Dynamic Constraint Models for Complex Production Environments

Roman Barták

Charles University, Faculty of Mathematics and Physics
Department of Theoretical Computer Science
Malostranske namesti 2/25, 118 00 Praha 1, Czech Republic
e-mail: bartak@kti.mff.cuni.cz

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 usually 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. Currently, scheduling is the most successful application area of constraint programming.

In the paper we analyse the problems behind planning and scheduling in complex production environments. We give a survey of three conceptual models developed to model such environments. We discuss their industrial background and compare their advantages and disadvantages. The models were studied within the VisOpt project whose goal is to developed a generic scheduling engine applicable to various complex production environments. However the proposed conceptual models can be applied to other (non-production) problem areas where joined scheduling and planning capabilities are required.

Keywords: scheduling, planning, modelling, constraint satisfaction, optimisation

1 Introduction

Planning and scheduling attract high attention among researches in various areas of computer science. Sometimes, there is confusion what problems planning and scheduling deal with and what are the similarities and the differences. Roughly speaking, *planning* deals with finding plans to achieve some goal, i.e., finding a sequence of actions that will transfer the initial world into one in which the goal description is true. Planning has been studied in Artificial Intelligence for years and the methods developed there, like the STRIPS representation and planning algorithm [14], are the core of many planning systems. Industrial planning usually means finding a plan of production for a longer period of time. From this point of view, scheduling can be seen as a more detailed planning for shorter period of time. More precisely, *scheduling* deals with the exact allocation of resources to activities over time respecting precedence, duration, capacity and incompatibility constraints. Operations Research has a long tradition in studying scheduling problems and many successful methods to deal with the problem were developed there.

Recently, Constraint Programming (CP) attracts high interest among both planning and scheduling community [21] because of its potential for declarative description of problems with various real-life constraints. *Constraint programming* [6] is based on 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 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 [20]. And among them, the scheduling problems are the most successful application area [16,21].

“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]. However, relying on this statement is also the biggest danger of real-life projects based on constraints because generic constraint satisfaction and optimisation algorithms are not capable to tackle efficiently large-scale industrial problems brought by real life without additional help. Therefore, the modelling stage of the projects is crucial for the efficiency of the implementation. Also, integration of methods developed and checked by Artificial Intelligence and Operation Research people can improve efficiency of constraint systems using specialised global constraints and more advanced search techniques.

In the paper we give a survey of basic techniques, concepts and mechanisms developed within the project of generic scheduling engine. The task was to prepare a generic model capable to capture various scheduling problems in complex production environments. Nevertheless, the results are applicable to non-production areas, like transport problems, as well. We should also highlight that we combine both planning and scheduling capabilities in the models. We summarise the results of the first stage of the project where three conceptual models to represent the joined planning and scheduling problem were studied. We also give a description of industrial background behind the models. Opposite to [4], where static constraint models were described, we concentrate on dynamic representation of the conceptual models now.

The paper is organised as follows. In chapter 2 we specify the problem domain and its specialities. In chapter 3 we explain the similarities and differences of planning and scheduling problems and we argue for solving both problems within single system. In chapter 4 we sketch the basic ideas behind the models and we outline the differences between static and dynamic representation of the models. In subchapters 4.1 – 4.3 we describe the three conceptual models studied in the project. The paper is concluded with some final remarks.

2 Problem Area

In the VisOpt scheduling project [5] we deal with complex production areas like plastic, petrochemical, chemical or pharmaceutical industries. The task is to develop a generic scheduling engine that can be customised easily for particular environment via description of resources, initial situation and expected 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. Some resources can handle several tasks at a time (during *batch production* in a bath or in an oven etc.) and task can be scheduled to multiple alternative resources. Also the order of tasks processed by the resource is not arbitrary but the processed task influences what tasks can follow. Consequently, we must follow the *transition patterns* and assume *set-up times* between tasks as well. The processing time is usually variable and there is defined *working time* when the tasks can be processed in respective resources.

Alternative processing routes, *alternative production formulas* and *alternative raw materials* are other typical features of above mentioned industry areas. In addition to core products it is possible to produce *by-products*, that can be used as raw material in further production, or *co-products*, that can be sold as an alternative to the ordered product. Processing of both by-products and co-products must be scheduled because of limited capacity of warehouses where all types of products are stored. Also the compatibility constraints describing which products can be processed, i.e., produced, moved or stored together, must be considered. Last but not least there is 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.

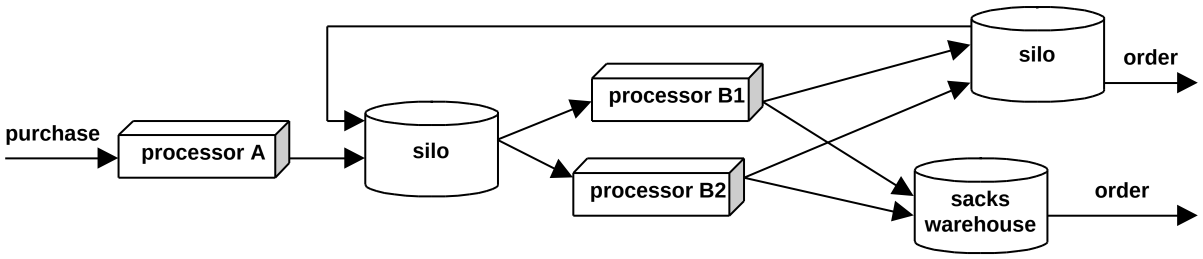

Figure 1 (example of complex production environment)

Typically, the production is not driven by custom orders only but it is possible to schedule production for store, i.e., not everything that is produced was ordered. Therefore the system should manage some planning capabilities as well.

The task is to produce most profitable schedule or a schedule close to optimum. Looking for an optimal schedule or for a schedule close to optimum is not new for scheduling community but note that most current scheduling systems use the makespan as an objective function. The idea of minimising makespan, i.e., minimising 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 production environments where expensive set-ups must be considered. Therefore a more general notion of cost function should be used as the objective function. Note that we do not describe handling cost in detail in the paper and this will be a subject of another paper.

The solved problem is close to the group of resource constrained project scheduling problems (RCPSP). RCPSP [10,13] is a generalisation of job shop scheduling [3] in which tasks can use multiple resources, and resources can have capacity greater than one (more tasks can be processed together). Nevertheless, the RCSPS is still defined by a set of tasks and a set of resources, and both sets are known in advance. Unfortunately, this is not our case because when alternative processing routes, by-products and production for store are assumed then the set of all tasks/activities is not known beforehand.

3 On the edge of Planning and Scheduling

In most current APS (Advanced Planning and Scheduling) systems the planning and scheduling components are implemented separately in different modules and the communication between the modules is limited. Such decomposition seems natural because planning and scheduling deal with a bit different tasks. In particular, planning tackles the problem of finding plans to achieve some goal, i.e., finding a sequence of actions that will transfer the initial world into one in which the goal description is true. While scheduling deals with the exact allocation of activities to resources (or resources to activities) over time respecting precedence, duration, capacity and incompatibility constraints. On the other side both planner and scheduler must consider the same environment limits and constraints even if the planner uses a more general view while the scheduler deals with the details.

A typical structure of the system with separate planning and scheduling modules is following. The planner generates plans, i.e., sets of activities, and the scheduler assigns the activities from the plan to resources and determines the exact start and completion times. If the scheduler finds that it is impossible to schedule the current set of activities, i.e., the plan is too tight, then it backtracks to the planner that prepares another plan.

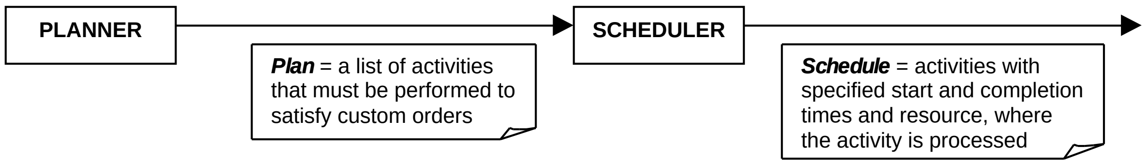
PLANNER
SCHEDULER
Plan = a list of activities that must be performed to satisfy custom orders
Schedule = activities with specified start and completion times and resource, where the activity is processed

Figure 2 (separate planning and scheduling)

The separation of planning and scheduling stages has some disadvantages that cannot be ignored. First, if the communication between both modules is limited, and usually it is limited, then the system behaves like a generate-and-test algorithm whose efficiency is known to be not very good. By limited communication we mean that scheduler “does not inform” about the conflicts the planner so the planner generates next plan blindly. This is usually the case because the planner does not care about the details that the scheduler must consider. On the other hand, if the plans produced by the planner are too free, i.e., easy to schedule, then we can expect that the resulting schedule will be less profitable than it could be. This is because for example more products could be made with available resources. Consequently, the planner should prepare the profitable plans that can be scheduled, but this requires the planner to have the detail knowledge about the factory that is not a feature of typical planner.

In the previous chapter we sketched some problems of complex scheduling environments that require joined planning and scheduling stages. For example, alternative processing needs the planner to choose the alternative path or it is up to the scheduler to choose the right alternative. We prefer the second case because it gives the scheduler more freedom and the alternative can be chosen in such a way that that other scheduled activities are respected. Consequently, the scheduler should have some planning capabilities. Another example that requires planning is using by-products and cycling. Again, the planner does not usually handle these problems because of too many details but they can dramatically influence the production, e.g., if there are too many by-products filling the stores then we cannot continue in production of regular products. Finally, there is a production of store that needs planning. However, note that we produce for store for example if there is no “ordered production” and it is cheaper to continue current activity than stooping the producer. Naturally, such production for store is a result of current schedule because it depends of the schedule of ordered production. Consequently, this task cannot be solved at the planner level only but at the scheduler level even this is a typical planning task.

To summarise above discussion, we propose the scheduler to handle typical planning task, in particular, to generate sequences of actions satisfying the orders. However note that it does not mean omitting the planning at all. Usually, industrial planning means finding a plan of production for a longer period of time, in particular deciding what should be produced but no how to produce it. We expect to preserve this planning stage but we prefer to encapsulate all the tasks concerning “how to produce” problem into single scheduling engine.

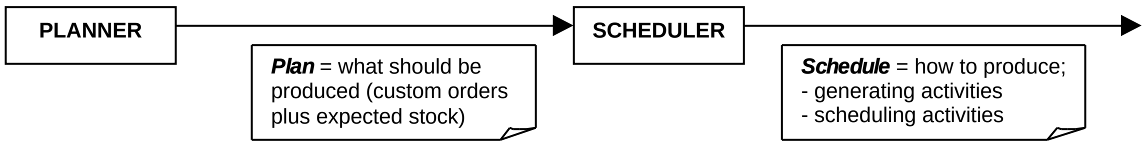
PLANNER
SCHEDULER
Plan = what should be produced (custom orders plus expected stock)
Schedule = how to produce;
- generating activities
- scheduling activities

Figure 3(our view of planning and scheduling)

4 Conceptual Models

Scheduling can be characterised as the problem of assigning resources to given set of tasks/activities in such a way that some constraints are satisfied and some goals are achieved. The constraints may regard both the activities and the resources; the goals may concern the optimal use of the resources or the optimal accomplishment of the tasks. According to discussion in previous chapter we slightly modified this traditional definition of scheduling problem and we allow generating of activities during scheduling. In particular, the scheduling is initialised by some set of activities and during scheduling other activities necessary for processing the initial activities are introduced to the system, i.e., planning is performed in run-time.

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. The search code is keep separate from this declarative model. This is exactly what we have done by designing the conceptual models describing the resources and dependencies between resources. But note that the entities and constraints can be introduced dynamically during scheduling.

In the project we studied three conceptual models with two different views of time. The time-line model expects time to be discrete, i.e., we are jumping from one time point to another. For example, we can model some situation from minute to minute or from hour to hour, treating each minute or hour as a different time point. The second view of time expects event-based time, i.e., time steps don't have to be uniform; we consider time steps between interesting events, in particular between changing activities in the resource. We studied two conceptual models that use event-based time, namely order-centric and resource-centric models. These conceptual models differ in the way of managing dependencies between activities.

In [4] we described the static representation of the conceptual models. This representation expects all entities and constraints to be introduced before the scheduling starts. This is a traditional representation of problems solved using CP because user can use constraint satisfaction packages without modification. A complicate description of constraints in case of many alternatives is the main disadvantage of static models for combined scheduling and planning. Because alternative processing routes are typical for our problem area we propose to use dynamic representation where the variables and constraints are introduced during scheduling. In this representation we join the planning stage that is responsible for introducing activities (variables and constraints) to the system with scheduling stage that schedules the activities, i.e., it finds exact start and completion times and values of other activity parameters. In the remaining part of the paper we concentrate on description of conceptual models with specification of basic entities and constraints.

4.1 Time-Line Model

The first conceptual model that we studied in the project is a time-line model. Time-line model 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 (point between two slices) we describe the situation of each resource using several variables, for example, what is currently produced or stored. It is assumed that the behaviour of resource is homogeneous between two consecutive time points (within time slice), i.e., the key events like changing activity occur only at the edge of two consecutive time slices.

The duration of time slices must be defined according to the duration of activities that can be processed by the resources. Because of slice homogeneity, it is required that the activity change is at a time point only. Consequently, each activity takes one or more time slices (but no fragment of single slice). We can compute the duration of time slice for given resource as a common divisor of duration of all activities that can be processed by the resource.

Also we need to synchronise different resources so we prefer the time slices to have the same duration in all resources. This refines the slices further because the duration of time slice should be equal to a common divisor of duration of all activities in all resources. On the other side we prefer smaller number of slices (i.e., longer duration of slice) because it implies smaller number of variables to assign a value, i.e., less work to do. Together, the *duration of slice* is computed as a greatest common divisor of duration of all activities in all resources.

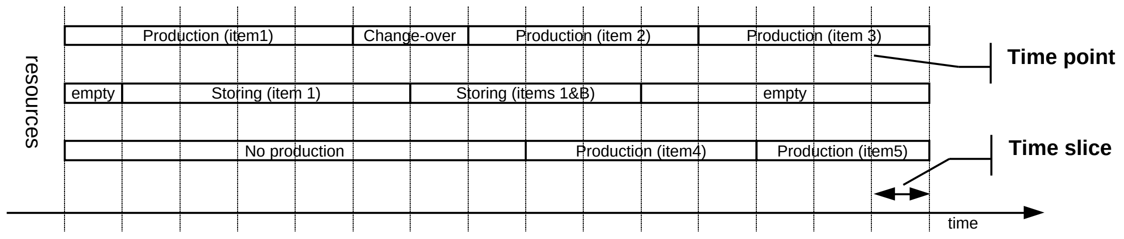

Figure 4 (a time-line model)

Now, when the time slices are defined we can describe the situation of resources at time points. The situation at each time point is described simply by 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 resource etc. Then, we can introduce constraints between these variables that describe the resource features, like compatibility and capacity constraints.

***Capacity constraint*:**

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

Sum of quantities of all procesed items does not exceed the capacity of the resource.

***Compatibility constraint*:**

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

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

In most cases, the current situation of resource is influenced by the previous situation as well as by other resources. Again, there is no problem to define constraints between variables from different time points and even from different resources. For example, in case of store, the current quantity of the item is influenced by the previous quantity, by new supplied quantity (item is supplied to the store) and by the consumed quantity (item is removed from the store).

Supplier/consumer constraint:

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

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

The above supplier/consumer constraint also justifies the necessity to use a common “clock frequency” for all resources which allows us to synchronise resources easily.

In [4] we presented a static representation of the time-line model as a matrix of variables where one axis corresponds to variables describing resources and the second axis corresponds to time points (the number of time points specifies the scheduled duration). This representation requires normalising the resource description in all time points.

resource	variable name	variables							
Processor A	state								
	…								
Processor B	state								
	…								
…									
Store	Item 1 quantity								
	Item 2 quantity								
	…								
	time points	1	2	…					N

Figure 5 (the representation of the time-line model)

We can use the matrix in the dynamic representation as well. The dynamic representation has the advantage of introducing the constraints during scheduling dynamically. In the static representation [4] we require all the constraints to be “fired” before the scheduling starts. This requirement complicates typically the description of the supplier/consumer constraints. For example if there are more activities that can be processed in given time point and each activity uses different input, i.e., different supplier/consumer constraint, then we must encapsulate the information about activity in the constraint. Naturally, this makes the constraints more complicated. In dynamic representation, we can introduce the respective supplier/consumer constraint when the activity in the time point is known. In the mixed planning and scheduling algorithm, the planning component is responsible for decisions about activities so when the planner assigns the activity to the time point, it introduces all necessary constraints to the scheduling component.

In the time-line model, there is no problem to represent initial situation as well as to capture any required future situation. We can simply set the values of variables in time point 0 to describe the initial situation and we can restrict the domains of variables in future time points to capture desired future situations. Naturally, such domain restriction is valuable because it can navigate the scheduling and decrease the size of search space. Also, we can use the same mechanism to implement heuristics derived from user’s experience like specification of minimal stored quantities of items etc.

The time-line model is very general and because of its nature it is capable to capture all scheduling situations. Unfortunately, the model has one big disadvantage that prevents its usage in real large-scale problems. This drawback is hidden in the definition of 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 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. As a consequence, there are many time points and therefore a huge number of variables to label. Therefore, we can expect not very good efficiency from the time-line model if applied to real-life 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 small number of variables to describe the situation,
- the ratio between time slice duration and scheduled duration is higher, i.e., either the scheduled duration is not very long or the duration of time slice can be longer.

4.2 Order-centric Model

Order-centric model[1] is a traditional model for job-shop scheduling [3] where event-based time is used (event = changing activity in the resource). It is based on idea of defining the chain of activities necessary to produce the ordered item or, generally, to satisfy the order. The goal is to schedule such production chains for all orders respecting the resource limits. In Figure 6 we show an example of production chain. You may notice that by the notion of production chain we mean not only a linear sequence of activities but also, for example, a tree of activities with the root corresponding to the final product or the order.

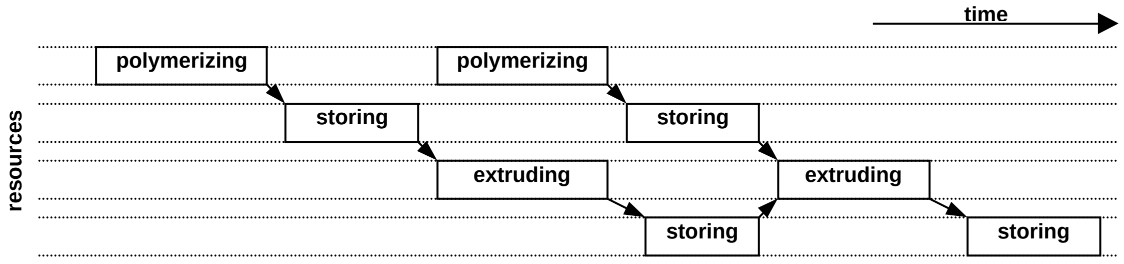

Figure 6 (a production chain in the order-centric model)

In the order-centric model, we describe activities using set of variables. Typically, start and completion times (or duration) are assigned to each activity as well as the resource where the activity will be processed. The value of these variables is being found during scheduling. Other parameters, like processed quantity are usually constant because we know what quantity of the item is processed in advance (the quantity is derived from the ordered quantity).

In general, there are two groups of constraints used in the order-centric model to restrict the combinations of values in variables. First group consists of constraints between activities within single production chain. We call these constraints *supplier/consumer dependencies* because they describe the movement of items between resources (supplying and consuming). Depending on the parameters describing activities, there may be timing and quantity constraints between activities. The timing constraint specifies the order of activities in the production chain, i.e., the precedence of former activity before later activity in the chain. The quantity constraint is used if variable quantities of the item can be produced. This constraint binds the produced and consumed quantities in consecutive activities in the production chain.

Timing constraint:

end(Activity 1) + transport_time(resource(Activity1),resource(Activity 2)) = start(Activity 2)

if Activity 1 foregoes Activity 2 in the production chain.

Quantity constraint:

quantity(Activity 1,Item) = quantity(Activity 2,Item)

for each item produced in Activity 1 and consumed in Activity 2.

The second group of constraints describes features and limitations of resources; therefore we call them *resource constraints*. These constraints are evoked if the activity is assigned to respective resource only so in the static model we must include the parameter for resource in the constraint while in the dynamic model we can fire these constraints when we assign the activity to a resource.

[1] Sometimes, the model is called task-based model [8] but we prefer the notion of order-centric model as we assign a production chain to the order.

There may be a constraint binding the resource with activity duration (it is used if the processing time is variable). Also there are usually a capacity (cumulative) and compatibility constraints restricting the set of activities that can be processed together in single resource. We can use these constraints to model disjunctive scheduling as well (if we set different specification to each activity then only one action can be processed at time).

Compatibility constraint:

specification(Activity 1) ≠ specification(Activity 2) & resource(Activity 1) = resource(Activity 2)

end(Activity 1) ≤ start(Activity 2)   start(Activity 1) ≥ end(Activity 2)

If Activity 1 and Activity 2 are assigned to the same resource and they are incompatible (i.e., they have different specification) then they cannot be processed together.

Capacity constraint:

$$T \quad R \qquad \sum_{\substack{Activity \\ start(Activity)\le T\le end(Activity)\quad \& resource(Activity)=R}} consumed(Activity, R) \le Capacity(R)$$

At each time the capacity of the resource is not exceeded.

The order-centric model can be represented as a set of production chains where each production chain is represented as a list of activities. Note that even if we use more complex structure of the production chain, like a tree, it is still possible to sort activities in some way to get the list. For example, we can sort the activities using earliest start time and, in case of clash, use the latest start time or random order.

In [4] we proposed the static representation of the model using matrix as shown in Figure 7. In such case we should normalise both the length (to the longest chain) and the sets of parameters (to the superset of all parameters). Then, some variables are void during scheduling (we may set them to zero).

	Activity 1	Activity 2			…			Activity M
prod. chain 1	Resource							
	Start							
	End							
	…							
prod. chain 2	Resource							
	Start							
	…							
…					…			
prod. chain N								

Figure 7 (the representation of the order-centric model)

In dynamic representation where the constraints and activities can be introduced during scheduling it is probably better to use lists of activities to represent production chains. Then there is no problem with different length of the production chains and the activities may be specified using different sets of variables.

In the dynamic representation we expect that only first (last) activity in the production chain be initialised. All other activities are generated by the planning component during scheduling. This approach also solves the problem with alternative processing routes in the static representation where normalisation is required [4,17]. Now, the planning component chooses the right alternative, i.e., the continuation of the production chaing, during scheduling. Similarly, we can solve the problem with set-up times by introducing set-up activities when necessary. 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 scheduling.

The next problem is the processing of *by-products*, i.e. the production of non-ordered items that can be re-used in future production. Now, the situation is more complicated because it requires two or more production chains to interact in a more advanced way. In particular, if a by-product is produced in one production chain then another production chain can use this by-product as raw material. To model by-products we need sharing of activities between two or more production-chains as Figure 8 shows.

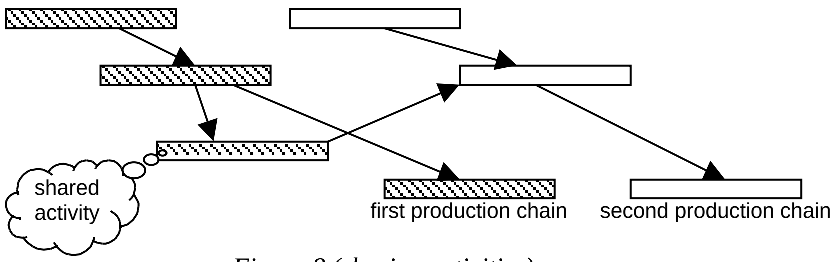

Figure 8 (sharing activities)

In the proposed dynamic model, the planning component is responsible for introducing activities to the scheduler. Therefore, the planner can check if the shared activity is already introduced and then it either introduces a new sharing activity or it re-uses already present activity to join two production chains.

Finally, there is a problem with the *production for store*, i.e., non-ordered production. Unfortunately, this problem cannot be solved in pure order-centric model because there are no production chains for non-ordered items. It seems that if we integrate the planner into scheduler (as proposed above) then the planner could generate activities for production of non-ordered items. However, this requires the planner to find the “gaps” in production, i.e., to check results of scheduling in a more detail way which complicates significantly the communication between planning and scheduling components. Notice that production for store is different from processing of by-products because the processing of by-products is initialised by one of the production chains.

Order-centric model is a perfect model for order-driven production and usually, a static representation is used for this model. We showed that by using the dynamic representation we could solve easily the problems with alternative processing routes and by-product. However, if scheduling of production for store is required then this model is of little help.

4.3 Resource-centric Model

Resource-centric model is similar to order-centric model in the way of using activities and event-based time. Now, we are working with the list of activities per single resource and the chain of activities per order is handled implicitly by means of dependencies between resources.

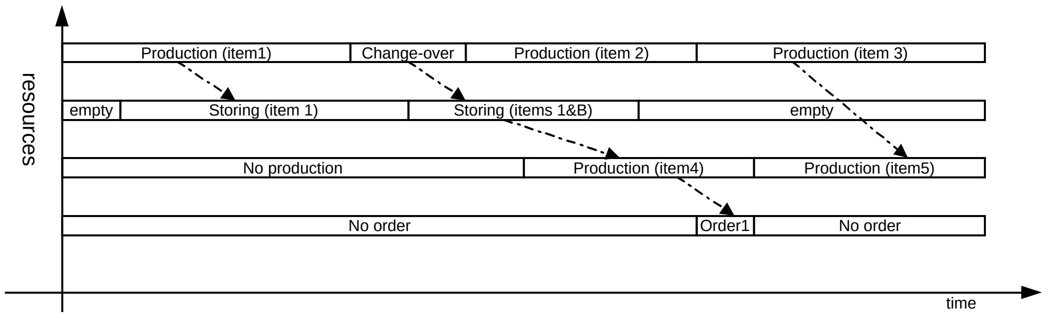

Figure 9 (a resource-centric model)

Resource-centric model is more appropriate for the description of a factory than order-centric model. This is because we concentrate on specification of resources and this specification is independent of actual set of orders. This is similar to the time-line model, where we also describe resources primarily.

Each resource in the model is described using the set of activity types[2] that can be processed by the resource and using resource constraints. The parameters of activity (activity type) are the same as in the order-centric model with two exceptions:

- we do not need the parameter identifying the resource for action because each action is assigned to the resource implicitly,
- we need parameters identifying supplying and consuming activities.

The resource-centric model can be represented as a set of resources where each resource is described by the list of activities. The length of this list depends on the actual schedule so dynamic representation, where activities are introduced during scheduling, is natural. We expect that the list of activities per resource be initialised by single activity describing the starting situation. The remaining activities are generated by the planning component using the transition schema defined for the resource and the actual partial schedule. The transition schema describes which activities can follow other activities.

In the resource-centric model it is easier to express *resource constraints* because we know which activities belong to the resource. Of course, the planner introduces the resource constraints as soon as respective activities are generated. The set-up time can be naturally modelled using set-up activities that are part of the transition schema.

There is absolutely no problem with alternative production routes and with by-products. Alternative processing is handled implicitly because when activity is scheduled then respective supplying and consuming activities must be scheduled as well (see below). Consequently, alternative processing routes emerge from the resulting schedule. By-products are handled in the same way as other products because, again, everything that is produced must be consumed. Visibly, we can also schedule the production without orders but if any

[2] Because we describe the abstract activity we use the notion of activity type. During scheduling we use activity types to generate real activities. The relation between activity type and activity is the same as the relation between class and object in OOP.

order is present then it is used as a consumer of some products so we must schedule the resources in such a way that the ordered products are available at specified times.

One question remains to answer only: how to model the *supplier/consumer dependencies*? Because we know the structure of the factory, we know how the resources are connected. Consequently, for each item we can identify the group of supplying activities and the group of consuming activities from respective resources. Now we may add a list of variables to each supplying activity specifying the quantities transferred to respective consuming activity and vice versa. The parameter *supplied(X,Y)* in the resource X represents the supplied quantity from the resource X to the resource Y and the parameter *consumed(X,Z)* in the resource X represents the quantity consumed by the resource X from the resource Z. Then the supplier/consumer dependency consists of three constraints:

- the sum of supplied quantities is equal to the produced quantity and the sum of consumed quantities equals to the overall consumed quantity,

$$\sum_{X\quad \mathrm{consumer\_of(A)}} \sup plied(A,X) = produced(A)$$

$$\sum_{X\quad \mathrm{supplier\_of(B)}} consumed(B,X) = consumed(B)$$

- the quantity in A supplied to B equals to the quantity in B consumed from A,

$$supplied(A,B) = consumed(B,A)$$

- if the interchanged quantity is greater than 0 then the time difference between the end of supplying activity and the start of consuming activity is equal to the transport time (because of previous constraint we can use both *supplied* and *consumed* parameter).

$$supplied(A,B)>0 \qquad end(A)+transport\_time(A,B) = start(B).$$

Supplier/consumer constraints are introduced to the scheduler when planner generates respective activities. In particular, when the planner introduces a new activity to the system then it introduces respective consuming and supplying activities as well or uses activities already present in the system as suppliers or consumers.

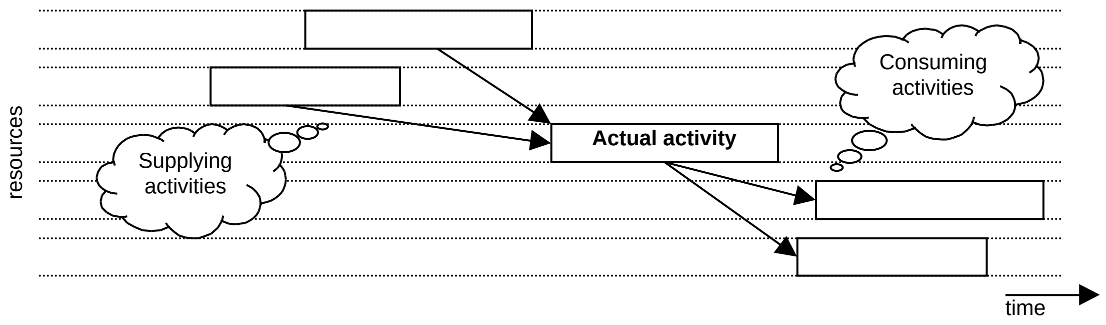

Figure 10 (supplier/consumer dependency)

The resource-centric model is general similarly to the time-line model because it is based on the description of resources rather than on the description of orders. It is also more efficient than the time-line model because we can use different scheduling resolution[3] in different resources. Consequently, less number of variables is introduced to the system. Finally, the resource-centric model can process orders but it does not rely on orders completely. So, it can be used to manage both planning and scheduling tasks.

[3] The resolution of scheduling is defined by the minimal duration among activities of the resources.

5 Final Remarks

In the paper we give a survey of three conceptual models for combined planning and scheduling in complex production environments. Opposite to [4] we concentrate on the dynamic representation here that allows introducing constraints during scheduling. We think that dynamic representation is more natural than static representation in case of combined planning and scheduling when the activities are generated during scheduling. At least, it simplifies the description of constraints and, consequently, makes the model more transparent.

In the time-line model the differences between static and dynamic representation are not significant, mainly because of the static nature of the model (we know the number of time points in advance). Perhaps, the static representation is slightly better for the time-line model because in allows propagation from the scheduling component to the planning component naturally. But the constraints are more complicated in the static representation than in the dynamic representation

The order-centric model is a traditional model of job-shop scheduling. Usually static representation is used for this model but if alternatives should be modelled then we propose the dynamic representation that is more transparent. Nevertheless, the order-centric model is still not capable to exploit fully the planning capabilities and the production for the store cannot be modelled.

The resource-centric model is probably the best model for combined planning and scheduling. This model is more appropriate for the factory specification because it concentrates on the description of individual resources rather than on production chains. It can also be used for scheduling of non-ordered production. We propose to use the dynamic representation of the resource-centric model because it enables more transparent description of constraints.

In the paper we present the concepts of generic scheduling engine. In particular we describe how the scheduling problem can be modelled using constraints. We choose the dynamic representation of the resource-centric model as the most promising model for complex production environments. The time-line model can also be used but we expect worse performance. The capabilities of the order-centric model are too limited to model complex production environments. The next research will cover the search procedures both for the planning component of the systems (what activities should be introduced) and for the scheduler (what values should be assigned to variables).

6 Acknowledgements

Author’s work has been supported by the Grant Agency of the Czech Republic under 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.

7 References

[1] Baptiste, P., Le Pape, C.: 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, 1995

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

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

[4] Barták, R.: Conceptual Models for Combined Planning and Scheduling, submitted to CP99 Workshop on Large Scale Combinatorial Optimisation and Constraints

[5] Barták, R.: VisOpt – The Solver behind the User Interaction, White Paper, InSol Ltd., Israel, May 1999

[6] Barták, R.: On-line Guide to Constraint Programming, http://kti.mff.cuni.cz/~bartak/constraints/

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

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

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

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

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

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

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

[14] Fikes, R. E., Nilsson, N. J.: STRIPS: A new approach to the application of theorem proving to problem solving, in: Artificial Intelligence Vol. No. 3-4, pp. 189-208

[15] Lamma, E., Mello, P., Milano, M., Temporal Constraint Handling in Scheduling Problems, Invited Paper at Intersymp95, Baden-Baden, August 1995

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

[17] Pegman, M.: Short Term Liquid Metal Scheduling, in: Proceedings of PAPPACT98 Conference, London, 1998

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

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

[20] Tsang, E.: Foundations of Constraint Satisfaction, Academic Press, London, 1995

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