MAK€ – A System for Modeling, Optimizing, and Analyzing Production in Small and Medium Enterprises

Roman Barták[1], Con Sheahan[2], Ann Sheahan[3]

[1] Charles University in Prague, Faculty of Mathematics and Physics
Malostranské náměstí 2/25, 118 00 Praha 1, Czech Republic
bartak@ktiml.mff.cuni.cz

[2] University of Limerick, Limerick, Ireland
Con.Sheahan@ul.ie

[3] ManOPT Systems Ltd., National Technology Park, Limerick, Ireland
Ann.Sheahan@manopt.com

Abstract. The paper presents a performance prediction and optimisation tool MAK€. This tool allows users to model enterprises in a visually rich and intuitive way. It is capable to automatically generate a scheduling model describing all choices the user can do when optimizing production. This model goes to the Optimiser Module that generates schedules optimizing on-time-in-full performance criterion while meeting the constraints of the firm and the customer demand. This module is implemented using constraint-based solving techniques with specific search heuristics for this type of problems. Finally, the Performance Manager Module shows the decision maker what is the best possible outcome for the firm given the inputs from the Enterprise Modeller. The system demonstrates capabilities of constraint-based scheduling that is one of the killing application areas of constraint programming.

Keywords: production scheduling, problem modelling, optimization, analysis

1 Introduction

Though there exists a vast amount of research in the area of scheduling there is still a large gap between practical problems and research results especially in the area of production optimization for small and medium enterprises. This gap is partly due to missing modeling and visualization tools that would allow easy transformation of real-life problems to optimization models and the results back to customers [8] and partly due to large distance of academic algorithms from the existing problems. MAK€ tool developed by ManOPT Systems addresses the above gaps by providing a streamlined feature rich environment where the user could do all of the following in a simple, efficient and user-friendly way:

- Specify how a particular product is manufactured (i.e. define a workflow describing the manufacturing of a single product).

- Enter a work order from a customer – customers request certain quantities of products that the factory can manufacture, together with a desired deadline.
- Generate a schedule for the order – a schedule should be a complete description of what elementary activities should be performed, in what exact times should they run and what resources should they use (machines or people). Executing such a schedule should result in efficient fulfilling of the work order.
- Display the generated schedule in the form of a Gantt chart and modify it interactively.
- Analyze the generated schedule for possible opportunities of improvement.

The MAK€ tool is a result EU funded projects EMPOSME and ValuePOLE that gave a unique opportunity for co-operation between academia and industry as usually researchers and final customers are too far from each other to communicate directly the needs on one side and the possibilities on the other side. In the project we demonstrate the recent research advancements in the areas of formal problem modeling and solving, namely using Temporal Networks with Alternatives [5] to describe workflows, in real-life industrial setting. The goal was to make optimization technology accessible to practitioners without any knowledge of optimization techniques. Constraint Programming techniques were used in the scheduling engine also called optimizer and this paper focus on this part of the MAK€ tool. Nevertheless, we will try to give a broader picture of the area to show that there is a long path from the customer to the optimizer where many important decisions must be done before the formal is presented to the optimizer to generate a schedule.

2 The Problem

Frequently, when people are applying Constraint Programming techniques to real-life problems, they focus on solving a particular problem for which they formulate (manually) a constraint model and then fine tune the model to achieve acceptable performance for a given sort of data [13,16]. This is not the case of MAK€ project which is intended to provide a tool for production optimization in small and medium enterprises in general. It means that there is no single problem to be solved but rather a collection of problems with a common core – production of certain pieces of items that are manufactured, assembled, packed and delivered to a customer. It means that the system collects data about the particular enterprise in the form common for enterprise systems; it generates an optimization model fully automatically; it solves the optimisation problem, it shows the result to the user, and finally it supports analysis of the result. Obviously, there are many optimization problems appearing in enterprises starting with optimizing layout of the factory, minimizing waste production and energy consumption, optimizing resource utilization and others. In MAK€ we focus on optimizing the production schedule.

There exists a huge number of various scheduling problems [10] with many algorithms to solve the particular classes of problems. In the MAK€ system, the scheduling problem is not described explicitly by the user, but it is rather derived automatically from data about enterprise provided by the user. These data consists of description of bill of materials, workflows, resources, and custom orders. *Bill of materials* (BOM) can be seen as a hierarchical structure (tree) with the final product on top (in the root) and items from which this product is composed below. BOM determines how the final product is assembled from its component parts. For example to produce a table we need a main board and four legs where the main board consists of the board itself and an attached drawer. Notice that BOM describes not only the structure of the product but also the quantities of required components. BOM is accompanied by the description of manufacturing routes which describe *workflows* for each basic part. Both bill of materials and workflows define temporal constraints between individual manufacturing operations which can be for example expressed by the minimal time distance between two operations. For each operation, there are one or more *machines, tools, operators*, etc. which are required to carry it out. These resources have assigned availability calendars and performance characteristics that can be used to pre-compute durations of operations (together with information about quantity of products from BOM). Bill of materials, workflows, and resources describe the enterprise, but we need custom orders to dictate what needs to be manufactured. Each *order* specifies the ordered item, its quantity, and the delivery date. The major objective that we focus on is *on-time-in-full* delivery. It means that the goal of optimizer is producing a schedule where the ordered products are ready to ship at requested dates. Notice that the input data are not in the format of a typical scheduling problem so when formalising the scheduling problem to be solved, we need to take in account all possible scenarios that can be obtained from input data.

The closest formalism for the underlying scheduling problem is probably the Extended Resource Constrained Project Scheduling Problem [15]. We describe the scheduling problem as a set of non-interruptible activities, where each activity has a fixed duration and has assigned a set of unary resources to which the activity must be allocated if the activity is decided to be part of the schedule. The activities are connected in a temporal network with alternatives [5] that describes the workflows. Temporal network with alternatives (TNA) is basically a directed acyclic graph where nodes correspond to activities and arcs are annotated by simple temporal constraints specifying the minimal and maximal time distance between the start times of activities (see Figure 1). The main difference from traditional temporal networks is specifying the *branching constraints*. If there are more arcs going from the node then either parallel or alternative branching is specified in TNA. The type of branching describes the flow of the process. The *parallel branching* means that either all activities (the activity and all its direct successors) are present in the solution or no activity is present (see *collectMaterial* in Figure 1). The constraint imposed by the parallel branching is also assumed in the case of having exactly one successor. The *alternative branching* means that either the activity together with exactly one of its direct successors (predecessors) in a TNA is present in the solution or no activity is present (see *weldTube* and its predecessors in Figure 1). This is a way to describe splitting of the process into several alternatives. Note also, that the same mechanism can be used to model alternative resources (the process splits into alternative activities and each of the activities is allocated to one of the alternative resources). Similarly, we describe alternative or parallel joining of processes if there are more arcs going into a node. We may also assume *auxiliary activities* with zero duration and no resource requirements in the TNA to model auxiliary time points in TNA (see top left node in Figure 1). Using alternatives in TNA is the main difference from the traditional scheduling problems as the scheduling task now includes a selection of activities [9] together with their allocation to time while respecting the temporal, branching, and resource constraints. To state it differently: the task is to select a subset of activities that satisfies the branching constraints (this was called a P/A graph assignment problem in [5]) and decide the start time of each selected activity in such a way that the temporal constraints from TNA are satisfied and selected activities allocated to the same resource do not overlap in time. This is a form of integrated planning (selection of activities) and scheduling (allocation of activities) problem.

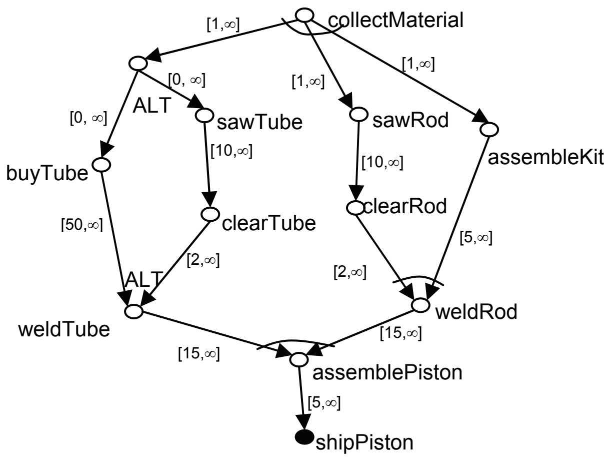

Fig. 1. A manufacturing process with alternatives (the alternative branching is marked by ALT, the other branchings are parallel; arcs are annotated by simple temporal constraints expressing minimal and maximal distance in time).

Note that selecting a consistent subset of nodes from a general TNA where some nodes are preselected (a P/A graph assignment problem) is an NP-complete problem [5]. Fortunately, real-life workflows frequently have a specific structure [1], which we call a *nested TNA*, where the P/A graph assignment problem is tractable [6]. Nested structure means that the network is obtained by a decomposition process starting with single arc and decomposing any arc into a set of arcs and nodes called a nest as Figure 2 shows (TNA in Figure 1 is also nested). The bad news is that adding simple temporal constraints [14] to nested TNA makes the problem NP-complete again [7]. What makes the problem hard is using the "deadline" constraints, in terms of TNA it means using the maximal distance between the activities. In practice, users usually specify only the minimal distance constraints though there exist industries, such as metallurgy, where maximal distance between activities is important.

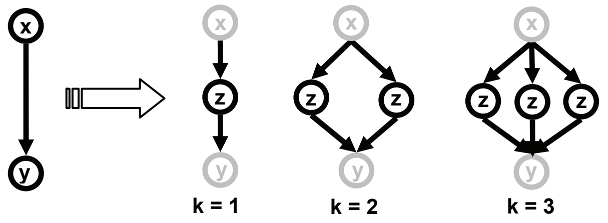

Fig. 2. Arc decompositions in nested graphs.

So far we focused on the feasibility problem, that is, formalizing the constraints to be satisfied by the schedule. As we mentioned at the beginning we are in fact solving an optimization problem with an on-time-in-full objective. This objective is reflected in the formalization of the problem as follows. Some activities, typically those describing the delivery operation, have assigned a due date with earliness and lateness costs. Assume that E is the earliness cost, L is the lateness cost, D is due date and T is the scheduled time when the activity finishes. Then the cost of the activity is E.max(D-T,0) + L.max(T-D,0). Now the task is to find a feasible schedule minimizing the sum of costs of all the selected activities.

In summary there are two main issues to be solved when integrating the optimiser into a MAK€ tool. First, it is necessary to generate the scheduling model from existing data. It means converting the bill of materials, workflow descriptions, resources, and demands to scheduling concepts such as activities, temporal and resource constraints, and objectives and converting the schedule back to the enterprise model. We solved this problem by defining translation rules that automatically transform the notions and related data between the enterprise system and the scheduling engine [8]. Though the translation may seem easy (which is really not the case if real data are assumed), it is important to highlight that users do not specify the scheduling problem and the problem is generated automatically. The second issue is solving the scheduling problem itself and this is where Constraint Programming was used.

3 Why CP?

The first question one needs to ask is why the users should use new optimisation software. According to our experience the reason is never because the users want to try a new technology. This claim holds across all business areas including the most innovative ones such as space exploration [18]. It is almost impossible to go to a company with an offer of a new and theoretically better system and get the contract. There must be internal necessity inside the company that pushes the company to implement a new technology. Typically, this necessity goes from the outside of the company, from the competitors. Currently, there is an increased competition in the area of mass production from Far East. Big multi-national companies solve this problem by moving their production facilities to areas with low labour cost which helps them to decrease the cost of their products but sometimes with the trade-off of lower quality. This strategy is not applicable to small and medium enterprises (SMEs) that are typically family-owned and closely connected to their area of origin. Such companies can compete by providing high quality products that are built-to-order or even engineered-to-order so the products fit exactly the specific needs of their customers. However, this brings huge variety of produced items and increases complexity of work organization which implies that traditional scheduling methods (in these companies) based on Excel sheets and manual production of work plans are less applicable. There is also increased demand for decreasing production cost by optimizing production processes and utilisation of resources. Existing enterprise resource planning tools are less applicable for SMEs as they are too expensive, too complex, and too hard to customize for SMEs so there is a need for new tools that fit well the requirements of SMEs. MAK€ tool has already been available for several years and it is used in a couple of manufacturing companies. It used a heuristic scheduling algorithm that was capable to automatically generate schedules, but it was less flexible and too connected to exiting scheduling practice (using preferred process routes) rather than providing alternative and justified solutions based on optimisation. This is where Constraint Programming enters the scene.

Scheduling is definitely not a new application area to Constraint Programming [2] and it some sense, constraint-based scheduling is the prominent and the most influential application area for CP with many successful applications [3,13,16]. The reason could be the flexibility of constraint models that allows adding side constraints appearing in real-life problems without big problem. Another reason is a support for specific scheduling constraints in existing constraint satisfaction packages [16,17]. Many specific global constraints for example for describing resources have been proposed in recent years [2,17,20] which simplifies significantly problem modelling and brings advanced scheduling techniques to fingertips of regular CP users. The MAK€ tool brings this technology further to practitioners which are even not familiar with the optimization techniques. The reasons to use Constraint Programming in MAK€ tool can be summarized as:

- *flexibility*, it is easy to modify the formal model by side constraints
- *efficiency*, it is possible to integrate specific solving techniques as additional inference (constraint propagation)
- *customization*, it is easy to use search heuristics derived from the problem specification
- *familiarity*, it is a technology that developer is familiar with.

4 How CP?

As specified in the previous section, we decided for a pure CP approach to solve the problem. Thought hybrid techniques are very popular and can solve some problems faster, they are also harder to implement and maintain as they require expertise from more areas. Basically, we believe that a pure solving approach is fine when the produced solutions are satisfactory which seems to be the case of the MAK€ tool. Moreover, CP technology provides enough flexibility to integrate specific inference techniques and search heuristics which seems crucial for the types of problems that we are solving. In this section we will describe the core ideas of the CP model with some specific inference techniques and search strategy.

The constraint model and search strategy were implemented in SICStus Prolog 4. We used the unary resource constraint `disjoint1`, arithmetic and logical constraints from `clpfd` library of SICStus Prolog and we implemented special inference procedures (see below) for temporal constraints using the global constraints interface in SICStus Prolog.

4.1 The core model

For the MAK€ tool we decided for a "light" constraint model with only two types of variables (start time and validity) and a few types of constraints (integrated branching and temporal constraints, unary resource constraints, and some auxiliary constraints). This is a big difference from our previous scheduling approach used in Visopt ShopFloor system [3] which used a very complex dynamic constraint model. The main reason for the light model was fast development and easy maintenance. This gives us opportunity to focus on core features and implement them as efficiently as possible and to explore more alternatives how to implement the core constraints and search strategies. On the other hand, this approach requires some real-life features to be compiled to this light model. This is done during the translation of the enterprise model to the scheduling model. For example, the availability calendar is modelled by auxiliary activities that are fixed in time and occupy the resource when the resource is not available. In the rest of this section, we will describe the light constraint model.

The traditional scheduling features are modeled in a standard way. For each activity we have a start time and duration variables. The domain of start time variable is defined by the time window for a given activity. The domain of duration variable consists of two values: zero and the constant processing time of the activity specified in the problem formulation. These variables participate in the *resource constraint* describing the unary resource to which the activity is allocated (as we mentioned we use `disjoint1` from SICStus Prolog). Using zero duration is a method to describe optional activities in traditional resource constraints that do not support optional activities [2,20]. If duration is zero then the activity is ignored in the unary resource constraint and vice versa the resource constraint may also set duration to zero if there is not enough capacity to process the activity.

To model selection of activities we use a Boolean validity variable for each activity. This variable is assigned to 1 (*true*) if the activity is selected (then also the duration variable is non-zero for real activities) and to 0 (*false*) if the activity is not selected (then the duration variable is zero). The duration variable can model selection of activities, but we use the validity variable for simplicity reasons and also to cover auxiliary (milestone) activities, which have zero duration by definition (but could be selected to the schedule). For real activities the relation between duration and validity variables is described using the constraint:

$$Val_A = 1 \Leftrightarrow Dur_A > 0$$

The validity variable together with the start time variable participates in the constraints describing the temporal network with alternatives. It is possible to use a straightforward model of branching constraints in the following form. If activity B follows directly activity A in some parallel branching then the constraint is:

$$Val_A = Val_B.$$

If *Branch* is a set of direct successors (predecessors) of activity A in alternative branching then the branching constraint can be described using arithmetic formula:

$$Val_A = \Sigma_{B \in Branch}\, Val_B.$$

Note that the above model for branching constraints does not achieve global consistency when alternative branching is present even if the TNA is nested. Assume constraints A = 1, A = B + C, B + C = D over the Boolean variables, which describes a simple nest with two alternative nodes B and C. Obviously D = 1, but standard arc consistency techniques cannot infer this information. Adding a redundant constraint A = D improves inference there, actually this constraint can substitute constraint B + C = D. In [6] we showed how such a constraint model for nested graphs can be automatically generated. We are however not using these redundant constraints in the current version of MAK€ tool because there is no assumption about the workflows to be nested and detecting the nested structure is an expensive process for large general graphs.

Recall that the temporal relation between the start times of activities *A* and *B* is described by a pair $[a_{A,B}, b_{A,B}]$ representing minimal and maximal distance. This relation can be naturally represented using the following constraint (we assume that 0 is the schedule start point):

$$Val_A * Val_B * (Start_A + a_{A,B}) \leq Start_B \;\wedge\; Var_A * Var_B * (Start_B - b_{A,B}) \leq Start_A.$$

The above straightforward model is appropriate if there is only a small number of alternative branchings. For such problems, values of most validity variables are known and the constraints propagate well. However, if the number of alternative branches increases then the straightforward model contains many (hidden) disjunctions that do not propagate well in most constraint solvers. In particular, notice that the lower bound of start time variables never increases until both validity variables are set to 1. In such a situation a more *pro-active approach* to domain filtering is better. The pro-active model always filters out infeasible time points from the start time variable (even if the activity is not yet known to be in the solution) and when the domain of the start time variable is to become empty then the filtering algorithm sets the corresponding validity variable to 0 rather than emptying the domain of the start time variable and generating a failure (empty domain in constraint satisfaction means a failure which causes backtracking in the search procedure).

We will describe now the filtering rules that propagate changes of domains between the constrained variables, namely, the values that violate the constraint are removed from the domains. Let $d(x)$ be the domain of variable x, that is, a set of values, and for sets A and B, $A \bullet B = \{a \bullet b \mid a \in A \wedge b \in B\}$ for any binary operation $\bullet$ such as + or –. Assume that arc (i, j) is a part of a parallel branching, so in the solution either both nodes i and j are valid and the temporal relation must hold, or both nodes are invalid and the temporal relation does not play any role (the domains of temporal variables are irrelevant provided that they are non-empty). Hence, we can always propagate the temporal relation provided that we properly handle its violation. Let $UP = d(Start_j) \cap (d(Start_i) + \langle a_{i,j}, b_{i,j}\rangle)$. The following filtering rule is applied whenever $d(Start_i)$ changes:

$$\begin{array}{ll} d(Start_j) \leftarrow UP & \text{if } UP \neq \varnothing \\ d(Val_j) \leftarrow d(Val_j) \cap \{0\} & \text{if } UP = \varnothing. \end{array}$$

Note that $UP = \varnothing$ means violation of the temporal relation which is accepted only if the nodes are invalid. If the nodes are valid then a failure is generated because the above rule makes the domain of the validity variable empty. Symmetrically, let $DOWN = d(Start_i) \cap (d(Start_j) - \langle a_{i,j}, b_{i,j}\rangle)$. The following filtering rule is applied whenever $d(Start_j)$ changes:

$$\begin{array}{ll} d(Start_i) \leftarrow DOWN & \text{if } DOWN \neq \varnothing \\ d(Val_i) \leftarrow d(Val_i) \cap \{0\} & \text{if } DOWN = \varnothing. \end{array}$$

The following example demonstrates the effect of above filtering rules. Assume that the initial domain of temporal variables is $\langle 0, 70\rangle$, the validity of nodes is not yet decided, and there are arcs (i, j) and (j, k) with temporal constraints [10, 30] and [20, 20] respectively. The original constraints do not prune any domain, while our extended filtering rules set the domains of temporal variables $Start_i$, $Start_j$, and $Start_k$ to $\langle 0, 40\rangle$, $\langle 10, 50\rangle$, and $\langle 30, 70\rangle$ respectively. If the initial domain is $\langle 0, 20\rangle$ then the original constraints again prune nothing, while our extended filtering rules deduce that the participating nodes are invalid (we assume that logical constraints in the form $Val_x = Val_y$ are also present).

The propagation of temporal constraints in the alternative branching is more complicated because we do not know which arc is used in the solution. Therefore, the filtering rule uses a union of pruned domains proposed by individual arcs (from non-invalid nodes) which is similar to constructive disjunction of constraints. Let x be the principal node of a fan-in alternative sub-graph and $y_1,\ldots, y_k$ be all branching nodes. We first show how domains of the branching nodes are propagated to the principal node. Let $UP = d(Start_x) \cap \cup_{j=1,\ldots,k}\{(d(Start_{y_j}) + \langle a_{y_j,x}, b_{y_j,x}\rangle) \mid d(Val_{y_j}) \neq \{0\}\}$. The following filtering rule is applied whenever any $d(Start_{y_j})$ or $d(Val_{y_j})$ changes:

$$\begin{array}{ll} d(Start_x) \leftarrow UP & \text{if } UP \neq \varnothing \\ d(Val_x) \leftarrow d(Val_x) \cap \{0\} & \text{if } UP = \varnothing. \end{array}$$

It may happen that set UP is not an interval but a set of intervals. Then we may use an interval hull which makes filtering less time and space consuming but a smaller number of inconsistent values is filtered out. The propagation from $d(Start_x)$ to $d(Start_{y_j})$ is done exactly like the *DOWN* propagation described above and similar filtering rules can be designed for fan-out alternative sub-graphs. The properties of the pro-active filtering procedure are described in [7]. In the constraint solver, the pro-active filtering is realized as inference for the global constraint over the validity and temporal variables. The possibility of integrate such special inference procedures to the underlying constraint solver without influencing the rest of the model is one of important advantages of CP technology.

4.2 Search strategy

There are two types of decision variables in the constraint model, the start time variables and the validity variables. The duration variables are auxiliary and their value is fully determined by the value of validity variables. Hence the search strategy focuses on selection of activities (assigning the validity variables) and time allocation (assigning the start time variables). Note that selecting the activities also means deciding the alternative resources, if they are defined. This is one of the consequences of the light constraint model – we can focus on the selection of alternative process routes and time allocation only and still cover features such as resource allocation.

Rather than using generic labeling procedures available in constraint solvers, we decided to implement own search strategy driven by the objective function. The search procedure consists of two stages. In the first stage we select the activities to form the schedule and we decide their order if they share a resource. In the second stage we decide the particular start times. The first stage can be seen as a generalization of *Precedence Constraint Posting* (PCP) scheduling strategy [11] while the second stage is a known *Set a Start Time* (SST) strategy [12]. The ideas how to extend the PCP to support optional activities is described in [4]. In MAK€ we are using a similar technique but the technical details are of commercial value so they cannot be revealed. Briefly speaking, we are doing left-to-right scheduling where we are ordering the activities from left (earlier times) to right (later times). We select the activity based on its minimal start time, its slack [19] and validity status, and its recommended start time derived from the due date. For the selected activity, the validity variable is set to 1 and the precedence constraints are posted between this activity and all not-yet scheduled activities that share a resource with the currently selected activity. If a failure is detected (via inference or later during search), the selected activity is known not be the first one so its minimal start time is increased and the activity is remembered to be after some other valid activity that will be selected later during search. The last option, if postponing the activity also fails, is setting the validity status of the activity to zero which means that the activity is not part of the solution (this is frequently done via inference if alternative to this activity is selected). After the first stage successfully finishes, we obtain a set of valid (selected to the schedule) activities connected via a simple temporal network [14]. Solving STN is tractable, but recall that we are optimizing the on-time-in-full criterion. Hence we order the activities based on their earliness and lateness costs and try to allocate them to their due date, if possible. More precisely, assume that we take an activity with the largest lateness cost parameter and due date D. Then we post a constraint that the activity finishes before or at D (then the lateness cost is zero). The alternative branch during search uses a constraint that activity finishes after D. The same mechanism is used for the earliness cost. The activities are selected based on the largest lateness or earliness cost parameters and depending on which type of cost is the largest one we use one of the above two branching schemes. After allocating the activities with costs we instantiate the start times of all other activities to their minimal possible values. Thanks to strong inference of temporal constraints, the second stage will never fail and will always produce a solution.

The above search procedure looks like a greedy search but it allows exploration of alternatives in case of failure. The failure may occur only if there are some hard deadlines (or maximal distance between some activities) otherwise it is always possible to postpone the activities with the penalty of increased cost of the solution. The search procedure can also be encapsulated in a standard branch and bound procedure where first a feasible solution is found and then we look for a better solution. However, experiments will real data showed that the first found solution has acceptable quality.

5 Added Value of CP

There have been many attempts to enhance the economic performance of firms with scheduling tools. Most have failed due to their limited ability to represent the typical SME where there are a large diversity of workflows and resource alternatives being applied to a very large diversity of products with tight delivery time lines. In the case of the MAK€ a key contribution of the CP approach was the ability to effectively represent this complex problem as one unified model. Once this initial representation was defined the CP approach supported an iterative development of the MAK€ tool where the validation the proposed scheduling solutions was completed on actual production problems from the SME end user partners. The structure of these actual problems tend to be quite different from the synthetic benchmark problems sets common in the literature. The end users in particular made an important contribution to the refinement of the objective function so that the CP search was focused on finding solutions that were of greatest practical significance to the SME partner firms involved rather than the more traditional makespan type objective functions. The MAK€ application has had a total of 15 years of use in five actual differing production facilities with no change in the model being applied to each facility. This general model of enterprises exploits the expressive properties of CP and has dramatically reduced the level of customer model development required for each new implementation. This has dramatically reduced the cost the MAK€ tool development and deployment. This has enabled the allocation of resources to creating a GUI that more closely meets the decision support needs of practitioners. The end users are presented with the consequences of the proposed scheduling solutions on a suite of Key Performance Indicators (KPIs) for the enterprise. These are the focus of the practitioners when the MAK€ tool is being used in production mode. In production mode a critical requirement of the application end users was that the scheduling solutions were generated quickly and could not be improved with manual interventions. The end user population had minimal interest in understanding the technology used to generate the schedule solutions. The ability of the decision makers to comprehensively evaluate the proposed scheduling solutions via a schedule analyser proved to be critical in establishing the credibility of the MAK€ tool with the end users. This method of presenting the proposed scheduling solution proved to be more important to end users than arguments based on optimality proofs being achieved for large populations of representative problem sets.

The payback for each facility has been very rapid since the workflow preparation effort is modest and required only once of for each product whereas the MAK€ can be run 50-60 times per day in production mode. The typical pay back for MAK€ is less than six months since the KPI focus of the application is aligned with the scheduling solutions being proposed. This is far superior to the current state of the art ERP and manufacturing execution systems.

Acknowledgments. The research is supported by the Czech Science Foundation under the contract P202/10/1188 and by EU Funding Scheme Research for the benefit of SMEs: FP7-SME-2007-1 under the project ValuePOLE (contract 222218).

References

1. Bae, J., Bae, H., Kang, S.-H., Kim, Z.: Automatic Control of Workflow Processes Using ECA Rules. In IEEE Transactions on Knowledge and Data Engineering, vol. 16, no. 8, pp. 1010-1023 (2004)
2. Baptiste, P.; Le Pape, C.; Nuijten, W.: *Constraint-based Scheduling: Applying Constraints to Scheduling Problems*. Kluwer Academic Publishers, Dordrecht (2001)
3. Barták, R.: Visopt ShopFloor: On the edge of planning and scheduling. *Proceedings of CP2002*, LNCS 2470, pp. 587-602, Springer Verlag (2002)
4. Barták, R.: Search Strategies for Scheduling Problems with Optional Activities. *Proceedings of CSCLP 2008 Annual ERCIM Workshop on Constraint Solving and Constraint Logic Programming*, Rome (2008)
5. Barták, R. and Čepek, O.: Temporal Networks with Alternatives: Complexity and Model. *Proceedings of the Twentieth International Florida AI Research Society Conference (FLAIRS)*, pp. 641–646, AAAI Press (2007)
6. Barták, R. and Čepek, O.: Nested Temporal Networks with Alternatives: Recognition, Tractability, and Models. *Artificial Intelligence: Methodology, Systems, and Applications (AIMSA 2008)*. LNAI 5253, pp. 235-246, Springer Verlag (2008).
7. Barták, R.; Čepek, O.; Hejna, M.: Temporal Reasoning in Nested Temporal Networks with Alternatives. *Recent Advances in Constraints*, LNAI 5129, pp. 17-31, Springer-Verlag (2008)
8. Barták R.; Little J.; Manzano O.; and Sheahan C. From Enterprise Models to Scheduling Models: Bridging the Gap. *Journal of Intelligent Manufacturing,* Volume 21, Number 1, pp. 121–132, Springer Verlag (2010)
9. Beck, J. Ch. and Fox, M. S. Constraint-directed techniques for scheduling alternative activities. *Artificial Intelligence* 121, 211–250 (2000)
10. Brucker, P. (2001). *Scheduling algoritms*. Springer Verlag.
11. Cesta, A.; Oddi, A.; and Smith S.F. Iterative Flattening: A Scalable Method for Solving Multi-Capacity Scheduling Problems. In *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 742-747, AAAI Press (2000)
12. Godard, D.; Laborie, P.; and Nuijten W. Randomized Large Neighborhood Search for Cumulative Scheduling. *Proceedings of the 15th International Conference of Automated Planning and Scheduling (ICAPS)*, pp. 81-89. AAAI Press (2005)
13. Delgado, A.; Møller Jensen, R.; Schulte, Ch.: Generating Optimal Stowage Plans for Container Vessel Bays. *Principles and Practice of Constraint Programming - CP 2009*, LNCS 5732, pp. 6-20 (2009)
14. Dechter, R.; Meiri, I.; Pearl, J.: Temporal Constraint Networks, *Artificial Intelligence* 49, pp. 61–95 (1991)

15. Kuster, J.; Jannach, D.; and Friedrich, G. Handling Alternative Activities in Resource-Constrained Project Scheduling Problems. In *Proceedings of Twentieth International Joint Conference on Artificial Intelligence (IJCAI-07)*, pp. 1960–1965 (2007)
16. Laborie P. IBM ILOG CP Optimizer for Detailed Scheduling Illustrated on Three Problems. *Integration of AI and OR Techniques in Constraint Programming for Combinatorial Optimization Problems*. LNCS 5546, pp. 148-162, Springer Verlag (2009)
17. Laborie P. and Rogerie, J. Reasoning with Conditional Time-intervals. In *Proceedings of the Twenty-First International FLAIRS Conference*, pp. 555-560, AAAI Press (2008)
18. Rabenau, E., Donati, A., Denis, M., Policella, N., Schulster, J., Cesta, A., Cortellessa, G., Fratini, S. and Oddi, A.: The RAXEM Tool on Mars Express - Uplink Planning Optimisation and Scheduling Using AI Constraint Resolution. In *SpaceOps-08. Proceedings of the 10th International Conference on Space Operations*, Heidelberg, Germany, May 12-16, 2008
19. Smith, S.F. and Cheng, Ch.-Ch. Slack-Based Heuristics For Constraint Satisfaction Scheduling. *Proceedings of the National Conference on Artificial Intelligence (AAAI)*, pp. 139-144. AAAI Press (1993)
20. Vilím, P.; Barták, R.; Čepek, O. Extension of O(n log n) filtering algorithms for the unary resource constraint to optional activities. *Constraints*, Volume 10, Number 4, pp. 403-425. Springer Verlag (2005)