Computers in Industry* **42** (2000), pp. 203-220.

[20] Law, A., Kelton, D.: *Simulation Modelling and Analysis*. McGraw-Hill, 2000.

[21] Leachman, R. C., Dincerler, A., and Kim, S.: Resource-constrained scheduling of projects with variable-intensity activities, *IIE Transactions* **2**2(1) (1990), pp. 31-39.

[22] Maropoulos, P.G.: Digital enterprise technology – Defining perspectives and research priorities. In: *Proc. of the 1st CIRP (UK) Sem. on Digital Enterprise Technology (DET02)*, September 16-17, 2002, Durham, UK, Part V: 3-12.

[23] Monostori, L.; Kádár, B.; Pfeiffer, A.; Karnok, D.: Solution approaches to real-time control of customized mass production, *CIRP Annals - Manufacturing* Technology **56**(1) (2007), pp. 431-434.

[24] O'Reilly, J.J., Lilegdon, W.R.: Introduction to FACTOR/AIM. In: *Proc. of the 1999 Winter Simulation Conference*, (1999), pp. 201 – 207.

[25] Pfeiffer, A., Kádár, B., Monostori, L.: Stability-oriented evaluation of rescheduling strategies by using simulation. *Computers in Industry* **58**(7) (2007), pp. 630-643.

[26] Pfeiffer, A.: *Novel Methods for Decision Support in Production Planning and Control*, PhD Thesis, Budapest University of Technology and Economics, 2007.

[27] Pochet, Y., Wolsey, L.A.: *Production Planning by Mixed Integer Programming*. Springer, 2006.

[28] Rabelo, L., Helal, M., Jones, A., Min, J., Son, Y.J., Deshmukh, A.: A hybrid approach to manufacturing enterprise simulation. In: *Proc. of the 2003 Winter Simulation Conference*, (2003), pp. 1125-1133.

[29] Sabuncuoglu, I., Kizilisik, O.M.: Reactive scheduling in a dynamic and stochastic FMS environment. *International J. of Production Research* **41**(17) (2003), pp. 4211-4231.

[30] Tavares, L. V.: A review of the contribution of Operational Research to project management. *European Journal of Operational Research* **136** (2002), pp. 1-18.

[31] Váncza, J., Kis, T., Kovács, A.: Aggregation – the key to integrating production planning and scheduling. *CIRP Annals - Manufacturing Technology* **53**(1) (2004), pp. 377-380.

[32] Weglarz, J.: Project scheduling with discrete and continuous resources. *IEEE Trans. Systems, Man and Cybernetics* **9** (1979), pp. 644-650.

[33] Wullink, G.: *Resource Loading under Uncertainty*, Ph.D. Thesis, Twente University Press, The Netherlands, 2005.