Extended version of the paper to be presented at PACLP2000, Manchester, UK

Mixing Planning and Scheduling to Model Complex Process Environments

Roman Barták[*]

Charles University, Prague, Czech Republic
`bartak@kti.mff.cuni.cz`
`http://kti.mff.cuni.cz/~bartak`

Abstract

In most current APS (Advanced Planning and Scheduling) systems the planning and scheduling tasks are handled separately using different methods and technology from the areas like Artificial Intelligence and Operations Research. Recently, Constraint Programming becomes a roof over several solving technologies that allow us to solve both planning and scheduling tasks within single declarative framework.

In the paper, we suggest mixing planning and scheduling tasks within single system that is capable to solve more complicated scheduling problems in complex process environments. We analyse different views of planning and scheduling, identify the similarities and the differences of both tasks and we propose a general structure of scheduler with some planning capabilities. In the second part of the paper, we compare various modelling approaches from the mixed planning and scheduling point of view. We concentrate on capabilities of the models to capture typical problems in complex process environments primarily but the conclusions are applicable to other (non-process) problem areas. The presented results make the basics of a generic scheduling engine that is currently implemented within the VisOpt project.

1 Introduction

Planning and scheduling attract a high attention of computer science community because of their real-life applicability and challenging complexity. However, despite of their similar character, planning and scheduling problems are usually handled independently using different methods and technologies.

Roughly speaking, *planning* deals with finding plans to achieve some goal. More precisely, a planning task is defined as finding a sequence of actions that will transfer the initial world into one in which the goal description is true (Pool et all, 1998). Naturally, the possible sequences of actions are restricted by constraints describing the limitations of the world.

Opposite to planning, *scheduling* deals with the exact allocation of resources to activities over time, i.e., finding a resource that will process the activity and finding the time of processing (Brusoni et all, 1996). Again, the scheduler must respect several constraints like the precedence, duration, capacity and incompatibility constraints.

In the industry, the border between planning and scheduling tasks is moved to a different level and it becomes a little bit fuzzy. Also, the main difference between traditional planning and scheduling, i.e., the generation of activities in planning vs. assigning of activities

[*] Supported by the Grant Agency of the Czech Republic under the contract no. 201/99/D057.

to resources and time in scheduling, is suppressed here. Both industrial planning and scheduling deal with the task of finding a sequence of activities to achieve some goal and assigning these activities to resources. The main difference is in the resolution of the resulting plan or schedule. While the industrial planning deals with the task of finding “rough” plans for longer period of time where activities are assigned to departments etc., the industrial scheduling deals with the task of finding detail schedules for individual machines for shorter period of time. From this point of view, scheduling can be seen as a high-resolution short-term planning. The similarity of industrial planning and scheduling brings us to the idea of using a mixed approach that can be applied to both areas.

While Operations Research (OR) has a long tradition in studying scheduling problems and many successful methods to deal with the problem were developed there, the planning problems are usually solved using Artificial Intelligence (AI) technology. Recently, Constraint Programming (CP) attracts high interest among scheduling community because of its potential for declarative description of problems with complicated real-life constraints. Also, the potential of CP becomes uncovered in solving the planning problems as well. Moreover, CP can exploit the checked methods from OR and AI to improve the efficiency of constraint systems using global constraints and more advanced search techniques.

Constraint programming (Bartak, 1997) is based on the idea of describing the problem declaratively by means of constraints, logical relations among several unknowns (or variables). After stating the constraints, the solution, i.e., an assignment of a value to each unknown from respective domain, is being found in such a way that all the constraints are satisfied. It is possible to state constraints over various domains, however, currently probably more than 95% of all constraint applications deal with finite domains (Tsang, 1995). And among them, the scheduling problems are the most successful application area (Wallace, 1994).

In the paper we propose to mix both planning and scheduling tasks into single system using the CP framework. In particular, we suggest extending the traditional scheduler, i.e., the allocation of activities to available resources over time, by planning capabilities, i.e., by run-time generation of activities. Some features of complex process environments, where the activities are not known in advance and their appearance depends on allocation of other activities, take advantage of such mixed approach. We give a list of typical features of complex process environments in Section 2 of the paper where the problem area is specified. In Section 3, we compare planning and scheduling tasks in deep and we propose how to extend the traditional scheduler by planning capabilities. Section 4 is dedicated to description and comparison of several modelling approaches to scheduling problems. We study the capabilities of the models to capture problems in complex process environment and we give an overview of conditions when the particular model can be applied successfully. The paper is concluded with some final remarks describing our first experience with implementation of the proposed methods in the VisOpt project of generic scheduling engine for complex process environments (Bartak, 1999a).

2 Problem Area

The problem area that we deal with can be characterised as a complex process environment where a lot of complicated real-life constraints bind the problem variables. Typical examples of such environments can be found in plastic, petrochemical, chemical, pharmaceutical and food industries. One of the goals of the VisOpt planning and scheduling project (Barták, 1999a) is to develop a generic scheduling engine that can be customised easily for particular environment via the description of resources, initial situation and required future situations.

The problem domain can be described as a heterogeneous environment with *several resources* interfering with each other. Currently we are working with producers, movers and stores, later other resources like workers and tools will be added. The task is to generate (to plan) activities necessary to satisfy custom orders (and other marketing requirements) and to allocate (to schedule) the activities to resources over time.

There exist alternative resources for processing the activity and some resources can handle several activities at a time (this is called batch processing). In case of batch processing, compatibility and capacity constraints restricting which products and in what quantities can be processed, i.e., produced, moved or stored together, must be considered. Also the order of activities processed by the resource is not arbitrary but the currently processed activity influences what activities can follow. Consequently, we must follow the *transition patterns* and assume the *set-up times* between the activities as well. The processing time is usually variable and there is defined a *working time* when the activities can be processed in resources.

Alternative processing routes, alternative production formulas and *alternative raw materials* are other typical features of above mentioned industry areas. In addition to the core products it is possible to produce the *by-products*, typically during set-ups. The by-products should be used as a raw material in further production and there is a push to use them this way because they will fill-up the available storing capacity otherwise. Consequently we must schedule processing of by-products. During production of the core product some *co-products* may appear. The co-products can be used to satisfy other orders, they can be sold as an alternative to the ordered item or they can be processed further as a raw material. Again, processing of co-products must be scheduled as well because of limited capacity of warehouses where all the products are stored. Last but not least there is a possibility of *cycling*, i.e., processing the item for several times for example to change features of the item or just to clean up the store, and *re-cycling*, i.e., using of by-products and co-products as a raw material. See Figure 1 for example of complex process environment with cycling/re-cycling.

Typically, the production in complex process environments is not driven by the custom orders only but it is necessary to schedule the production for store according to the factory patterns and the forecast. It means that the scheduler should handle some planning tasks too.

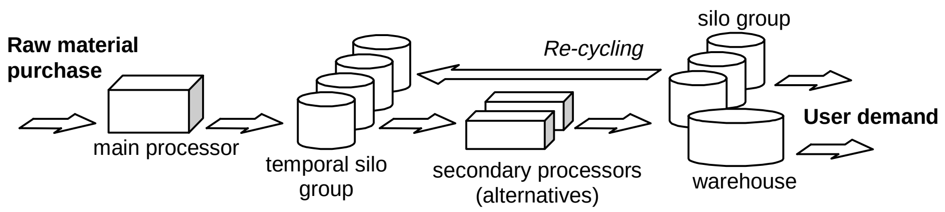

Figure 1 (example of complex process environment)

Typically, in a scheduling project the users deal with finding no arbitrary schedule but an optimal schedule. Usually a makespan is used as the objective function (Caseau, Laburthe, 1994, 1996, 1997). The idea of minimising the makespan, i.e., the maximum completion time of the activities, follows the assumption that shorter production time implies lower cost and lower cost implies higher profit. However, this is not necessarily true in many complex process environments where expensive set-ups must be considered. Also, makespan may be used if all the activities are known in advance, but, again, this is not a case in many complex process environments due to set-ups and production for store. Therefore in the VisOpt project the task is to schedule the most profitable production for fixed period of time (more precisely, we are looking for a schedule with good profit). The profit is measured here by the overall production cost and by the price for selling the products delivered according to the custom orders. We do not describe the cost handling in detail in the paper; it will be a subject of next paper.

The solved problem hardly fits into any category of typical scheduling problems as used by OR because of many complex constraints that make it hard to be tackled by pure OR methods. It also does not fit into any basic category of CP scheduling problems due to dynamic character when new activities appear during scheduling. Perhaps, it is closest to the group of resource constrained project scheduling problems (RCPSP). RCPSP (Crawford, 1996) is a generalisation of job shop scheduling (Baptiste et all, 1995) in which activities can use multiple resources, and resources can have capacity greater than one (more activities can be processed together). The definition of RCPSP as well as of all other scheduling problems expects the set of activities to be known before the scheduling starts. Unfortunately, this is not necessarily true in the complex process environments where scheduling the activity to a particular resource or time may introduce new activities to the system. Typically, using alternative processing routes, by-products, co-products and production for store cause such behaviour. Using the foregoing planning phase provides a little help in such cases as we argue in the next chapter.

3 On the Boundary of Planning and Scheduling

Opposite to (Srivastava & Kambhampati, 1999) and according to our experience with APS (Advanced Planning and Scheduling) systems we argue that in most current APS (Advanced Planning and Scheduling) systems the planning and scheduling tasks are handled separately in different modules and the communication between the modules is limited. Such decomposition seems natural because the traditional planning and scheduling deal with a bit different tasks and different methods are used to solve the tasks. On the other side, in industry the notions of planning and scheduling are merged and the difference between them is fuzzier.

3.1 Traditional (separate) Planning and Scheduling

The traditional definition of planning says that *planning* tackles the problem of finding plans to achieve some goal, i.e., finding a sequence of activities that will transfer the initial world into one in which the goal description is true (Pool et all,1998). It means that a description of the initial world, the (partial) specification of the desired world and the list of available activities make the input of the planner. A solution is a sequence of activities that leads from the initial world description to the goal world description and it is called a plan. A typical planning task in the industry consists of finding the production sequences to satisfy the custom orders.

The traditional *scheduling task* deals with the exact allocation of activities to resources (or resources to activities) over time respecting precedence, duration, capacity, and incompatibility constraints (Brusoni et all, 1996). The set of activities, the list of resources and the specification of the constraints make the input to the scheduler. The output of the scheduler consists of the exact allocation of the activities to the resources over time.

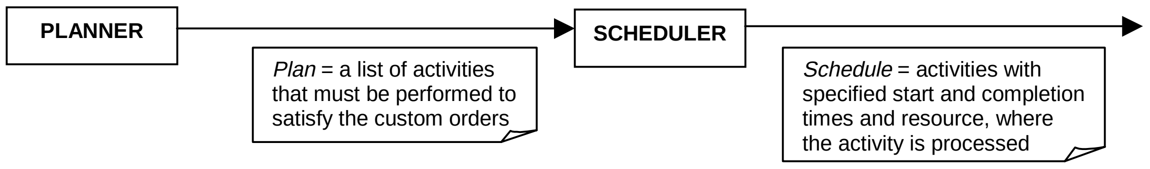
PLANNER
SCHEDULER
Plan = a list of activities that must be performed to satisfy the custom orders
Schedule = activities with specified start and completion times and resource, where the activity is processed

Figure 2 (separate planning and scheduling)

As Figure 2 shows the communication between separate planning and scheduling modules is simple: the planner prepares a list of activities as well as some constraints, namely the precedence and duration constraints, for the scheduler. The remaining constraints for the scheduling, like the capacity and compatibility constraints, and the list of resources are derived from the factory specification. This simple decomposition is the nicer side of the thing.

On the other side both planner and scheduler should consider the same environment limits and constraints even if the planner uses a more general view while the scheduler deals with the details. In reality the planner does not assume all the production details so it is possible that it generates too tighten plans that are impossible to schedule or it generates plans that are too released, i.e., the plan does not utilise the factory capacity fully.

Visibly, to get a good final schedule we must start with a good plan. Either a more informed planner, that assumes the same constraints as the scheduler does, is used or there must be backtrack from the scheduler to the planner to find better plan. This backtrack occurs when the scheduler finds a clash in the plan or it finds that the resources' capacities are not utilised fully. Naturally, backtrack from the scheduling stage to the planning stage complicates the communication between the planning and scheduling modules because the scheduler should inform the planner about the cause of backtrack via identification of the conflict or the not fully utilised resource. Also, the planner must be capable to exploit this additional information so it should handle some scheduling tasks as well. Briefly speaking, the planner should care not only about "what should be processed" but also about "how it should be processed".

Another problem in the traditional definition of the scheduler is that it expects all the activities to be known before the scheduling starts. As we sketched in Section 2 such requirement is sometimes inappropriate in complex process environment because the existence of some activities depends on allocation of other activities to resources. The typical example of such behaviour is using alternative processing formulas, alternative raw material, set-ups or producing by-products. Again, either the planner must consider some scheduling constraints and it introduces all necessary activities in advance or the scheduler must be able to generate activities during the scheduling that is a typical planning task.

3.2 Planning and Scheduling in Industry

In the real life, the notions of planning and scheduling are not strictly distinguished and sometimes there is confusion between them.

The notion of planning means preparing a plan but what is it a plan? We may have a *marketing plan* that describes the quantities and approximate release times of products using market forecast and current custom orders. This plan is usually for a longer time period and it is more accurate in earlier times than in later times. Notice that the marketing planning has almost nothing in common with the traditional AI planning described in the previous section. The result of marketing planning consists of the list of demands to the production so there are no sequences of actions that "change the world".

The marketing plan makes the input to *production planning* whose task is to generate a production plan, i.e. a sequence of activities necessary to satisfy the orders (demands) from the marketing plan. The definition of production planning is very close to the traditional planning but the production planning usually covers the allocation of the activities to factory departments as well, that is a typical scheduling task. Production planning uses information like BOM (bill of materials) to generate processing routes and to find what raw material should be ordered and when. Again, the production plan is prepared for a longer period of time.

Finally, there is a *production scheduling* which allocates the activities from the production plan to particular resources over time. The scheduler works with the detail information about the resources, like capacity and compatibility constraints, and the resulting schedule is prepared for a shorter period of time than the production plan (because of efficiency issues and unexpected changes in the environment that lead to modifications of the schedule). The definition of production scheduling is very close to the traditional scheduling, but sometimes during scheduling we need to introduce new activities to process by-products etc.

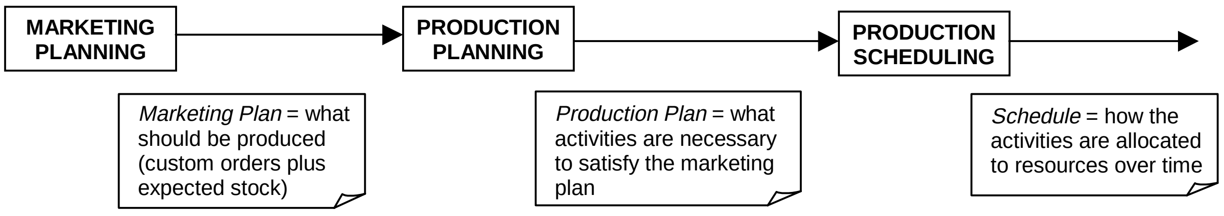
MARKETING PLANNING
PRODUCTION PLANNING
PRODUCTION SCHEDULING
Marketing Plan = what should be produced (custom orders plus expected stock)
Production Plan = what activities are necessary to satisfy the marketing plan
Schedule = how the activities are allocated to resources over time

Figure 3 (planning and scheduling in industry)

As you see, there is no conceptual difference between the production planning and scheduling. Both tasks cover the generating of activities as well as assigning the activities to the resources. The resolution of the result is the main difference between the production planning and scheduling. While the production planning works with departments and a longer period of time, the production scheduling handles individual machines in shorter period of time.

The similarity between the production planning and the production scheduling brings us to the idea of handling both tasks together within single mixed framework.

3.3 Mixing Planning and Scheduling

Let us now return to the specific features of the problem area that are described in Section 2 and analyse them from the planning and scheduling point of view.

First, there are alternative processing routes, alternative production formulas and alternative raw materials. The choice of the alternative is part of the planning task but the information necessary for good decision is available at the scheduler level because the decision may depend on particular allocation of activities to resources. For example, the resource that is chosen to process the activity has no access to some raw material (to the store with the material) so not all production formulas are available for the resource and, consequently, different supplying activities are used for the resource.

Second, there is a production of the by-products and the low-quality products that are produced as “waste” or during the transition between activities. Again, the planner is responsible for generating the activities to process these products but it is the scheduler that decides what and if any by-product appears by assigning the activities to a particular resource. Remind that processing of the by-products should be scheduled as well because they may fill the stores otherwise.

Third, there are transition patterns and set-up times that are usually modelled using special transition and set-up activities (Pegman, 1998). Again, the generation of these activities is part of the planning task but the existence of the activities depends on the allocation of other activities to resources that is a scheduling task.

Finally, there is a production for store. Normally, the marketing plan should specify the production of items that are not ordered by real customers. Nevertheless, sometimes it is more appropriate to delegate this decision to the scheduler. For example, it could be cheaper to schedule continuous production, i.e., to add new production activities, than stopping the machine. In such case, it is the decision of scheduler how the gap between activities is filled and what should be produced.

The discussions in the above paragraphs and sections justify our proposal of mixing the traditional planning and scheduling tasks into single framework. Briefly speaking, we suggest enhancing the traditional scheduler with some planning capabilities; in particular, we allow generating of activities by the scheduler. We call this enhanced scheduler simple a *production scheduler*. We expect to preserve the separate marketing planner that generates the demands for the production but the production scheduler can schedule non-ordered production too[1].

The production scheduler consists of the *activity generator* (former planner) that generates the activities and the *activity allocator* (former scheduler) that allocates the activities to the resources over time (almost) immediately. By attempting to allocate the activity to the resource after its introduction we can detect the clashes sooner as well as we can remove some alternatives via constraint propagation that restricts the domains of activity parameters. See Figure 4 for proposed structure of the mixed planning and scheduling system.

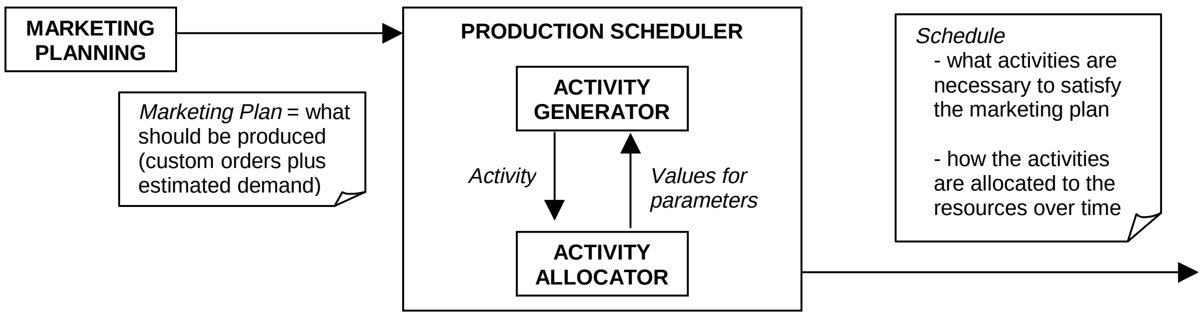
MARKETING PLANNING
Marketing Plan = what should be produced (custom orders plus estimated demand)
PRODUCTION SCHEDULER
ACTIVITY GENERATOR
Activity
Values for parameters
ACTIVITY ALLOCATOR
Schedule
- what activities are necessary to satisfy the marketing plan
- how the activities are allocated to the resources over time

Figure 4 (mixed planning and scheduling)

The communication between the generator and the allocator is simple via single activities. The generator introduces activity to the system and asks the allocator to schedule it. The allocator influences the generation of further activities by restricting the domains of parameters for already introduced activities. It can also ask the generator to introduce new activities explicitly, e.g., to generate set-ups, transitions, supplying or consuming activities, if the required activity is not present in the system. The generator is driven by the set of initial activities that can describe the initial situation as well as the future demands generated by the marketing planner. Also notice that depending on the resolution of the scheduling we can use the production scheduler both for the production planning and for the production scheduling described in the previous chapter. Naturally we use different activities and different resources for production planning and production scheduling but the overall scheduling mechanism is the same.

4 Modelling Complex Process Environments

In general, it is a good design principle to create a declarative and thus transparent model of the problem. All entities are described initially with constraints that define their nature and the relationships between them and the search code is kept separate from this declarative model. Constraint Programming supports exactly this form of modelling: the user states the problem, the computer solves it. However, relying on this statement is also the biggest danger of real-life projects based on constraints because the generic constraint satisfaction and optimisation

[1] The quantity of non-ordered production is restricted by the cost factor, e.g., assuming the cost of storing.

algorithms are still not capable to tackle efficiently large-scale industrial problems brought by real-life. At least without additional help. Thus, the constraint modelling, i.e., a description of the problem by means of constraints, is very important part of all projects. It is also a known true that a small change in the model or in the data can influence dramatically the performance of the system.

In the following sections we survey several constraint models for scheduling problems and we give general rules when the models are applicable. Nevertheless, we are aware of necessity to tune each model for particular application.

4.1 Constraints

Look first at the type of constraints that appear in a typical scheduling application. We will concentrate on scheduling in complex process environments but the constraint classification that we propose can be identified in other scheduling problems as well.

For a scheduling problem it is typical to have several resources that provide some services over time. Different resources have different capabilities that must be taken in account during scheduling. We distinguish between two groups of constraints concerning single resource.

There are constraints describing the limits of the resource at given time point, we call them *resource constraints*. Typical example of resource constraint is the capacity constraint stating how many activities can be processed in parallel (or how many items can be stored together etc.). Compatibility (incompatibility) constraint is another example of the resource constraint. The compatibility constraint states what activities can be processed together, i.e., what activities are compatible. While the capacity constraint is a “quantity type” constraint (how many?), the compatibility constraint can be seen as a “quality type” constraint (what?). See Figure 5 for example of resource constraints

***The capacity constraint*:**

$\forall T \quad \sum_{items} Quantity(R,T,Item) \le Capacity(R)$, where T is a time point and R is a resource.

The sum of quantities of all processed items does not exceed the capacity of the resource.

***The compatibility constraint*:**

$$\forall T \quad (Quantity(R,T,Item1) = 0 \vee Quantity(R,T,Item2) = 0)$$

If Item1 is incompatible with Item2 then they cannot be stored or processed together.

Figure 5 (the resource constraints)

The second group of constraints concerning single resource describes the behaviour of the resource in time. In particular these constraints specify what future situations may follow the current situation, for example we can specify that the same type activity may follow the given activity or a set-up activity must follow it. Thus, we define transitions between activities and therefore this group of constraints is called *transition constraints*.

***The transition constraint*:**

$\forall T \quad \mathrm{activity(R,T) = A \Rightarrow activity(R,T+1) \in \{A, setup\text{-}A\text{-}B, setup\text{-}A\text{-}C\}}$,

where T is a time point and R is a resource.

After the activity A only the same activity A or a set-up activity may follow.

Figure 6 (the transition constraint)

Resource and transition constraints specify fully the features and capabilities of single resource. If we describe the situation of the resource at given time by the set of variables, e.g. what activities are processed, then the resource constraints bind the variables of single time point while the transition constraints bind the variables of different, usually consecutive time points. See Figure 7 showing the constraint classification in the Gantt chart.

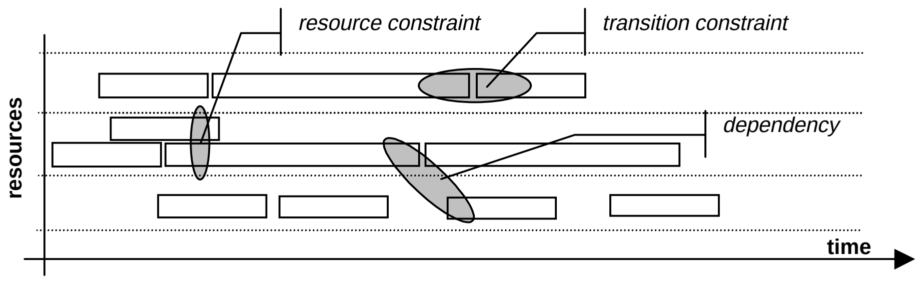

Figure 7 (constraint classification)

The resources in a typical scheduling problem are hardly independent so in the model we must also describe the relations between resources. Naturally, the constraints between different resources are called *dependencies* and they bind variables describing different resources at perhaps different time points. Typical example of a dependency in production scheduling is the supplier-consumer dependency that specifies the relation between supplying and consuming activities. In other problems we may require two activities to be processed in parallel by two resources, for example two lecturers teach two equivalent courses in parallel.

The supplier/consumer constraint:

$$Quantity(R,T,Item) = Quantity(R,T-1,Item) + \sum_S Quantity(S,T-X_{S,R},Item) - \sum_C Quantity(C,T+X_{R,C},Item)$$

The quantity of Item in the resource R in the time T is computed using the quantity in time T-1 plus the sum of supplied quantities minus the sum of consumed quantities. $X_{A,B}$ is a transport time between the resources A and B expressed in multiples of slice duration.

Figure 8 (the dependency)

The classification of constraints into one of above groups is not fixed and it depends on what objects in the problem we choose as resources.

4.2 Time-line vs. Activity-based Models

Both production planning and production scheduling are tasks that deal with time and therefore modelling of time is necessary there. Generally, two different views of time are used: discrete time and event-based time.

We call the model based on discrete time a time-line model. The *time-line model* (also called a timetable approach) is a general method of describing dynamic processes using discrete time intervals. First, we divide the time line into sequence of time slices with identical duration and at each time point (the point between two slices) we describe the situation of each resource using several variables. We may also describe the situation of the resource in the time-slice that is more often in timetabling applications.

It is assumed that the behaviour of resource is homogeneous between two consecutive time points, i.e., the key events like changing activity occur only at the edge of two consecutive time slices. Thus, the duration of time slices must be defined according to the duration of activities that can be processed by the resources so it should be a common divisor of activities' duration. Naturally we prefer longer duration of the slice because it means smaller number of variables and consequently less work to do when the variables are labelled. Together, the *duration of slice* is computed as a greatest common divisor of duration of all activities in all resources.

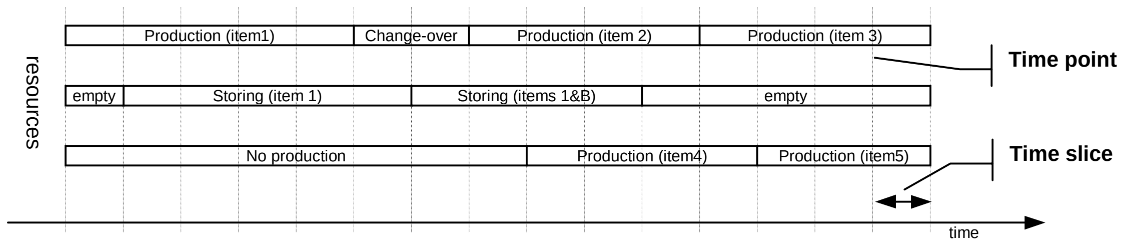

Figure 9 (the time-line model)

Now we can describe the situation at each time point using a set of variables. For example in case of store there is a variable for each item that can be stored and this variable specifies the stored quantity. Other variables can specify the state of the resource etc. The resource constraints bind variables in the time point and they can express compatibility between stored and processed items or capacity limits of the resource. The transition constraints bind variables from consecutive time points but still within single resource. Finally, there are dependencies binding variables from different resources and from (usually) different time points. We give examples of constraints for time-line model in the previous section and in (Barták, 1999b) & (Barták, 1999c).

Note that if we can define the discrete intervals, i.e., if we know the scheduled period and the duration of all the activities, then we know the number of time points as well. Because we know the set of variables describing the situation for each time point in each resource we have all the variables at the beginning of scheduling and there is no problem to post all the constraints before the scheduling starts. Consequently, we can use arbitrary constraint satisfaction method including local search.

If we study the time-line model from the mixed planing and scheduling point of view we see that there is no necessity to use separate activity generator. This is because the activity in given time point is specified by a special variable so generating/choosing the activity means just assigning the value to this variable. Consequently, the activity generator is integral part of the labelling procedure and we can use AI planning techniques, which propose the activity to be chosen, there.

Transparency and generality are two main advantages of the time-line model. It can capture various planning and scheduling tasks naturally and it is also easy to express heuristics. Last but not least, the time-line model can incorporate existing solving techniques.

Unfortunately the time-line model has the disadvantage of using huge number of variables when applied to a large-scale problem. This drawback is hidden in the definition of the time slice duration. In most cases we can expect that this duration will be very short, even if the activities take long time. For example, if there are activities with the duration 25 and 26 seconds respectively, we still have to define the time slice to have the duration 1 second because we don't know the order of activities beforehand and we need a time point for each possible activity change. As a consequence, there are a large number of time points that implies huge number of variables to label. Therefore, we can expect not very good efficiency from the model if applied to the large-scale scheduling problems. However, we believe that the model can be applied successfully in cases when:

- the description of resources is not very complicated, i.e., we use a small number of variables to describe the situation,
- the ratio between the time slice duration and the scheduled duration is higher, i.e., either the scheduled duration is short or the duration of the time slice is long.

Time-line model can also be used in combination with activity-based models that are described further. In this case we can exploit the time line model to reduce the number of combinations of activities in time points. Because we need not to describe the schedule fully by the time-line model there, we may use longer duration of time slices and thus the model becomes tractable. Another possibility to make the model tractable is to use progressive duration of time slice, i.e., a longer duration of slice in later times. This is justified by the assumption that the schedule may be less precise in later times.

Because of efficiency issues of pure time-line model, we turned our attention to a more common representation of the scheduling problems using activities. In *activity-based models*, as we call this approach, we use event-based time where the event corresponds to activity change in the resource or, more precisely, to start and completion of the activity[2]. Activity describes some homogenous duration of processing, for example a production of given item in production scheduling or a lecture in school timetable. See Figure 10 for scheduled activities in the form of Gantt chart.

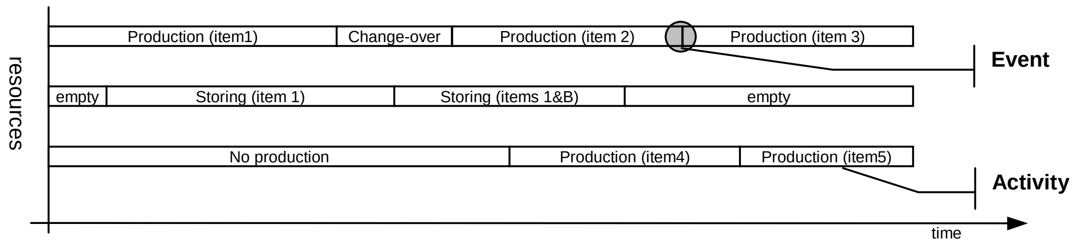

Figure 10 (the activity-based model)

We describe the behaviour of each activity by a set of variables. Typically, these variables include start and completion time of the activity and depending on the organisation of activities (see next section) we need a variable indicating the resource where the activity is processed or variables identifying the dependent activities. Notice that the number of variables corresponds to the number of activities to be scheduled.

In the activity-based models, the resource constraints bind the variables from several activities that are processed by single resource at given time. Notice that until we know the allocation of the activities to the resource over time we cannot decide which activities are bound by the resource constraint. This feature introduces dynamic behaviour to some models where constraints are posted during scheduling. In some cases it is possible to include the "trigger" into the resource constraint itself so the constraint can be posted at the beginning of the scheduling but it is fired only in particular situations. In many problem areas this trigger is very simple, we just check if the activities are scheduled to the same resource and if there is an overlap between the activities. However, in other problem areas like complex process environments, the trigger may complicate the constraint significantly.

Transition constraints bind the variables from (usually) consecutive activities scheduled for single resource. Similarly to resource constraints, we cannot decide which activities are

[2] It is possible to have gaps between activities when no activity is processed by the resource.

bound by the transition constraints until we know the allocation of activities. Now the situation is even more complicated because we cannot decide about posting the transition constraint using only the description of the connected activities. For example, we need to know that there is no other activity between these two connected activities. Again, in many problem areas we do not use the transition constraints at all or the transitions are simple. But in complex process environments the transitions and set-ups play important role and we must be able to express them in form of transition constraints.

Finally, there are dependencies describing relations between activities processed by different resources. If we know all the activities in advance, we also know which activities are dependent. For example, in production scheduling the dependencies express the supplier/consumer relation saying that one activity must complete before another activity etc. However, if we introduce new activities during scheduling then we must be able to identify the dependent activities to post the dependency constraint.

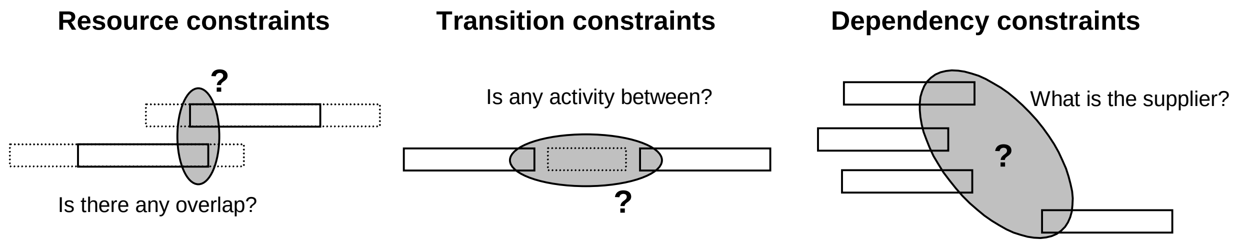

Figure 11 (dynamics of constraints in the activity-based models)

As we indicate in above paragraphs the idea of mixed planning and scheduling becomes more complicated in activity-based models than in the time-line model. In activity-based models we need the activity generator that is responsible for generating new activities as well as for posting corresponding constraints. Now the structure of the scheduler corresponds more to the structure proposed in section 3.3 (see Figure 4) with separate activity generator and activity allocator.

Activity-based models require the user to be able to identify the activities. In some environments the activities appear naturally but for example, it could be more complicated to define activities in continuous production. Nevertheless, it is still possible to divide such production into sequence of batches that form the activities. In such case the duration of activity determines the resolution of the schedule.

Let us identify the typical problem areas now where the time-line model and activity-based models are used. We may notice that the time-line model prevails in the areas with fixed resolution for all resources where the duration of activities is almost the same and the activity duration is a small multiple of the time slice. In such areas, like school timetabling, the number of time-slices is not large so the main disadvantage of the time-line model, i.e., the huge number of variables, is not relevant here. In fact, the number of variables is comparable to the activity-based models but the constraints are usually easier in the time-line model.

Activity-based models prevail in the areas like production scheduling where it is natural to identify the activities, the duration of activities is variable and various resources are used. Large-scale problems in complex process environments are typical example that requires usage of activity-based models because the number of variables in the time-line model is not tractable here.

4.3 Task-centric vs. Resource-centric Models

In the previous section we described the activity-based models in general but we may further distinguish between these models using the form of activity organisation.

In many scheduling problems the activities are organised into tasks where the task is defined by a sequence of activities that must be processed to solve the task. In production scheduling this sequence determines the production chain, i.e., it defines the track of an item through the factory starting from the raw material and finishing with the final product. Typically, we assign the production chains to the custom orders so in (Barták, 1999b-d) we called such models order-centric models. Because satisfaction of a custom order is just a special form of task, we will use the more frequent term *task-centric model* here. Note that by the task we mean not only a linear sequence of activities but also, for example, a tree of activities as Figure 12 shows.

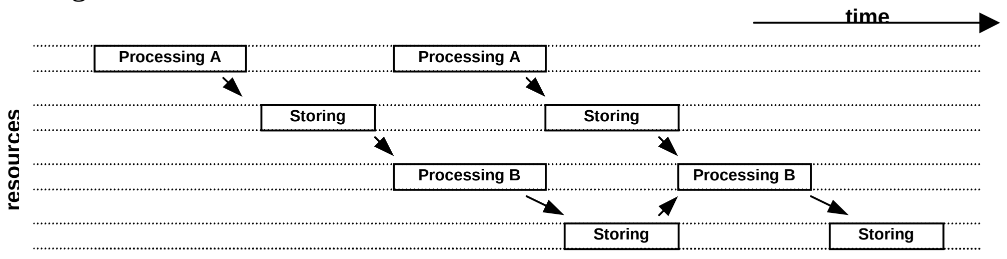

Figure 12 (a task-centric model)

By organising the sequences of the activities into tasks, we naturally describe the (supplier-consumer) dependencies between the activities. Consequently, the dependency constraints can be posted before the scheduling start. The resource and transition constraints bound activities from different tasks typically, thus we either need to equip these constraints by the triggers or we should post the constraints dynamically. The choice of the method depends on the complexity of the constraints. If the resource and transition constraints are easy enough then the static constraints with simple triggers are the preferable solution.

The task-centric model is appropriate for problem areas where we know all the tasks in advance and we know how to decompose the tasks into activities. However, in complex process environments, there exist typical problems that make the task-centric model less applicable.

The first problem is using the *alternative processing routes* that are typical for chemical industry. This means existence of several ways how to produce the item (solve the task), i.e., instead of single production chain we have a set of more or less different production chains per task. The typical example is inserting re-heat activity if the whole chain of activities takes too long (Pegman, 1998).

The second problem is modelling the *set-up times* (Pegman, 1998) and other more complicated resource constraints. The set-up time specifies the necessary duration/gap between consecutive activities processed in single resource and, consequently, it depends on the order of activities chosen during the scheduling. The set-up time can be modelled by introducing special set-up activities with specified duration dependent on the previous activity in the schedule.

The next problem is the processing of *by-products* and *co-products*, i.e., the production of non-ordered items that can be re-used in the future production. Now, the situation is more complicated because it requires two or more production chains to interact in a more advanced way, i.e., if a by-product is produced in one production chain then another production chain may use this by-product as a raw material. This problem has a close connection to the previous problem because by-products are produced typically during the set-ups.

Finally, there is a problem with the *production for store*, i.e., non-ordered production. Unfortunately, this problem cannot be solved in pure task-centric model fully because there are no tasks defined for non-ordered items.

The proposed mixed planning and scheduling approach helps us to solve all above problems using the same way. If we allow generating the activities during scheduling then there is no problem to start the scheduling with several "core" activities and introduce new activities if necessary. We show how to represent the task-based model allowing such dynamic behaviour in the next section.

In addition to tasks there exists orthogonal method of organising activities by resources. In the model, that we call *resource-centric model*, we describe all the activities that may be processed by given resource. Then, the activities used to satisfy the task are grouped via dependencies implicitly during the scheduling. The resource-centric model is more appropriate for the description of a factory than the task-centric model. This is because we concentrate on the specification of resources and this specification is independent of the actual set of tasks. This is similar to the time-line model, where we also describe the resources primarily.

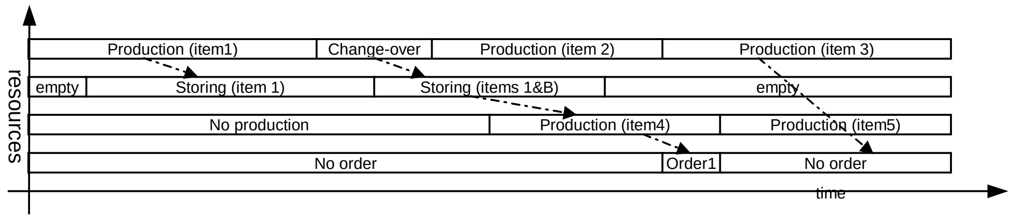

Figure 13 (a resource-centric model)

In the resource-centric model we do not assign activities to resources but we allocate resource activities over time in such a way that all the dependencies between activities hold (see Figure 13). Note that expressing the resource and transition constraints is now easier because we know the activities belonging to the resource. However, the dependencies are more complicated in the resource-centric model and they must be introduced dynamically. Also notice that the problems mentioned above for the task-centric model are not relevant here, because the resource-centric model uses explicitly the mixed planning and scheduling approach; the activities are introduced during scheduling.

We may see the resource-centric model as a complement to the task-centric model. Currently, probably the task-centric model is more widely used because an order-driven production is scheduled typically, there are no transition constraints and the resource constraints are rather simple. However as soon as more complicated resource constraints appear and the transition constraints (set-ups with by-products) become crucial then we believe the resource-centric model be more appropriate for such problems. This is because the complexity of dependencies is similar in both models but the resource and transition constraints are easier to express and maintain in the resource-centric model.

4.4 Dynamic vs. Static Representation

A typical method of solving problems using constraint programming includes three steps:

1. introduce variables,
2. post constraints among the variables, and
3. label the variables using propagation techniques and/or (local) search.

This traditional methodology expects the static representation of the problem where all the variables and the constraints are known in advance. The static representation has the advantage that all constraint solving methods can be applied. Also, if we know all the constraints we may expect better propagation to reduce the domains of variables.

As we argued in previous sections, mixed planning and scheduling can help us to solve some complicated problems and because this approach prefers generating new activities during scheduling we need some form of dynamic introduction of variables and constraints as well. Nevertheless, the "dynamic character" of mixed planning and scheduling does not exclude the static representation of the models as we show in the next paragraphs.

The variables in the time-line model are naturally represented statically because we know the number of time points (time slices) and we know what variables are necessary to describe the resources. Because we have all the variables, there is no problem to post the constraints among them before the scheduling starts. This is perfect for the resource constraints binding variables in single time point as well as for the transition constraints binding variables in consecutive time points. However, the static representation is questionable in case of dependencies because, for example, we do not know what suppliers and consumers are used in given time point till the activity is chosen here. We can use the static representation of the constraints using triggers that fire the constraint. The problem here is that many dependency constraints are posted even if most of them will not be actually used. Also, the propagation through the constraints with triggers is restricted till the constraint is fired. Therefore we propose to postpone the introduction of dependency constraints till enough information is available to create the constraint between right variables. This is a dynamic representation that allows us to save memory and time necessary to check triggers (see Figure 14 for comparison of static and dynamic dependencies).

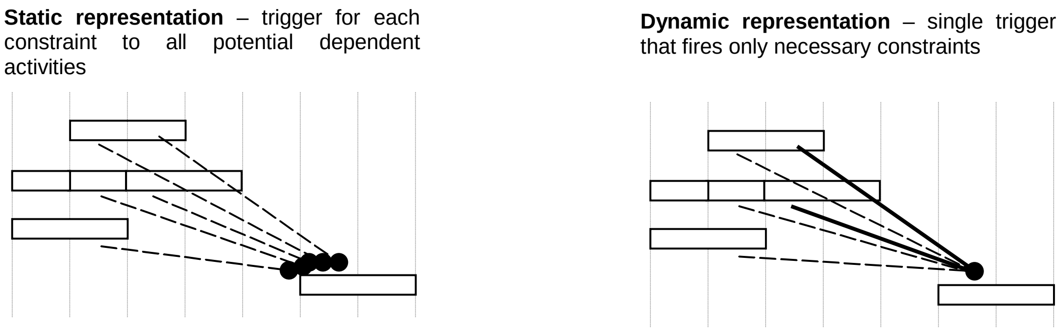
Static representation – trigger for each constraint to all potential dependent activities
Dynamic representation – single trigger that fires only necessary constraints

Figure 14 (static vs. dynamic dependencies in the time-line model)

In activity-based models the situation is more complicated because if we use mixed planning and scheduling approach with introduction of new activities then we do not know all the activities (variables, constraints) in advance. Typically, the task-centric models use static representation that makes them hardly applicable to complex process environments as we sketched in the previous section. The static representation for the resource-centric model restricts the scheduling to ordering of activities in the resource so this representation is restricted to few problems only.

To overcome the limits of static representation we propose to use a dynamic representation where the activities and the constraints between activity variables are generated dynamically. The structure of such system corresponds fully to the production scheduler that was proposed in section 3.3. The production scheduler starts with several activities, in case of task-based model these activities are the roots of the production chains, in case of resource-based model the initial activities may describe the initial situation of each resource and the required future situations. New activities are generated according to the allocation of already present activities. The advantage of dynamic approach is that we do not need the constraints with complicated triggers. The drawback of full dynamic representation is loss of the propagation; if there are no constraints (until the constraints are introduced) then we cannot propagate the values between variables.

To suppress the handicap of weak propagation, we propose a semi-dynamic representation that shares the advantages of static representation (good propagation) with advantages of dynamic representation (simple constraints – without complicated triggers). The basic idea of the semi-dynamic representation is to express statically everything that is known in advance. The objects and constraints that are very complicated to express statically (like different data structures in different activities and constraints with complicated triggers) are expressed dynamically which means that they are generated during scheduling.

In activity-based models we may estimate the number of activities to be scheduled and instead of introduction of real activities we generate only empty shells for the activities (we call the shells virtual activities) that will be filled by a real activity during scheduling. In the activity shell we may use the variables that are common for all activities so we can post constraints among these variables immediately. If the activity in the shell is known then other variables describing the specific activity are added into the shell and remaining constraints can be posted. See Figure 15 showing the differences between static, dynamic and semi-dynamic representations for activity-based models.

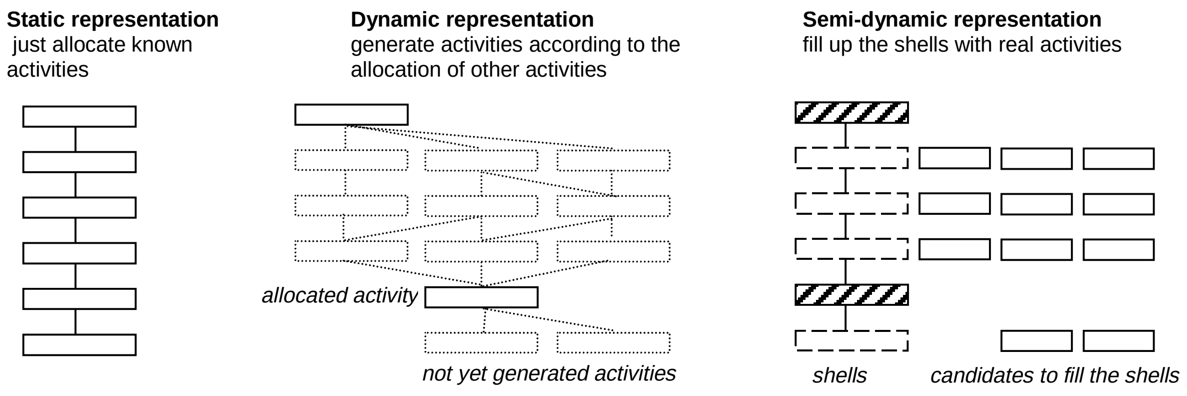
Static representation
just allocate known activities
Dynamic representation
generate activities according to the allocation of other activities
Semi-dynamic representation
fill up the shells with real activities
allocated activity
not yet generated activities
shells
candidates to fill the shells

Figure 15 (comparison of representations of activity-based models)

In most current scheduling applications the static representation is used because of efficiency issues. Also, when all the constraints are posted then the scheduler may concentrate on labelling only. Nevertheless as noted in (Nareyk, 2000), the planning is more variable and it is impossible to predict which actions will be used in which combination. Therefore, the conventional (static) formulation of constraint satisfaction problem (CSP) is too restrictive to solve planning problems and the dynamic representation is more suitable for them. Naturally, the same statement holds for the scheduler enhanced with some planning capabilities like the schedulers for complex process environments. For representation of constraint models in mixed planning and scheduling environment we argue for using the semi-dynamic representation that overcomes the limits of conventional CSP but still preserves CSP advantages like constraint propagation.

5 Concluding Remarks

In the paper we analyse the models for solving scheduling problems in complex process environments using the constraint programming technology. We study both planning and scheduling tasks there and we propose an innovative approach for solving such problems using the scheduler enhanced by some planning capabilities. The advantages of this method are presented on examples of solving typical problems in complex process environments.

We compared three constraint models used to solve scheduling problems and we identified their advantages and drawbacks. We also give some guidelines when the particular model can be applied and when it goes better than the other models. The results are summarised in Table 1 and Table 2.

		model		
		time-line	**task-centric**	**resource-centric**
constraint	**resource**	*easy*	*complicated*	*easy*
	transition	*easy*	*complicated*	*easy*
	dependency	*little complicated*	*easy*	*complicated*

Table 1 (expressing constraints in models for scheduling problems)

The time-line model is a very good model to capture dynamic processes, as it is easy to express all the constraint groups there. However because of its huge size the time-line is applicable only if the number of time-slices is not very large, e.g., in timetabling applications. The other two models are based on the notion of activity and we can see them as complementary models. While the task-centric model is more suitable for scheduling of order-driven with simple resource and transition constraints, the resource-centric model is better for complex process environments where the resource and transition constraints are rather complicated.

		model		
		time-line	**task-centric**	**resource-centric**
problem	**non-ordered production**	*implicit*	*no (limited)*	*implicit*
	cycling	*implicit*	*limited*	*implicit*
	alternatives	*implicit*	*limited*	*implicit*

Table 2 (how the constraint models capture typical scheduling sub-problems?)

If we compare the constraint models from the point of view of expressiveness, then both the time-line and the resource-centric models are capable to solve the typical problems in complex process environments mentioned in Section 2. The order-centric model is less appropriate to capture these problems especially if the static representation is used.

In the paper we also present three different representations of the constraint models, the static, dynamic and semi-dynamic representation. We argue here for using the semi-dynamic representation that preserves the advantage of constraint propagation and provides the flexibility of dynamic constraints. Table 3 summarises some of the features of the studied representations.

	representation		
	static	**dynamic**	**semi-dynamic**
constraint posting	*before*	*during*	*before/during*
local search	*yes*	*no*	*no*
propagation	*good (???)*	*bad*	*good*
constraint expressiveness	*complicated*	*easy*	*easy*

Table 3 (some features of representations)

The paper summarises the results from (Barták, 1999b-d); we give a global view of the problem of planning and scheduling in complex process environments. We concentrate on the expressive power of the models and representations here but we also mention some efficiency issues. The theoretical results presented in the paper are currently verified in the implementation of the generic scheduling engine for complex process environments. This engine is being developed within the VisOpt project for InSol Ltd (Barták , 1999) & (VisOpt, 1999).

6 Acknowledgements

Author's work is supported by the Grant Agency of the Czech Republic under the contract number 201/99/D057 and by InSol Ltd. I would like to thank Yossi Rissin and the team of InSol for introducing me to the problem and for interesting and encouraging discussions concerning real-life problems of industrial planning and scheduling. I am also grateful to Helmut Simonis from Cosytec for drawing my attention to some existing applications in the area.

7 References

Baptiste, P., Le Pape, C. (1995a): A Theoretical and Experimental Comparison of Constraint Propagation Techniques for Disjunctive Scheduling, in: Proceedings of the Fourteenth International Joint Conference on Artificial Intelligence, Montreal, Canada

Baptiste, P., Le Pape, C. (1995b): Disjunctive Constraints for Manufacturing Scheduling: Principles and Extensions, in: Proceedings of the Third International Conference on Computer Integrated Manufacturing, Singapore

Baptiste, P., Le Pape, C., Nuijten, W. (1995): Constraint Based Optimisation and Approximation for Job-Shop Scheduling, in Proceedings of the AAAI-SIGMAN Workshop on Intelligent Manufacturing Systems, IJCAI-95, Montreal, Canada

Barták, R. (1997): On-line Guide to Constraint Programming, http://kti.mff.cuni.cz/~bartak/constraints/

Barták, R. (1999a): VisOpt – The Solver behind the User Interaction, White Paper, InSol Ltd., Israel, May 1999

Barták, R. (1999b): Conceptual Models for Combined Planning and Scheduling, in Proceedings of the CP99 Post-Conference Workshop on Large Scale Combinatorial Optimisation and Constraints, Alexandria, USA

Barták, R. (1999c): Dynamic Constraint Models for Complex Production Environments, in Proceedings of the 1999 ERCIM/CompulogNet Workshop on Constraints, Paphos, Cyprus

Barták, R. (1999d): On the Boundary of Planning and Scheduling: A Study, in Proceedings of PLANSIG99 Workshop, Manchester, UK

Bosi, F., Milano, M. (1998): Enhancing CLP Branch and Bound Techniques for Scheduling Problems, Tech. Report DEIS-LIA-98-002, Università di Bologna

Brusoni, V., Console, L., Lamma, E., Mello, P., Milano, M., Terenziani, P. (1996): Resource-based vs. Task-based Approaches for Scheduling Problems, in: Proceedings of the 9th ISMIS96, LNCS Series, Springer Verlag

Buzzi, S., Lamma, E., Mello, P., Milano, M. (1997): Consistent Orderings for Constraint Satisfaction Scheduling, Tech. Report DEIS-LIA-97-001, Università di Bologna

Caseau, Y., Laburthe, F. (1994): Improved CLP Scheduling with Task Intervals, in: Proceedings of ICLP94, pp. 369-383, MIT Press

Caseau, Y., Laburthe, F. (1996): Cumulative Scheduling with Task Intervals, in: Proceedings of JICSLP96, pp. 363-377, MIT Press

Caseau, Y., Laburthe, F. (1997): A Constraint based approach to the RCPSP, in: Proceedings of the CP97 Workshop on Industrial Constraint-Directed Scheduling, Schloss Hagenberg, Austria

Crawford, J.M. (1996): An Approach to Resource Constrained Project Scheduling, in: Artificial Intelligence and Manufacturing Research Planning Workshop

Lamma, E., Mello, P., Milano, M. (1995): Temporal Constraint Handling in Scheduling Problems, Invited Paper at Intersymp95, Baden-Baden

Lever, J., Wallace, M., Richards, B. (1995): Constraint Logic Programming for Scheduling and Planning, in BT Technical Journal, Vol. 13 No. 1, pp. 73-81

Nareyek, A. (2000): AI Planning in a Constraint Programming Framework, in Proceedings of the Third International Workshop on Communication-Based Systems (CBS-2000), to appear.

Pegman, M. (1998): Short Term Liquid Metal Scheduling, in: Proceedings of PAPPACT98 Conference, London

Pool, D., Mackworth, A., Goebel, R. (1998): Computational Intelligence – A Logical Approach, Oxford University Press, Oxford

Simonis, H., Cornelissens, T. (1995): Modelling Producer/Consumer Constraints, in: Proceedings of CP95, pp. 449-462

Smith, A.W., Smith, B.M. (1997): Constraint Programming Approaches to Scheduling Problem in Steelmaking, in Proceedings of CP97 Workshop on Industrial Constraint-Directed Scheduling, Schloss Hagenberg, Austria

Srivastava, B., Kambhampati, S. (1999): Scaling up Planning by teasing out Resource Scheduling, Tech. Report ASU CSE TR 99-005, Arizona State University

Tsang, E. (1995): Foundations of Constraint Satisfaction, Academic Press, London

VisOpt System web site (1999), http://www.visopt.com

Wallace, M. (1994): Applying Constraints for Scheduling, in: Constraint Programming, Mayoh B. and Penjaak J. (Eds.), NATO ASI Series, Springer Verlag