Dynamic Constraint Models for Planning and Scheduling Problems

Roman Barták*

Charles University, Faculty of Mathematics and Physics,
Department of Theoretical Computer Science,
Malostranske namesti 2/25, 118 00 Praha, Czech Republic

bartak@kti.mff.cuni.cz
http://kti.mff.cuni.cz/~bartak/

Abstract. Planning and scheduling attracts an unceasing attention of computer science community. However, despite of similar character of both tasks, in most current systems planning and scheduling problems are solved independently using different methods. Recent development of Constraint Programming brings a new breeze to these areas. It allows using the same techniques for modelling planning and scheduling problems as well as exploiting successful methods developed in Artificial Intelligence and Operations Research. In the paper we analyse the problems behind planning and scheduling in complex process environments and we propose to enhance the traditional schedulers by planning capabilities to solve these problems. We argue for using dynamic models to capture such mixed planning and scheduling environment. Despite of studying the proposed framework using the complex process environment background we believe that the results are applicable in general to other (non-production) problem areas where mixed planning and scheduling capabilities are desirable.

1 Introduction

The real-life applicability and challenging complexity of planning and scheduling attract a high attention among researches in various areas of computer science. Traditionally, the planning and scheduling tasks are solved independently using different methods and technologies. The *planning task* deals with 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. Planning has been studied in Artificial Intelligence (AI) for years and the methods developed there, like the STRIPS representation [13] and Graphplan planning algorithm [6], are the core of many planning systems. Opposite to planning, the *scheduling task* deals with the exact allocation of activities to available resources over time respecting precedence, duration, capacity, and incompatibility constraints [7]. Operations Research (OR) has

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

a long tradition in studying scheduling problems and many successful methods to deal with the problem were developed there.

Recently, Constraint Programming (CP) attracts a high interest among both planning and scheduling community because of its potential for declarative description of problems with various real-life constraints. *Constraint programming* [2] is based on the idea of describing the problem declaratively by means of constraints, logical relations among several unknowns (or variables), and, consequently, finding a solution satisfying all the constraints, i.e., assigning a value to each unknown from its respective domain. It is possible to state constraints over various domains, however, currently probably more than 95% of all constraint applications deal with finite domains [18].

At the present time, scheduling is probably the most successful application area of CP [19] while application of CP to planning is not so spread [14]. The reason for this disproportion can be found in the conventional formulation of the constraint satisfaction problem that expects all the elements, i.e., all the variables and all the constraints, to be specified in advance. This is not an obstacle in scheduling tasks where all the activities are known beforehand, however, the plans are highly variable and it is impossible to predict which activities will be used in which combinations. Also, in some problem areas like complex-process environments we do not know all the activities in advance and the appearance of the activity depends on allocation of other activities to resources. In such a case the scheduler needs to be enhanced by some planning capabilities which complicate the constraint model.

In [4] we analysed the main features of static constraint models when applied to problems in complex-process environments and in [5] we proposed the mixed planning and scheduling framework that can be used to solve these problems. In this paper we survey dynamic constraint models for solving mixed planning and scheduling tasks. These models are applicable to scheduling tasks where generation of new activities during scheduling is required. We study expressiveness and efficiency of the models and we give a comparison of the models using the typical problems in complex-process environment. The described models were studied in VisOpt scheduling project [3] whose goal is to develop a generic scheduling engine for complex-process environments. Nevertheless, we believe that the results are applicable to other areas like transport problems and pure planning.

The paper is organised as follows. In Section 2, we specify the problem area and we list the typical problems of complex-process environments there. In Section 3, we explain the similarities and differences of planning and scheduling tasks and we describe how to mix both planning and scheduling into a single framework. Section 4 is dedicated to the description of constraint modelling of scheduling problems. We classify the scheduling constraints there, describe the models of time and sketch the difference between representation of resources and tasks. In Section 5, we overview three dynamic constraint models for solving mixed planning and scheduling problems. The paper is concluded by some final remarks.

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 or food industries. The goal of the VisOpt planning and scheduling project [3] is to develop a generic scheduling engine that can be customised easily for a particular environment via the description of resources, demands, 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) the activities necessary to satisfy the custom orders and other marketing requirements and to allocate (to schedule) the activities to the 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, we must consider compatibility and capacity constraints restricting which products and in what quantities can be processed, i.e., produced, moved, or stored together. Also the order of the 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 the resources.

Alternative processing routes, alternative production formulas and *alternative raw materials* are other typical features of the above mentioned industrial areas. In addition to the core products it is possible to produce some low quality products called *by-products*, typically during set-ups. The by-products can 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 the co-products must be scheduled because of the limited capacity of the 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., re-using of the by-products and the co-products as a raw material.

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 as well.

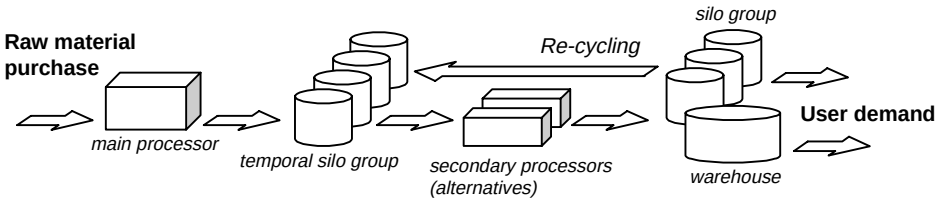

Fig. 1. A complex-process environment with re-cycling

In scheduling projects the users deal with finding no arbitrary schedule but an optimal schedule. Usually a makespan is used as the objective function [8,9,10]. 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 the 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 a good profit). The profit is measured here by the overall production cost and by the price of selling the products delivered according to the custom orders.

The solved problem hardly fits into any category of typical scheduling problems as used by Operations Research (OR) because many complex constraints 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 the dynamic characteristic when new activities appear during scheduling. Perhaps, it is closest to the group of resource constrained project scheduling problems [10,11]. Resource constrained project scheduling problem (RCPSP) is a generalisation of job shop scheduling [1] in which activities can use multiple resources, and resources can have capacity greater than one (more activities can be processed together). Nevertheless, the definition of RCPSP as well as of all other scheduling problems in CP still 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 section.

3 Planning vs. Scheduling

Although, planning and scheduling tasks seem very similar, they are defined differently and different solving technology is used. We first overview a conventional definitions of planning and scheduling and survey the traditional solving technologies.

Planning. The traditional AI 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 [16]. 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.

Conventional AI planning techniques use highly specific representation and algorithms but there is a pressure to use more general search frameworks like CP [14]. The advantage of such general framework is wider applicability and availability of ready-to-use methods. The specific features of a particular problem are then reflected at the modelling level only and not in the underlying search algorithms.

Scheduling. 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 [7]. 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.

Scheduling tasks are usually solved using techniques from OR and CP. Both frameworks expect the task to be specified fully in advance, i.e. all the problem variables and constraints must be known beforehand. Recently, new problem areas like complex-process environments use a partial specification of the problem that requires adding new variables and constraints during scheduling.

In industrial life, the boundary between planning and scheduling is shifted and both tasks are more similar that could be a source of confusion. There is a *marketing planning* that has nothing in common with the above described AI planning. The task of marketing planning is to prepare the demands for production using the information about the custom orders and the market forecast. The output of the marketing planning, the list of demands makes the input to the *production planning*. The definition of the production planning is closer to AI planning, the task is to prepare a plan of production using information like available stock, BOM (bill of materials), and demands. The plan consists of the list of activities that are usually assigned to factory departments. Finally, there is a *production scheduling* which allocates the activities from the production plan to available resources over time. Nevertheless, it is possible to introduce new activities during production scheduling if necessary.

As you see, the difference between production planning and production scheduling is fuzzier now; both tasks include generating of activities and their allocation to resources. The discrimination criterion is shifted to the resolution of resulting plan/schedule and to the different time horizon. While the production planning uses lower resolution (departments, days) and prepares plans for longer time period, the production scheduling prepares short-term high-resolution schedules (machines, minutes). The similarity of the production planning and the production scheduling brings us to the idea of using unified solving framework for both tasks. Nevertheless, there are other reasons to mix the planning and scheduling technologies.

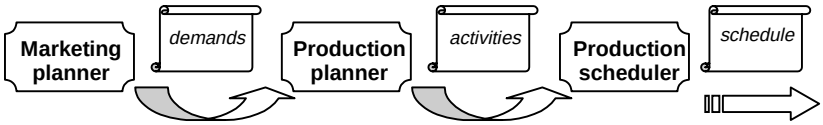

Fig. 2. A traditional view of planning and scheduling in industry with separate modules

Opposite to [17] and according to our experience we argue that in many current *Advanced Planning and Scheduling* (APS) systems the modules implementing the planner and the scheduler are independent. The separation of planning and scheduling is natural; the planner generates the activities and the scheduler allocates these activities to available resources. However, there are also disadvantages of such decomposition and these drawbacks become even more evident in some problem areas like complex-process environments.

First, backtracking from the scheduler to the planner is required if the clash[1] in the plan is found during scheduling or if the plan does not utilise the resources fully. Such backtracking is not desirable because it complicates the communication between the modules (the scheduler should inform the planner about the reason of backtracking) and it decreases the overall efficiency of the system. To restrict the number of backtracks we need a more informed planner which means the planner that uses similar information about the resources like the scheduler. However, this suppresses the advantage of low-resolution of planning. Another possibility to avoid backtracking is to postpone planning decisions until all necessary information is available. Planning with active decision postponement was implemented in the system Descartes [12] and we propose to postpone planning decisions even more to the scheduling stage.

More informed planner can prevent clashes during scheduling but it is not able to prepare plans in the problem area where the appearance of the activity depends on the allocation of other activities to the resources. There are several examples of such behaviour in complex-process environments. First, there are special set-up activities that must be inserted between two production activities [15] to ensure set-up of the machine. Second, there is a processing of the by-products that are produced typically during the set-ups. Some schedulers omit handling of by-products but this could be dangerous if the by-products may fill-up the stores used for regular products. Next, there is a re-cycling that is usually applied to by-products but it could be used with regular products as well, for example to clear the store. Finally, there is a non-ordered production. In some sense, the production planner could generate activities for non-ordered production but to prevent clashes it is more reasonable to postpone planning of non-ordered production until the scheduler provides information about spare capacity of the resources.

The discussions in the above paragraphs and sections justify our proposal of mixing the traditional planning and scheduling tasks into a single framework. Briefly

[1] Clash may be caused by choosing the bad alternative during planning which prevents allocation of activities to available resources.

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 simply a *production scheduler*. We expect to preserve the separate marketing planner that generates the basic demands for the production but the production scheduler is authorised to generate new activities for non-ordered production too.

The production scheduler consists of an *activity generator* (former planner) that generates the activities and an *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 3 for proposed structure of the mixed planning and scheduling system.

The communication between the generator and the allocator is simple via single activities. The generator introduces an 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 same production scheduler both for the production planning and for the production scheduling as described above. Naturally a different type of activities and different resources are used for production planning and for production scheduling but the overall solving mechanism is the same.

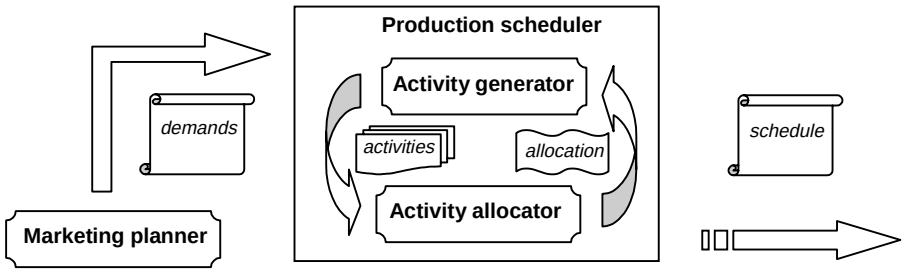

Fig. 3. A general framework for mixing planning and scheduling

4 Constraint Modelling

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 the constraints that define their nature and the relationships between them. The search code is kept

separate from this declarative model. Constraint Programming has a big advantage over other frameworks in declarative modelling capabilities. Some researches even claim that "Constraint programming represents one of the closest approaches computer science has yet made to the Holy Grail of programming: the user states the problem, the computer solves it" [E. Freuder, Constraints, April 1997]. The modelling capabilities of CP are really fascinating and the constraint models are very close to the description of real-life problems. This simplifies the maintenance of the models as well as the introduction of domain dependent heuristics necessary to solve large-scale problems. However, designing a stable constraint model that can be used to solve real-life large-scale problems is also the biggest challenge of current CP.

Scheduling is a typical application area of constraint programming and several standard constraint models to tackle scheduling problems can be identified. In this section we survey the features of these models that must be taken into account when deciding about the model for a particular problem area.

4.1 Constraint Classification

The constraints appearing in scheduling applications can be classified into several groups. We classify the constraints using their role in the scheduling problem into three categories: resource, transition, and dependency constraints. Such categorisation helps in choosing the right constraint model because different models handle better the constraints from different categories. Thus, we may choose the appropriate model more easily using the information about the spread of constraints in the proposed categories.

We concentrate on solving scheduling problems in complex-process environments here but we believe that such environments represent the typical problems in most scheduling applications. Thus we suppose that the proposed classification can be applied to other scheduling areas as well.

Resource constraints. The resource constraints describe the limits of the resource in single time point. A typical example of the resource constraint is the capacity constraint stating how many activities can be processed in parallel or how many items can be stored together etc.:

$$\forall Resource \ \forall Time \sum_{\substack{Activity \\ start(Activity) \le Time \le end(Activity)}} consumes(Activity, Resource, Time) \le capacity(Resource, Time)$$

Compatibility (incompatibility) constraint is another example of the resource constraint. The compatibility constraint states what activities can be processed together, i.e., in other words what activities/items are compatible. For example, if the *Activity1* cannot be processed together with the *Activity2* in the *Resource*, i.e., the *Activity1* is incompatible with the *Activity2*, then we can use the following compatibility constraint:

$$\forall Time \quad consumes(Activity1, Resource, Time) = 0 \ \vee \ consumes(Activity2, Resource, Time) = 0$$

While the capacity constraint is a “quantity type” constraint (how many?), the compatibility constraint can be seen as a “quality type” constraint (what?).

In many traditional scheduling problems the resource constraints are very simple, e.g., single-capacity resources are used only. However, these constraints are important when multiple-capacity resources like stores or batch processors are modelled. This is the case of complex-process environments.

Transition constraints. The transition constraints also restrict variables describing a single resource but opposite to the resource constraints they bind the variables from different time points. Typically, these constraints specify what future situations may follow the current situation. For example we may describe the transitions between the activities (thus transition constraints). If a machine set-up time is modelled using a special set-up activity (this is useful, if by-products are produced during the set-up) then the transition constraints naturally describe the insertion of the set-up activity between two production activities.

The transition constraints are typical for complex-process environments with many set-ups but they do not appear in many other scheduling problems.

Dependency constraints: The resources in a typical scheduling problem are hardly independent so we must also describe the relations between the resources in the model. We call such relation a dependency constraint. The dependency constraints bind variables describing different resources at perhaps different time points. A typical example of the dependency in production scheduling is a supplier-consumer dependency. It specifies the relation between the activity supplying an item and the activity consuming the item. A special case of the dependency is the precedence constraint in task-centric models (see below). 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 etc.

The dependency constraints are typical for many scheduling problems as they describe the relations between the activities belonging to a single task. Nevertheless, in problem areas like complex-process environments they may bind activities from different tasks as well.

Naturally, the above presented constraint classification depends on what objects are chosen as resources. In complex-process environments we use several resources like producers, movers and stores but it is also possible to add other resource types like workers and budget.

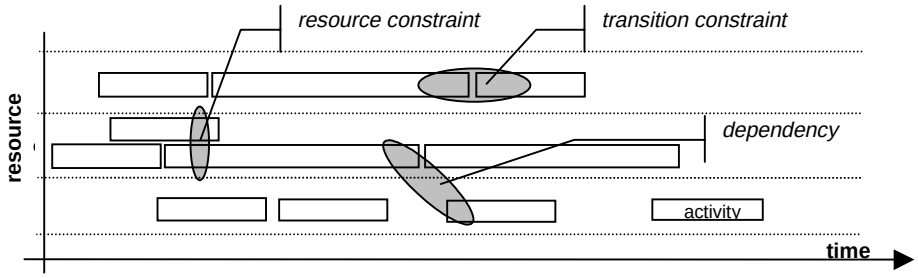

Fig. 4. Constraint categories in the Gantt chart

4.2 Modelling of Time

Both planning and scheduling tasks deal with time as one of the parameters. Therefore modelling of time is necessary in most planning and scheduling applications. In general we may distinguish between two different views of time: discrete time and event-based time.

Discrete Time. Perhaps the easiest way of modelling time is to divide the time-line into a sequence of discrete time intervals with the same duration. We call such intervals time slices. The duration of the time slice defines the resolution of the plan/schedule.

It is expected that the behaviour of the resource within the time slice is homogenous, i.e., the important events like the change of activity appear at the time point between two time slices only. Consequently, if we model activities using discrete time then the duration of the time slice must respect the duration of all the activities. More precisely, the duration of the time slice must be a common divisor of the duration of all activities[2]. If we work with activities that have no restrictions about their start and completion time then this requirement may lead to a huge number of time slices (high resolution) even if the duration of the activities is long (low resolution). Therefore, the discrete time model is used mostly in timetabling and in personal management applications where the time slices are specified directly by the problem area (like shifts in hospitals).

In the discrete time model the variables describe the situation either at the time points or at the time slices.

Event-based Time. Another view of the time line may capture only the important time points, so called events, when there is some change. We may either say that the duration between two consecutive events defines the activity or that the border

[2] We must synchronise the time slices in all resources (because of dependency constraints); thus the duration of all activities in all resources should be assumed.

between two consecutive activities is called an event. Therefore we speak about event-based time or activity based models.

In models that use event-based time we describe the situation of the resource by the processed activity. Each activity is characterised by a chunk of variables that should include the start time and the completion time (or duration) of the activity as well as the resource where the activity is processed.

Event-based view of time is preferable over discrete time when the density of events is not very large in comparison with the density of time points, i.e. when the ratio between the activity duration and the duration of time slice is large.

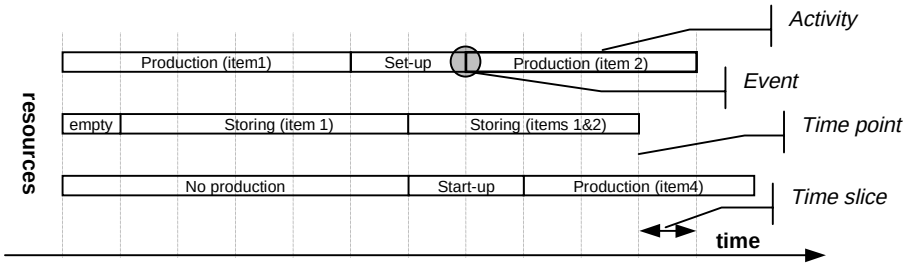

Fig. 5. Discrete time vs. event-based time in Gantt chart

4.3 Task vs. Resource Models

A final criterion that we use to distinguish between constraint models is the primary organisation of elements in the model. In particular we mean organisation of activities but we can organise time slices this way too. We distinguish between two main categories: grouping of activities per task or per resource [7].

Task-centric models. In many scheduling problems the activities are organised into tasks where the task is defined by a group of activities that must be processed to solve the task. In production scheduling the task corresponds typically to a custom order. In such case we speak about a production chain that defines the track of the items through the factory starting from raw material and finishing with the final product. Note that by the production chain we mean not only a sequence of activities but also a tree of activities or other graph structure (typically with a partial order defined among the activities using the precedence constraints). Generating the production chain is a planning task while allocating the activities in the chain to available resources is a scheduling task.

In the task-centric models, that are currently dominant in CP scheduling, the dependency constraints are preferred to the resource and transition constraints. We mean that it is easier to express the dependency constraints in such models as they describe the timing between the activities from single task typically using the precedence relation. It is more complicated to express the resource and transition

constraints a priori because they are dependent on allocation of the activities to the resources.

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 problems like set-ups, re-cycling and non-ordered production that make the task-centric model less applicable because it is not clear what tasks will be present in the schedule (a non-ordered production) and what activities will form the task (alternatives, set-ups, re-cycling).

Resource-centric models. Orthogonal to the task-centric model is the resource-centric model where the activities are organised per resources. In this model, we do not assign the activities to available resources but we order the activities in single resource in such a way that all the constraints be satisfied. In this model we describe the structure of the factory and the capabilities of the resources rather than particular tasks. The activities belonging to the task are grouped implicitly during the scheduling using the dependency constraints. Thus, the resource-centric model is re-usable for different sets of tasks and the same model can be applied to various sets of demands (tasks).

Resource-centric model concentrates more on expressing the resource and transition constraints because we know which activities belong to a given resource. The dependency constraints are used to synchronise the resources and their appearance depends on the ordering of activities in the resources. Thus, these constraints have a dynamic characteristic.

The resource-centric model is appropriate for problem areas like complex-process environments because it allows modelling set-ups, re-cycling, and non-ordered production. In this model it is natural to generate new activities during the scheduling so it is appropriate for solving mixed planning and scheduling problems.

We described the classification of models via grouping activities per task or per resource but the same classification can be applied to the time points/slices as well. However, when the discrete time models are used, the task should be linear because otherwise it is more complicated to model the task using a single time line.

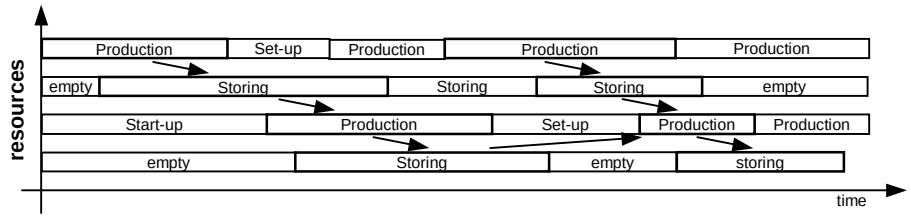

Fig. 6. Gantt chart is closer to the resource-centric model but we can identify the grouping per task there too

5 Dynamic Models

Conventional Constraint Satisfaction Problem (CSP) is defined statically; i.e., all the variables and all the constraints are specified in advance. Most methods for solving problems defined as CSP expect such static structure and they follow the schema:

1. Introduce variables.
2. Post constraints among the variables.
3. Label the variables respecting the constraints.

The main difference between the constraint satisfaction algorithms is in the labelling method; either they are extending a partial consistent labelling to a complete consistent labelling using constraint propagation or they are searching the space of complete (but inconsistent) labellings till the consistent labelling is found.

Because of the static nature of the constraint models, the task-centric model is usually preferred over the more dynamic resource-centric model. Unfortunately, as we described in Sections 2 and 3, to model complex-process environments we need some planning capabilities within the scheduler and this requires the dynamic characteristic of the constraint model when new entities (activities, constraints) are introduced during scheduling[3]. In the following paragraphs, we describe three main constraint models used in scheduling and we propose how these models can be extended to solve typical problems in complex-process environments.

5.1 Time-Line Model

The time-line model is a general method of describing dynamic processes using discrete time intervals and grouping time slices per resource. As mentioned in the

[3] We may use the static model to solve planning problems as well [4]. However, in such models all the possibilities must be captured which makes the model huge. This is because many variables are introduced to describe all the possible situations, but only few of them are really used. Also expressing the constraints is much more complicated and the value propagation is not very strong via such constraints.

previous section, it is possible to combine discrete time with grouping slices per task too. However, this combination is rarely used because it requires “linear tasks” only. Therefore we use discrete time and grouping per resource only in the following paragraphs. Such time-line model describes the factory in general because it concentrates on the specification of resources and it is independent of a particular set of tasks.

Variables. We describe the situation of the resources at each time slice (time point) using a chunk of variables. These variables may specify the processed activity, the quantity of stored items etc. We may use a unified description of each time slice for the resource, i.e., each time slice for a given resource is described using the same set of variables. Another possibility is to generate (some) variables in time slices dynamically, e.g., in case of store we may introduce a variable for an item’s quantity when we know that the item is stored in a given time slice. By distributing the variables into static and dynamic groups we may preserve the propagation power of constraints (see next paragraph) while keeping the memory consumption low. A good strategy is to define variables present in all time slices for a given resource as static (like the variable specifying the activity) while the variables whose appearance depends on the value of other (static) variables as dynamic (like the quantity of a processed item).

Remember that we are scheduling a fixed time period so we know the number of time slices. Consequently we know all the static variables in advance and we can also deduce their allocation in time. This simplifies capturing the initial situation of the resources as well as capturing the required future situations via setting the value or restricting the domains of variables.

Constraints. The knowledge about the structure of (static) variables enables us to introduce resource and transition constraints in advance and to exploit the power of constraint propagation. The resource constraints bind variables in single time slice and transition constraints bind variables from consecutive time slices. If any dynamic variable is included in the constraint then the constraint is posted as soon as all such variables are introduced to the system.

There remain the dependency constraints that express the supplier-consumer relations between the resources. These constraints bind variables from different time slices of different resources so they have dynamic nature here as their appearance depends on values of some variables. In particular, we can introduce the dependency constraint when we know which items are processed in the time slices.

Planning. The role of the activity generator (the planning module) is shifted a bit in the time-line model because, in fact, we do not generate activities here. The activity is introduced simply by assigning a value to the activity variable at the time slice. The planning module in the time-line model is responsible for the introduction of the dynamic elements, in particular of the dependency constraints.

Complex-Process Environments. The time-line model is a very general model that is able to capture all the problems in complex-process environments as presented in Section 2. The problems like by-products, re-cycling, non-ordered production, or alternatives are modelled naturally here. Unfortunately, the size of the model is very large when applied to real-life large-scale problems. This is caused by the formula for computing the slice duration using the duration of all the activities (see previous section). In particular, even if the duration of the activities is, say 60 and 61 minutes then the duration of the time slice is still 1 minute (the greatest common divisor of the activities' duration). Therefore many variables are redundant and, thus, we do not expect very good efficiency from this model when applied to complex-process environments.

5.2 Task-centric Model

A traditional constraint model for scheduling problems in the CP framework uses event-based time with grouping of activities per task. We call this model simply a task-centric model. This model is very popular among the scheduling community because of its static nature; all the elements are known in advance. Also, typically the resource constraints are very simple here and there are no transition constraints. Unfortunately, this is not the case in the complex-process environments so we propose here how to extend the conventional task-centric model to solve some of the problems in such environments.

Activities and variables. In the task-centric model, activities are used to describe the behaviour of the resources. Each activity is specified by a chunk of variables that include the start and the completion time of the activity (or duration) and the variable specifying the resource to which the activity is allocated.

In the static representation, we have the description of all the activities in advance, but in complex-process environments we need to introduce activities during scheduling. There are two ways how to establish such dynamic behaviour. First, we may use a fully dynamic representation where the activities are generated by the planning module. Second, we may estimate the maximal number of activities per task and instead of using the fixed activities we propose to use a shell for activity, i.e., we extend the variable set in the shell by the variable for the activity. By assigning a value to this variable we fill the shell by the activity. We prefer this second alternative, called semi-dynamic representation, because it preserves the advantages of the static representation, i.e., constraint propagation, and it is almost as powerful as the fully dynamic representation[4].

[4] In the fully dynamic representation, the number of activities in the task is not restricted at all; the decision about the activities is up to the planning module. In semi-dynamic representation we must decide about the maximum number of activities per task in advance which may complicate modelling of cycling and re-cycling.

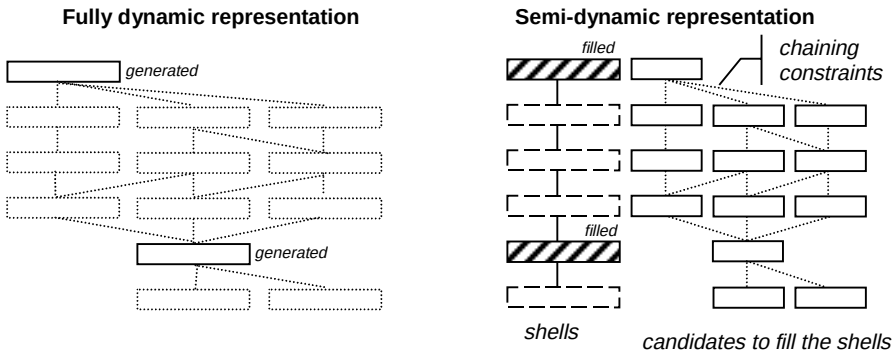

Fig. 7. In the fully dynamic representation (*left*), the planning module generates activities using the structure of all alternative production chains. In the semi-dynamic representation (*right*), the task of the planning module is to fill the prepared shells by activities using the chaining constraints (*dotted lines*).

Constraints. In the task-centric model it is easy to express the dependency constraints between activities of a single task. Usually, these constraints express the precedence relation between activities of the task, i.e., which activities must be completed before another activity starts. Using fully dynamic or semi-dynamic representation complicates a bit expressing these constraints because we do not know the activities in shells. In such case, the constraints are posted as soon as the activity variable is assigned.

In dynamic and semi-dynamic representations we may use another type of dependency constraint that binds activity variables. These constraints define the allowed production chains; in particular they are used to restrict the alternatives if some activity is known. Let's call these constraints *chaining constraints*.

Finally, there are resource and transition constraints that bind activities from different tasks. Again, these constraints are dynamic; i.e. they are introduced during scheduling when we know the allocation of activities to resources (when resource and time variables are assigned).

Planning. The dynamic representation of the task-centric model is a nice example of mixed planning and scheduling framework; the planning module introduces activities that are allocated by the scheduling module. During planning we use the information from the chaining constraints. The semi-dynamic representation can exploit the chaining constraints even more because the constraints reduce the domains of the activity variables automatically via constraint propagation.

The planning module is also responsible for the introduction of resource and transition constraints. In particular, it must identify the activities from different tasks that should be connected using these constraints. The planning module also decides about inserting special set-up activities.

Complex-Process Environments. The semi-dynamic representation was proposed with respect to solve typical problems in complex-process environments. There is no problem with *alternative production chains* in single task; the planning module chooses the alternative by filling the shells during scheduling.

As already mentioned, there is also no problem to use *set-ups*. We may simply include a set-up activity among other activities in the production chain [15]. We must just be careful to which production chain (task) the set-up activity is included. Typically, the set-up is inserted between two activities from different tasks. If there are no by-products then we may include it to the production chain of the second activity (or the first one, it does not matter). However, if any by-product is produced during the set-up then we recommend including the set-up activity to the production chain where the by-product is consumed. In case of co-products, we need even more advanced mechanism of sharing activities by two tasks. It is the responsibility of the planning module to identify the shared activities.

Finally, there is a *non-ordered production*. Unfortunately, to schedule non-ordered production we need to introduce new tasks during scheduling. This is not allowed in the semi-dynamic representation or we must prepare "empty" tasks in advance to be filled by activities of non-ordered production.

The task-centric model is perfect for environments driven by orders where the tasks do not interleave too much. Also, the resource and transition constraints should not be too complicated. Even after enhancing the model, the capabilities of the task-centric model to solve problems in complex-process environments are limited.

5.3 Resource-centric Model

Like the task-centric model, the resource-centric model uses event-based time but now the activities are grouped per resource. Therefore this model is closer to the description of real factory and it is independent of particular set of tasks.

We may use a static representation of this model when all the activities in the resource are known in advance. In such case the scheduling task is to order the activities in the resource respecting all the constraints. Naturally, the expressiveness of the static representation is not very high and a dynamic representation is more natural for this model.

Resource-centric model is orthogonal to the task-centric model so many techniques are shared between the models. However, we shall show that the resource-centric is more appropriate for complex-process environments as it can solve the typical problems in these environments more naturally.

Activities and variables. Again, we describe the behaviour of each resource using activities and each activity is specified by a chunk of variables. These variables include the start and the completion time (or duration), the process quantity and the links to dependent activities. Notice that we do not need a variable for resource because the activities are grouped per resource but we need links to supplying and consuming activities for dependencies.

Like in the task-centric model we may use either a fully dynamic representation where the activities are generated during scheduling by the planning module or a

semi-dynamic representation with shells. In the case of semi-dynamic representation, the upper estimate of the number of activities per resource must be computed. Remember that the scheduled period is fixed so we may compute this upper estimate using the scheduled duration and the duration of the activities (the shortest activity). Opposite to the task-centric model where this upper estimate may restrict the number of cycles, there is no restriction when the semi-dynamic representation is used here. Because the semi-dynamic representation exploits the power of constraint propagation, we prefer this representation here.

Constraints. In the resource-centric model, it is easy to express the resource and the transition constraints because we know which activities belong to the resource. In fact, we have the sequence of shells per resource and the resource and transition constraints are used to restrict the domains of activity variables.

There remain the dependency constraints that bind variables from the activities of different resources. These constraints are dynamic; i.e. they can be introduced when we know the allocation of the activities to the shells. Note that even if we know the activity in the shell, it is not easy to identify the dependent activities because the shells in other resources may not be filled yet and they may not be allocated in time.

Planning. In the semi-dynamic representation, the role of the planning module is shifted from generating activities to introducing dynamic constraints. In particular, the planning module is responsible for finding dependent activities, i.e., grouping activities per task. We may post the dependency constraints early when the activities in the shells are not known exactly and when the allocation of shells in the time is not precise. In this case a lot of potential connections is generated but only one of them will be selected later to form the dependency. This approach is close to the static representation. Another approach is to identify the exact dependent activity by the planning module. The disadvantage is that if we find later that the chosen activity is wrong we must backtrack to find another dependent activity. We propose something in-between; i.e. we post the dependency constraints when we know the activity in one of the dependent shells.

During posting the dependency constraints, the planning module uses the information about tasks to be scheduled. We mean, that the activities in the shells and the dependencies are not chosen “randomly” to satisfy the constraints but the planner prefers the activities to satisfy the demands.

Complex-Process Environments. The resource centric model has similar capabilities like the presented time-line model in modelling problems of complex-process environments. There is no difficulty to model set-ups and because the dependencies are generated during scheduling, there is no problem with by-products or co-products. Because the production chains (tasks) are not specified explicitly in advance we may choose the alternatives during scheduling. Finally, the scheduling is driven by user demands but we can introduce activities for non-ordered production without any difficulty.

The resource-centric model exceeds the task-centric model in problem areas with complicated resource and transition constraints and where scheduling of non-ordered

production is required. Depending on the resolution of the resulting schedule, it has lower memory consumption than the time-line model but it is also a bit complicated to express some constraints here. We believe that in general, the resource-centric model is the best model for complex-process environments.

6 Conclusions

In the paper we gave a survey of constraint models for scheduling problems and we proposed how to extend these models by adding some planning capabilities. This extension was motivated by real-life problems in the area of complex-process environments. We highlighted several such problems that are too complicated for traditional methods and we showed that the mixed planning and scheduling framework could capture these problems. We analyse the basic techniques of constraint modelling in scheduling problems and we studied three different constraint models. We argue for using a dynamic representation that prevails over the conventional static representation in areas where appearance of the activity depends on allocation of other activities to resources. Also, the dynamic representation makes the model more transparent and simplifies the constraints. As a result, we choose the resource-centric model that balances the efficiency and expressive power when applied to large-scale problems in complex-process environments.

The methods proposed in the paper are currently verified in the implementation of a generic scheduling engine for complex-process environments. This engine is being developed within the VisOpt scheduling project for InSol Ltd.

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 and to anonymous referees for useful comments.

References

1. Baptiste, P., Le Pape, C., Nuijten, W.: Constraint Based Optimisation and Approximation for Job-Shop Scheduling. Proceedings of the AAAI-SIGMAN Workshop on Intelligent Manufacturing Systems, IJCAI-95, Montreal, Canada (1995)
2. Barták, R.: On-line Guide to Constraint Programming. Charles University, Prague, `http://kti.mff.cuni.cz/~bartak/constraints/`
3. Barták, R.: VisOpt – the Solver behind the User Interaction. White Paper, InSol Ltd., Israel (1999), available at URL `http://www.visopt.com/`

4. Barták, R.: Conceptual Models for Combined Planning and Scheduling. *Proceedings of the CP99 Post-conference Workshop on Large Scale Combinatorial Optimisation and Constraints*, Alexandria, USA (1999) 2-14
5. Barták, R.: On the Boundary of Planning and Scheduling: a Study. *Proceedings of the Eighteenth Workshop of the UK Planning and Scheduling Special Interest Group*, Manchester, UK (1999) 28-39
6. Blum, A. L. and Furst, M. L.: Fast Planning Through Planning Graph Analysis. *Artificial Intelligence* 90 (1997) 281-300
7. Brusoni, V., Console, L., Lamma. E., Mello, P., Milano, M., Terenziani, P.: Resource-based vs. Task-based Approaches for Scheduling Problems. Proceedings of the 9th ISMIS96, LNCS Series, Springer Verlag (1996)
8. Caseau, Y., Laburthe, F.: Improved CLP Scheduling with Task Intervals. Proceedings of ICLP94, MIT Press, (1994) 369-383
9. Caseau, Y., Laburthe, F.: Cumulative Scheduling with Task Intervals. Proceedings of JICSLP96, MIT Press (1996) 363-337
10. Caseau, Y., Laburthe, F.: A Constraint based approach to the RCPSP. Proceedings of the CP97 Workshop on Industrial Constraint-Directed Scheduling, Schloss Hagenberg, Austria (1997)
11. Crawford, J.M.: An Approach to Resource Constrained Project Scheduling. Artificial Intelligence and Manufacturing Research Planning Workshop (1996)
12. Joslin, D. and Pollack, M.E.: Passive and Active Decision Postponement in Plan Generation. Proceedings of the Third European Conference on planning (1995)
13. Fikes, R. E. and Nilsson, N. J.: STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence* 3-4 (1971) 189-208
14. Nareyek, A.: AI Planning in a Constraint Programming Framework. *Proceedings of the Third International Workshop on Communication-Based Systems* (2000), to appear
15. Pegman, M.: Short Term Liquid Metal Scheduling. Proceedings of PAPPACT98 Conference, London (1998) 91-99
16. Pool, D., Mackworth, A., Goebel, R.: Computational Intelligence – A Logical Approach, Oxford University Press, Oxford (1998)
17. Srivastava, B. and Kambhampati, S.: Scaling up Planning by teasing out Resource Scheduling. Technical Report ASU CSE TR 99-005, Arizona State University (1999)
18. Tsang, E.: Foundations of Constraint Satisfaction. Academic Press, London (1995)
19. Wallace, M.: Applying Constraints for Scheduling. *Constraint Programming*, Mayoh B. and Penjaak J. (Eds.), NATO ASI Series, Springer Verlag (1994)