Towards Mixed Planning and Scheduling

Roman Barták[1]

Charles University, Department of Theoretical Computer Science
Malostranské náměstí 2/25, 118 00, Praha, Czech Republic
email: bartak@kti.mff.cuni.cz

Abstract. The conventional wisdom and practice is that planning and scheduling tasks are solved separately using different methods and approaches. However, recent development in industrial planning and scheduling demands for mixing both tasks to allow modelling of wider class of problems.

The purpose of this paper is to present a framework for mixing planning and scheduling tasks within single system. We analyse traditional views of planning and scheduling and we highlight the drawbacks of separating both tasks when applied to modelling complex process environments. We give some real-life examples where the mixed approach helps to model the problems and we propose a generic framework for such mixture. We also argue for using constraint programming as the underlying solving technology and, finally, we describe some constraint models based on the proposed framework.

Although, we concentrate on planning and scheduling in complex process environments we believe that the results contribute to both planning and scheduling communities in general.

1 INTRODUCTION

It is a common practice that planning and scheduling tasks are solved separately using different methods. 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. Moreover, the possible sequences of actions must respect the limitations of the world. Planning has been studied in Artificial Intelligence (AI) for years and the methods developed there, like the STRIPS representation [8] and the Graphplan planning algorithm [5], 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 [6,7]. Traditional scheduling methods developed in Operations Research (OR) are now being absorbed by Constraint Programming (CP) technology (and vice versa) that provides better declarative modelling capabilities [17].

Recent development shows that such strict decomposition between planing and scheduling is not desirable in some problems and that using planning techniques in scheduling and vice versa may contribute both to richer modelling capabilities and to higher efficiency of the systems. Convergence may be observed on both sides of the border. The planning community tackles problems typical for scheduling like planning under resource constraints [10] and new algorithms were developed here using the constraint programming technology [9] that is more tributary to scheduling. Nevertheless, to be fair we should also mention that there exists reverse movement [14] asking for removing resource scheduling from planning algorithms. Scheduling community is a bit self-contained too but there also exist feelings about the necessity to include some planning capabilities to scheduling algorithms [2]. Last but not least, the developers of constraint satisfaction algorithms listen more to the requirements of dynamic planning tasks by introducing concepts like Dynamic Constraint Satisfaction [16] and Structural Constraint Satisfaction [11].

In industrial applications the border between planning and scheduling is fuzzier than in academics and the discrimination criterion is shifted to a different level. Both industrial planning and scheduling deal with generation and allocation of activities; the difference between planning and scheduling task is shifted here to the resolution level of the resulting plan or schedule. While industrial planning deals with the task of finding “rough” plans for longer period of time and uses department etc. as the basic resource, the scheduling task is to prepare a detail schedule for individual resources, like machines and workers, with higher time resolution

The similar character of industrial planning and scheduling brings the idea of using single approach that can be applied to both areas. In the paper we propose such a framework for close co-operation between planning and scheduling modules. The basic idea is not very complicated: the planner introduces new activities to the system and these activities are immediately allocated to resources by the scheduler. The new contribution here is the instant allocation of

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

activities after their introduction. Note that we do not require allocating the activity completely but restricting the possible "positions" of the activity by means of constraint propagation. This prunes the search space of the planner as well and allows us to detect the clashes soon. Moreover, the scheduler may ask the planner to introduce new activities if the situation (the current partial schedule) requires them. Such behaviour of the scheduling engine was motivated by existence of real-life problems going beyond the horizon of conventional static scheduling and requiring the dynamics of planning.

The rest of the paper describes the proposed framework in detail. In Section 2 we specify the problem area and we outline the main difficulties. In Section 3 we overview the conventional definition of planning and scheduling tasks and we give examples where the strict separation of both tasks is too restrictive. Section 4 is dedicated to description of the basic structure of mixed planning and scheduling system. In Section 5, we look at the engine behind our framework and we describe how to express our ideas in terms of constraint programming technology. We conclude with summary of the paper and with description of early experience with the implementation.

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 task is to schedule most profitable production for fixed period of time.

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

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

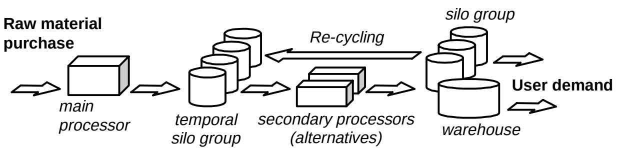

Figure 1. An example of complex-process environment

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

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 be able to introduce new activities during scheduling to "fill the gaps".

3 TRADITIONAL VIEW OF PLANNING AND SCHEDULING

Let's look at the traditional definition of planning and scheduling tasks first.

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. 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 [12]. 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 [6,7]. 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.

As mentioned above, the border between planning and scheduling is fuzzier in industrial life so it is not surprising that there exist systems providing both planning and scheduling functionally. Nevertheless, in such Advanced Planning and Scheduling (APS) systems, the modules implementing the tasks are still strictly separated following the above described conventional view of planning and scheduling. This decomposition seems natural because both tasks deal with a bit different problems (generation vs. allocation of activities) and different methods from different areas are used to solve the tasks. Also, the interface between modules implementing planning and scheduling is well defined; the planner generates a list of activities first and, then, push these activities to the scheduler that allocates them to available resources. Figure 2 describes the flowchart of such system with strictly separated planning and scheduling modules.

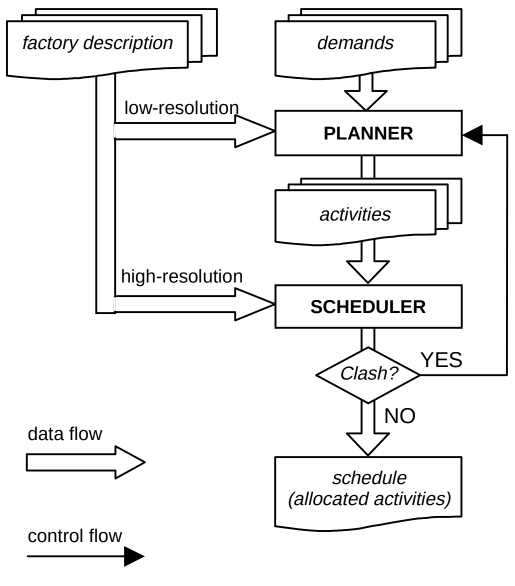

Figure 2. Separate planning and scheduling

3.1 Difficulties of separate planning and scheduling

The structure of APS system with separate planning and scheduling modules is satisfactory when the control and data flow through the system is linear, i.e. when we do not backtrack from the scheduler to the planner. In general such backtracking is not forbidden but, of course, it is not desirable because it decreases efficiency and complicates the interface between the planner and the scheduler.

We see two different reasons for backtracking from the scheduler to the planner. First, there is a possibility of clash in the plan, which prevents the scheduler to allocate the activities to available resources. The clash may be caused by choosing a bad alternative during planning. The second reason for backtracking could be a low-profitable schedule that may occur when the plan does not utilise the resources fully. Because the scheduler is not able to repair the plan alone (there is no activity "re-generator" in the scheduler) it must asks the planner for help. Moreover, the scheduler should inform the planner about the reason of backtracking and the planner must be able to repair the plan accordingly.

To reduce the number of backtracks or to eliminate them at all we may use a more informed planner that generates conflict-free enough-profitable plans. However, this requires usage of higher-resolution data and taking into account the typical scheduling constraints [10]. Actually, such informed planner absorbs the scheduler or, at least, it solves part of the scheduling task.

Another possibility to avoid backtracking is to postpone planning decisions until enough information is available. Decision postponement is a popular approach in the planning community; its active version using constraint programming was implemented in the Descartes planning system [9]. However, in current systems such decision postponement keeps within the planning context. What we propose is to extend the decision postponement as long as to the scheduling stage.

Before we describe the proposed mixed planning and scheduling framework let's look at other problems from complex-process environments that motivate mixing planning and scheduling. All these real-life problems have one common feature: the appearance of some activity depends on allocation of other activities to resources. Naturally, such problems cannot be solved by more informed planner and they require closer co-operation between planning (introduction of activities) and scheduling (allocation of activities).

Set-ups and transitions. Scheduling set-ups and transitions is a crucial problem in complex-process environments because of their considerable cost and duration. It is impossible to predict appearance of set-ups/transitions before the activities are allocated to the resource so we need to introduce special set-up/transition activities during scheduling like in [13].

This is necessary especially if by-products result from such activities.

Consumption of by-products. Many current scheduling systems do not care about by-products resulting from set-ups and transitions but in complex-process environments this is not desirable. First, we can use the by-products in further production as a raw material; second, the by-products may fill up the stores for final products. Unfortunately, until we know that the by-product appears (and this is during set-ups or transitions typically) we cannot introduce activities for processing the by-product.

Production for store. In some plant configurations, it is cheaper to continue in production rather than stopping the machine when all the ordered products were finished. Decision about such non-ordered production could be done at the planning level; i.e. the planner generates activities for non-ordered production. However, if the resulting plan is too ambitious then it may cause more clashes in the schedule and, consequently, more backtracking to the planner. Therefore, it seems more appropriate to delegate the decision to the scheduler which may "fill the gaps" in the schedule by activities producing for store.

4 MIXED PLANNING AND SCHEDULING

In previous paragraphs and sections we argued for mixing planning and scheduling components to solve some problems that conventional separate planning and scheduling systems cannot tackle. We also sketched the basic ideas behind such mixed framework, in particular introduction of activities during scheduling and postponing planning decisions till scheduling. Actually, both these methods describe the same thing from two different views: postponement of planning decisions is realised (partially) by introduction of activities during scheduling.

We shall describe the structure of the mixed planning and scheduling system using the notions of production scheduling that is our problem area. However, we believe that the same structure is applicable to other planning and scheduling areas like transportation problems as well.

The system consists of two components: activity generator (former planner) and activity allocator (former scheduler). You may see the relation between them in Figure 3.

The *activity generator* is responsible for introduction of new activities to the system. It may use the information about allocation of already posted activities when deciding about new activity. Also, if the activity generator "is not sure" about the next activity it may introduce a slot for activity and the real activity will be filled in this slot later by the activity allocator. Finally, the activity generator is responsible for posting constraints among activities, e.g., it ensures that production activity is connected to supplying and consuming activities etc. The mechanism of activity slots and the possibility to introduce constraints describing relations among (not-yet known) activities realises the decision postponement.

The *activity allocator* deals with the allocation of the activities to available resources over time. In general, the activity allocator decides about the values of activity attributes like start and completion time, the resource processing the activity etc. In fact, it can even decide about the activity itself if activity slot is used during scheduling. The activity allocator can also ask the activity generator to introduce new activity if it is missing and it may inform the generator about conflicts. Finally, it should be said that we might use the decision postponement strategy during activity allocation as well. Now, we do not choose the unique value for activity parameter immediately but we successively remove values from the variable domain that are inconsistent with values of other variables. By introduction of new activities and constraints we may restrict the domains further until they become singletons or conflict is detected. This technique is called constraint propagation and it is explained in detail in the following section.

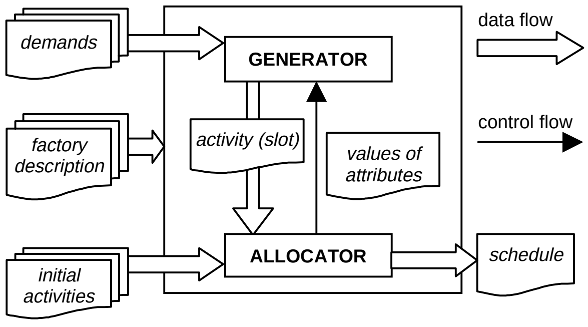

Figure 3. The structure of mixed planning and scheduling system

The scheduling (or is it planning?) is initiated by the list of demands, in production scheduling these demands corresponds to custom orders, and by some initial activities that may describe the initial state of the resources or requested future activities like maintenance. This generality of input data structure allows us to use the system in conjunction with traditional planners that generate activities in advance. The activities are posted to the activity allocator immediately while the demands are placed in the activity generator. Then, the activity generator uses the information about demands, about available resources, and about allocation of already posted activities (about the attributes' values of activities being allocated) when deciding about the introduction of new activity.

4.1 Benefits of proposed architecture

We proposed the mixed planning and scheduling framework to solve the problems in complex-process environments primarily. By allowing the dynamic introduction of activities during scheduling we are able to model problems like set-ups and transitions, processing of by-products and re-cycling, scheduling non-ordered production, or choosing alternative processing routes using the information about the current (partial) schedule. Naturally, the capabilities of the system depends on implementation of individual modules and on choosing the model of given problem [1]. We can identify some general advantages of the proposed framework.

Generality. The proposed architecture is very general and depending on the implementation of activity generator and allocator we get various planning and scheduling systems. We can use it to design generic planners where the role of the activity allocator is suppressed to solving planning constraints only. Or we can design the traditional scheduler where (almost) all the activities are inputted and only few activities are introduced during scheduling (for example to model set-ups).

Modularity. Note that the generator and the allocator are still separate modules so we may combine various planning and scheduling techniques. We could even use the same architecture to implement system with separate planner and scheduler (if the planner does not care about the attributes' values computed by the scheduler).

Formal interface. The important thing about the architecture is that it formalises the interface between the planner and the scheduler. The planner post the activities and (some) constraints among the activity attributes to the scheduler while the scheduler returns (partial) valuation of activity attributes.

Early detection of clashes. In general we cannot avoid backtracking during scheduling due to search nature of most solvers. Because we allow successive introduction of activities, we may detect the conflicts earlier so we backtrack sooner and it is not necessary to completely re-plan after the clash. Moreover, the activity generator may use the information about the reason of the clash directly; in particular it can find which constraints are violated.

Active decision postponement. The proposed architecture allows active decision postponement both in planning (via using the activity slots and constraints among not-yet known activities) and scheduling (via partial labelling of attributes).

Single factory description. From the modelling point of view, it is nice that we use single factory description for both planner and scheduler. We may still use parts of the description dedicated to planner or to the scheduler but now the information is available to the other module as well via closer co-operation between the modules.

5 APPLYING CONSTRAINTS FOR PLANNING AND SCHEDULING

When proposing the mixed planning and scheduling framework we assumed constraint programming to serve as the underlying engine behind the solvers. *Constraint programming* [15] 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.

At the present time, scheduling is probably the most successful application area of CP [17] while application of CP to planning is not so spread [12]. The reason for this disproportion can be found in the conventional static 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 the 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.

In the proposed framework, we are using constraint satisfaction technology in the scheduling module to evaluate attributes of activities primarily. The planning module uses the constraint engine via posting activities to the scheduler and via accessing the attributes' values. Naturally, the dynamic nature of the system, where activities (and thus the variables) and constraints are introduced during scheduling, must be considered when choosing the constraint solving package.

There exist two main approaches to solving CSPs (Constraint Satisfaction Problems); one based on constraint propagation, i.e., removing inconsistent values from variables' domains, second based on local search, i.e., altering complete but inconsistent labelling of variables towards (more) consistent labelling. Because of dynamic nature of mixed planning and scheduling, it is more complicated to use local search (we do not have a complete set of variables and constraints beforehand) but as already mentioned the constraint propagation is a valuable technique especially for the implementation of active decision postponement. We propose using CLP (Constraint Logic Programming) as the underlying programming environment as it allows adding new constraints during search as well as removing the constraints upon backtracking. Almost all CLP systems use constraint propagation techniques to solve the constraints.

5.1 Constraint classification

Before we describe how to model mixed planning and scheduling problems let's survey the classification of constraints in scheduling and resource-constrained planning problems. We proposed this classification in [3] where examples of particular constraints are given.

According to the role of the constraint in the problem we may classify the constraints into the following groups:

- *resource constraints*, that capture limitations of the resource in given time (like capacity),
- *transition constraints*, specifying available transitions between the activities in single resource (like set-ups), and
- *dependencies*, capturing relations between the activities of different resources (like supplier-consumer relations).

The Figure 4 shows where constraints of particular type can be found in the Gantt chart displaying the schedule.

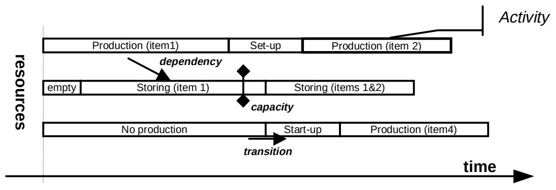

Figure 4. Constraint classification in the Gantt chart

Naturally, this classification depends on what objects are chosen as resources. Note also, that constraints of particular type may not appear in some scheduling problems or the complexity of constraints of different type may be different. For example, in many scheduling problems there are no transition constraints and the resource constraints are very simple (single-capacity resources). We propose to use the information about constraints of particular type when deciding which constraint model is appropriate for a particular problem.

5.2 Constraint models

The proposed architecture of the mixed planning and scheduling provides enough flexibility to solve various problems but the quality (efficiency) of the resulting system is highly dependent on how the problem is modelled.

In [1] we studied three models used in scheduling applications, namely time-line, task-centric, and resource-centric model. In [3] we give guidelines how to choose among these models and how these models can be applied to mixed planning and scheduling problems.

Timetabling or **time-line model** uses discrete time divided into time slices (usually equally distributed) and, then, it describes the situation at each time slice. This model is less appropriate for pure planning as it enforces the resource allocation. However, it can be used in a mixed planning and scheduling environment without troubles. In fact, we do not need a separate activity generator here because all the time slots are known beforehand (a time slot is a time slice in particular resource). Consequently, the main role of the system is to fill these slots be activities respecting all the constraints. Note that because of equal distribution of time slices, it is possible that some (most) activities will occupy several consecutive slots as Figure 5 shows. Consequently, the number of time slots could be much larger than the number of activities and a huge number of variables and constraints must be introduced to model the problem. Thus, this model is less appropriate for large-scale problems, explanations can be found in [3]. On the other side, because all the variables and constraints are posted beforehand we may use popular and very efficient local search algorithms here.

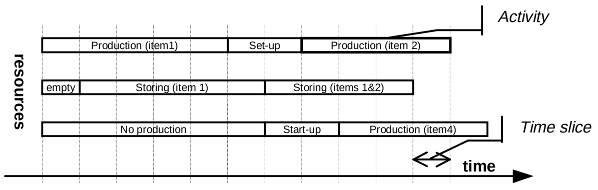

Figure 5. A time-line model with activities going through several time slices.

The second group of constraint models for scheduling problems is based on activities rather than on time slices. Activity-centric models use event based time and they seem to be better suited for modelling large-scale problems because of smaller memory consumption (the number of activities is usually much smaller than the number of time slots in production scheduling). Depending on the activity grouping we distinguish between two types of activity-based models, namely task-centric and resource-centric models.

If the activities are grouped per task or per order (in production environment) we are speaking about **task-centric** or **order-centric** model. This model is currently the most widely used constraint model in the production scheduling but in [1,3] we showed that it is less appropriate to model complex-process environments due to limited modelling capabilities. In the task-centric model it is natural to express supplier-consumer dependency constraints that bind activities of single task. Typically, these constraints are expressed in the form of a precedence relation between activities. However, it is more complicated to express complex resource and transition constraints because until we know the allocation of the activities to the resource, it is not clear which activities should be connected using such constraint. Moreover, the task-centric model has restricted capabilities when modelling processing by-products and non-ordered production.

Because of the above mentioned difficulties of the task-centric models we turned our attention to activity grouping per resource. We call such models **resource-centric models**. In the resource-centric model the capabilities of individual resources are captured rather than the production chains corresponding to given orders. Thus, it is natural to express the resource and transition constraints here because it is known which activities belong to a given resource. However, expressing the dependencies is more complicated here because it is not clear which activities from different resources are related.

We prefer to use the resource-centric model in the complex-process environments because it can describe all the typical problems of the area (capabilities of the task-centric model are limited) and it is less memory consuming than the time-line model. The comparison and arguments for choosing a particular model can be found in [1,3].

5.3 Model representation

When a particular constraint model is chosen, the question is how to implement it. It is possible to follow directly the architecture of mixed planning and scheduling systems proposed in Section 4 which suggests to generate activities dynamically during scheduling. Such **fully dynamic representation** has two main advantages. First, there is no restriction about the number of activities to be generated (typically, in pure planning the number of activities in the plan is not known). Second, the constraints are kept in a readable form because they are introduced together with the activities so no complicated triggers are necessary. However, the experimental implementation showed the drawbacks of the dynamic representation. It is possible that the activity appears twice or more times in the system and it is not easy to identify such duplicates and to merge them. The duplicates are generated because there are several reasons to introduce new activity. The activity is introduced as:

- a follower of known activity in single resource,
- a supplier of known activity, and
- a consumer of known activity etc.

The second problem is limited constraint propagation caused by dynamic introduction of new activities.

Because the scheduled duration is given in our problem area it is possible to estimate the number of activities in the schedule. Therefore we also studied the conventional **static representation** of the models, which has the advantage of exploiting fully the constraint propagation. In particular the method of scheduling alternative activities proposed in [4] looks promising to model complex problem environments. This method requires all the alternative activities to be generated in the form of a process plan.

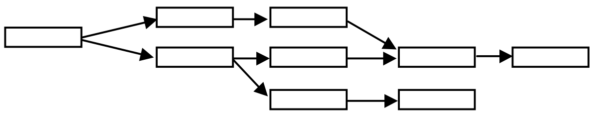

Figure 6. A process plan with alternatives. Each rectangle corresponds to an activity and the plan corresponds to the path from the leftmost to the rightmost activity.

Unfortunately, when the number of alternatives is very high, like in complex-process environments, then the process plan is huge. Another disadvantage of the static representation is necessity to use conditional constraints with complicated triggers. The trigger fires the constraint when it is known that the activity is present in the schedule. This has another drawback – the propagation through conditional constraints is not so powerful due to disjunctive character of such constraints.

Because both fully dynamic and static representations have significant drawbacks when applied to large-scale real-life scheduling problems in complex process environments we turned our attention to **semi-dynamic representation** based on slots. We generalised the notion of slot used in time-line model by unsticking the slot from a fixed time period. Then the slot can be described as an empty shell that is being filled by an activity during scheduling. Because we can estimate the number of slots in the schedule (remind that the schedule duration is fixed) we can generate the slots in advance. Naturally, it is possible that some slots remain empty after the scheduling or, alternatively, they are filled by some void activity.

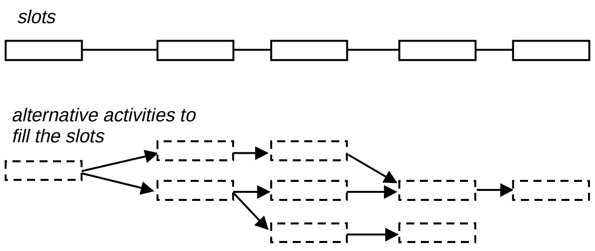

Figure 7. A slot chain. Its length is equal to the longest chain of alternative activities that fill the slots.

Common activity attributes like start time and duration can be moved to the slot so it is possible to post the constraints among them immediately and, thus, to exploit the constraint propagation. Activity specific attributes like quantities of processed items are introduced as soon as the activity in the slot is known. Another nice feature is that we can use constraint propagation to decide about the activity in the slot if special activity attribute is assigned to each slot. Consequently, we need no special mechanism for activity generation.

The proposed slot representation consists of a static part (common attributes of activities, attribute specifying the activity in the slot, and constraints among them) that is introduced before the scheduling starts and a dynamic part (activity specific attributes and dynamic constraints) that is posted during the scheduling. This helps to exploit the constraint propagation while keeping the dynamic constraints in a simple form. Note that there are two reasons for the constraint to be dynamic: first, the constrained variables are not posted yet (this is the case of constraints involving the activity specific attributes), second, the set of constrained variables is not known even if the variables are already posted. In the first case, the dynamic constraint can be introduced as soon as all the attributes are generated, i.e., when the activity in the slot is known. In the second case, it is necessary to look for variables involved in the constraint during scheduling which is the main difficulty of the slot representation. Efficient handling of such dynamic constraints is the main part of our current research.

6 CONCLUSIONS

In the paper we describe a framework for mixing planning and scheduling. We concentrate more on description of motivation for such framework, on highlighting general features and benefits and on explaining the technology behind rather than on specification and experiments with particular implementation.

Nevertheless, the proposed framework is currently being implemented as part of the generic scheduling engine within the VisOpt project [18]. First experience confirms our expectation about general capabilities of the framework; there is no difficulty to model all the typical problems in complex-process environments. This is a big advantage over other scheduling systems that are not able to tackle such complex industries. The experiments with the implementation also show some interesting results concerning efficiency of the system. It is necessary to find balance between active decision postponement and memory consumption especially when applied to large-scale problems. Also, the implementation indicates that the fully dynamic model, where all the variables and constraints are introduced dynamically, does not exploit the power of constraint propagation and thus the active decision postponement becomes more passive. Consequently, it is better to use semi-dynamic representation where as much as possible activities (activity slots) are posted beforehand (the role of activity generator is suppressed). This observation illustrates why most current scheduling systems deal with the static problems only.

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 discussions concerning the scheduling applications and for drawing my attention to some existing applications in the area.

REFERENCES

[1] 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

[2] 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

[3] Barták, R.: Dynamic Constraint Models for Planning and Scheduling Problems. *Proceedings of ERCIM/CompulogNet Workshop on Constraints*. LNAI Series, Springer Verlag (2000), to appear

[4] Beck, J.Ch. and Fox, M.S.: Scheduling Alternative Activities. *Proceedings of AAAI'99*, USA (1999) 680-687

[5] Blum, A. L. and Furst, M. L.: Fast Planning through Planning Graph Analysis. *Artificial Intelligence* 90 (1997) 281-300

[6] 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)

[7] 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)

[8] 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

[9] Joslin, D. and Pollack M.E.: Passive and Active Decision Postponement in Plan Generation. *Proceedings of the Third European Conference on Planning* (1995)

[10] Koehler, J.: Planning under Resource Constraints. *Proceedings of 13th European Conference on Artificial Intelligence*, Brighton, UK (1998) 489-493

[11] Nareyek, A.: Structural Constraint Satisfaction. *Proceedings of AAAI-99 Workshop on Configuration*, 1999

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

[13] Pegman, M.: Short Term Liquid Metal Scheduling. *Proceedings of PAPPACT98 Conference*, London (1998) 91-99

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

[15] Tsang, E.: *Foundations of Constraint Satisfaction*. Academic Press, London (1995)

[16] Verfaillie, G. And Schiex, T.: Solution Reuse in Dynamic Constraint Satisfaction Problems. *Proceedings of the Twelve National Conference on Artificial Intelligence AAAI* (1994) 307-312

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

[18] VisOpt web pages, `http://www.visopt.com/`