Integrated Modelling for Planning, Scheduling, and Timetabling Problems*

Roman Barták[1], Hana Rudová[2]

[1] Charles University, Malostranské námìstí 2/25, Praha, Czech Republic
e-mail: bartak@kti.mff.cuni.cz

[2] Masaryk University, Botanická 68a, Brno, Czech Republic
e-mail: hanka@fi.muni.cz

Abstract. Planning, scheduling, and timetabling problems are typically solved using different techniques despite the fact that all these areas seem very similar and they are definitely connected. In this paper we propose a generic model covering scheduling and timetabling problems as well as some problems from planning under time and resource constraints. Our model is motivated by real-life problems and our goal is to cover various aspects of these problems in the common model.

Keywords: scheduling, timetabling, planning, modelling, integration

1 Introduction

Despite the fact that planning, scheduling, and timetabling are closely related problems, they are solved using rather different techniques. Sometimes there is even confusion to which category a given problem belongs because the notions of planning, scheduling, and timetabling have slightly different meaning in different environments. For example in industry, the notion of planning often means long-horizon, low-resolution form of scheduling, i.e., activities are assigned to departments rather than to machines and timing is in months rather than in minutes (see [1] for a deeper comparison). This is sometimes called planning under time and resource constraints [12] to distinguish it from pure AI planning problems. The role of planning under time and resource constraints is increasing and because these planning problems are closely related to scheduling there exist tendencies to integrate both approaches [1,16,18] as a special track on such integration at the 6th European Conference on Planning (2001) shows. On the other hand, there is an attitude to keep resource scheduling separated from planning [19] supported by the idea of using up-to-date special algorithms to solve the problems separately and propagating the results using, e.g., a waterfall model. In [1], we argued for the methods based on the integration because we believe that such integration is more natural from the user

* Research supported by the Grant Agency of the Czech Republic under the contract number 201/01/0942.

point of view. Moreover, recent development of constraint satisfaction technology provides the tools that can be used to model and solve planning, scheduling, and timetabling problems [2,5,13,15,20].

In this paper, we describe a first attempt to model various aspects of planning, scheduling, and timetabling problems using a single unified model. This model could be something like PDDL [10] for the AI planning problems, but we do not specify a formal modelling language at this stage. Moreover, we do not cover whole AI planning; we restrict ourselves to planning under time and resource constraints (even if in some sense time and resources can be omitted from the model). More precisely, our model is intended to cover real-life scheduling and timetabling problems with some planning features. Our primary goal is to provide a generic model where the user can describe a real-life scheduling/timetabling problem. The basic notion in the model is an activity, activities are grouped in tasks and they are allocated to resources. At this stage we concentrate on expressiveness of the model rather than on solving a particular scheduling/timetabling problem. This model is indented to serve as a bridge between various solving techniques. At subsequent stages, we expect to provide rules that (semi-) automatically recommend a solving technique depending on the specification of the problem in this model.

The paper is organised as follows. We first define the notions of planning, scheduling, and timetabling as we use them in this paper and we sketch the main differences between these notions. Then we describe the motivation behind our research, i.e., providing a model integrating planning, scheduling, and timetabling. The main part of the paper contains a description of the model, namely specification of actions, resources, and tasks, as well as the description of objectives. We conclude with highlighting the advantages of our model and describing a future work.

2 Preliminaries

Before starting the specification of the model, let us first briefly define the notions of planning, scheduling, and timetabling. As we mentioned in the introduction, meaning of these terms could be slightly different depending on what community defines them. Moreover, the border between them is fuzzy so it may be even ambiguous to classify a particular problem. In this section we try to extract the main features that differentiate between traditional planning, scheduling, and timetabling. This helps us to classify our model in the traditional terminology.

Planning. Let us start with the definition of planning that is very close to general problem solving. The traditional AI 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. It means that a description of the initial world, the (partial) specification of the desired world and the list of available actions make the input of the planner. A solution is a sequence of actions that leads from the initial world description to the goal world description and it is called a plan. Note that the structure of the plan could be more general than a simple sequence, say a directed graph where nodes are marked by actions and the arcs describe time

precedence between the actions. In planning, neither the structure of the plan nor the actions in the plan are known in advance so the task is to generate the actions and to connect them into the structure forming the plan. This structure can not be arbitrary; the actions in the plan must follow some rules defined for example via pre-conditions and post-conditions of the actions (e.g., STRIPS rules). The notion of resource is not used explicitly in traditional planning. However, the use of resources could be encoded in pre-conditions and post-conditions of the actions (e.g. empty hand in the block world problem). Also, a relative time is used during planning only, e.g., an action A precedes an action B without particular specification when the given action must be performed. Recently, planning is becoming more aware of resources and time, so we are speaking about planning under time and resource constraints.

Scheduling. It may seem that planning under time and resource constraints is equivalent to the traditional scheduling but it is not true. 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. 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. The main difference between scheduling and planning under time/resource constraints is that in scheduling, we know the set of activities in advance while in planning we have to generate the activities. This difference also explains the major interaction between traditional planning and scheduling: first we plan/generate the set of activities and then we schedule/allocate these activities to resources. In [1], we proposed to integrate both planning and scheduling such that the activities can be introduced during scheduling.

Timetabling. Timetabling stays apart from the "fight" between planning and scheduling and it acts like a completely separate subject (even with a separate conference). However, when we look in detail at timetabling, we find it to be very close to scheduling. As pointed out by Wren [21] we can even say that timetabling is a special case of scheduling. Let us take a look on his definition of timetabling: it is the allocation, subject to constraints, of given resources to objects being placed in space-time, in such a way as to satisfy as nearly as possible a set of desirable objectives. In comparison with scheduling, we can see that the importance of the resource allocation is restrained. Also, the presented objective function emphasises different criteria than the cost optimisation over resources what is typical for scheduling. Important difference is that there are only rare direct relations between activities, most of the relations are expressed via resources (e.g., two activities can not be allocated to a single time slot). Thus timetabling uses different techniques to solve the problems, but similarities to scheduling are evident. In [14], we studied timetabling from a general point of view.

3 Motivation

Despite the differences among traditional definitions of planning, scheduling, and timetabling, relationship among these subjects is evident. Moreover, in practice it is quite frequent that a given problem is positioned at the border between planning and scheduling (since now we will include timetabling in scheduling, if no special feature of timetabling is required). It means that we have to solve both planning and scheduling sub-problem to get a final solution. Typically, this is done separately as described above but it could be done within a single system as we proposed in [1]. The approach separating planning and scheduling usually requires two more or less independent models, which makes the system less transparent from the user point of view ("I have a single problem why should I use two different applications to solve it?"). So first motivation behind our proposal is to use a single model to describe the problem. An integrated planner and scheduler can then use this model directly or the model can be converted to separate sub-models that are then solved using different technologies. Using a single model of the problem makes the modelling easier for the user, as he or she is shaded off from the implementation details. Note that we do not describe at this stage how to solve our generic model, this will be the subject of our future research. We expect to use constraint programming technology as the main solver that can integrate other techniques from traditional scheduling and planning. Some ideas of such solver can be found in [2,3].

Using a single generic model can also simplify solving a set of similar problems. When we are exposed to a particular problem, we can propose a special planner/scheduler to solve the problem. This is a current practice in companies like Cosytec (www.cosytec.com) that provide highly customised solutions, i.e., a dedicated system is designed for every customer. Another approach, used e.g. by Parc Technologies (www.parc-technologies.com), is to provide software covering a problem area rather than a single customer. The customer then describes/models the problem within the system, that is capable to solve all the problems in a given area. Our intention is to provide even more general model that allows description of planning/scheduling problems from different areas. At this stage, we would like to cover the core features of various planning/scheduling/timetabling areas. User-defined constraints can be added to tune the model to a particular problem in the next step.

There is another aspect of defining a generic model describing various problems - such unified model can help in the classification of the problems. There exists a known classification of scheduling problems using the triple *machine environment* | *job characteristics* | *optimality criterion* by Graham et al. (e.g., see [8]) but this classification is rather academic and it is hard to fit a real-life problem to one of these classes. Moreover note that this classification does not include a language to specify a particular problem. We would like to provide a framework where the user describes the real problem and using this description we could classify (more or less automatically) this problem or its parts to known categories. Subsequently, this classification can help in the selection of the suitable solving technology. Moreover, the unified model helps in the design of benchmarks when the language for description of the model will be provided. Such language, PDDL [10], is used in planning to describe the problem in a unified way. In the timetabling community, there is also effort to define a standard format for the problem specification, e.g., the

language called STTL [11]. Taking into account both standard approaches from the independent areas of planning and timetabling, we would like to get a new general specification for the combined planning and scheduling problems.

4 The Model

In the proposed model, we try to extract the common features of planning (under time and resource constraints), scheduling, and timetabling. We started from real-life problems of scheduling (integrated with planning) [1,2,3] and timetabling [14, 17] rather than from academic definitions. Currently, we would like to cover the core features of these problems; user-defined constraints can be added later. Our ambition is not to replace either PDDL or STTL as our problem area is integrated planning and scheduling (these languages cover the areas of AI planning and timetabling separately). Also, no formal language is provided there yet, we rather describe what features such model should cover and how.

There are three parts of the model: activities, tasks, and resources. The basic problem is to allocate activities to resources that are required by the activities. Moreover, the activities must be chosen such that the tasks are fulfilled. Tasks and resources describe the allowed structure of activities in the final plan/schedule, they also define timing of activities (which activities must precede other activities). Note that we allocate activities to particular time too; the notion of time is hidden in the definition of tasks and resources.

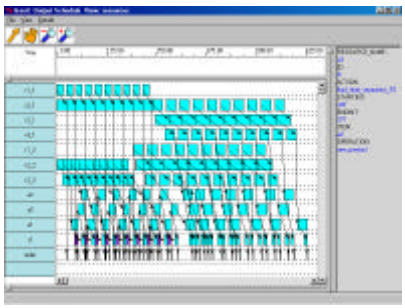

Fig. 1. Final schedule in the form of a resource-oriented Gantt chart. Rectangles represent the activities, the horizontal axis describes time and the vertical axis describes the resources. Arcs connect the activities belonging to a single task (they represent supplier-consumer relation).

4.1 Activities

Activity can be easily identified in planning, scheduling, and timetabling problems, even if the different notions are used to name this object there. Planning calls it action, traditional scheduling often considers operation, and timetabling works with meeting, course, etc. Activity in scheduling/timetabling is typically specified by its

duration. Actions in planning have usually assigned no duration if we are not working with time. Still some duration can be assigned to the action because it takes some time to perform it. When the duration is not relevant it can be easily set to zero. So, we have an activity as the basic scheduled/planned object and its duration as the first common parameter. Note also that the duration can be variable, e.g., it can be constrained by the resource that processes the activity.

It is possible that the activity can be processed at any time (perhaps depending on other activities, see description of tasks and resources) but in many problems **time windows** are defined to specify when the activity can actually be processed. In traditional scheduling, a release date and a deadline can be defined[1]. It means that the activity can not start before the release date and can not finish after the deadline. The release date and the deadline define a single time window for the activity, we can allow more time windows to be defined for the activity. For example, some activities can be processed only in specified time windows due to security or other reasons (e.g., noisy activities can be processed at day time only).

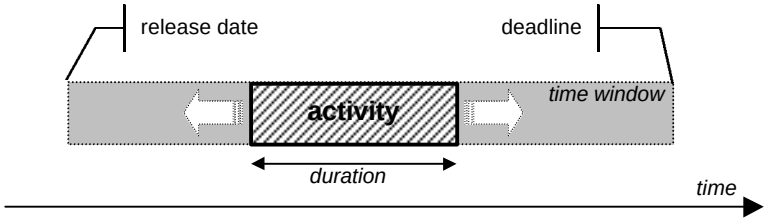

Fig. 2. Activity and its parameters at glance.

Parameters of the activity can distinguish between problems of different type. In Table 1, we show what are the typical activity parameters for planning, scheduling, and timetabling problems.

	Planning	**Scheduling**	**Timetabling**
Duration	no	yes	mostly fixed
Time windows	no	yes	sometimes

Table 1. Comparison of planning, scheduling, and timetabling using the activity parameters (by planning we mean planning without time and resource constraints).

4.2 Resources

Activity is typically allocated to a resource(s), so it consumes some capacity of the resource. We distinguish resources of two kinds: renewable (or reusable) and consumable.

[1] A release date and a deadline are defined for jobs (tasks) in traditional scheduling. However, we can define them for operations (activities) to increase the generality of the model.

A **renewable resource** is used by the activity, so when the activity is processed, it consumes the capacity of the resource, but as soon as the activity finishes, the capacity is returned back to the resource (see Figure 3 part a). Machines and classrooms are typical examples of the renewable resource.

A **consumable resource** vanishes when the activity is executed, i.e., capacity of the resource is not restored when the activity finishes (see Figure 3 part b). Fuel is a typical example of the consumable resource. Note that in addition to activities that consume a resource, we can also define activities that produce a resource, e.g., refuelling. In this case, available capacity of the resource is increased when the activity is finished (see Figure 3 part c).

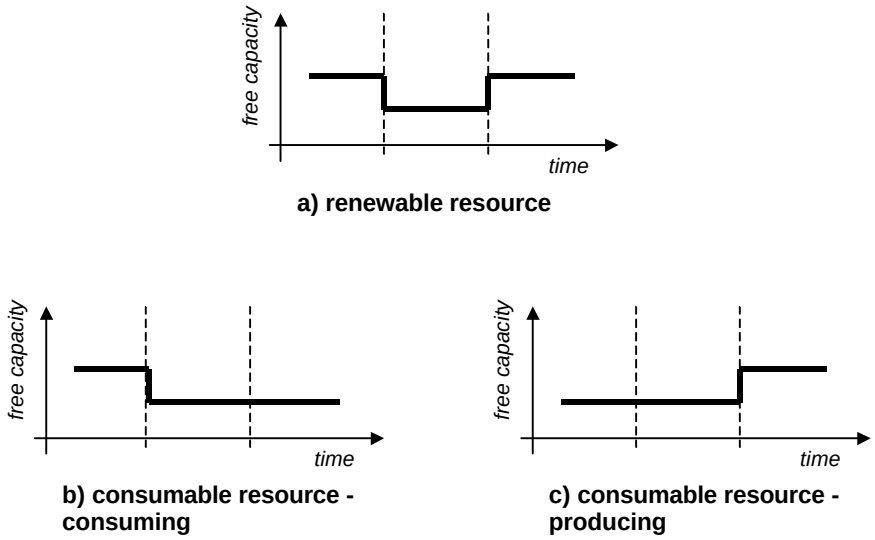

Fig. 3. Basic scenarios of consuming/producing resource capacity (dashed lines indicate the start and the end of the activity). Note that other scenarios can be designed as well, e.g., a continuous change.

To model the above scenarios of consuming/restoring a resource, we need two parameters of the resource: a capacity and an initial level. The **initial level** describes amount of the resource available at the schedule start. This parameter is useful for consumable resources, in case of renewable resources, the initial capacity equals to the capacity.

The **capacity** restricts the number of activities that can be processed at a renewable resource at the same time. In case of a consumable resource, this parameter may describe the size of a fuel tank, etc. Note that the capacity can be set to the supreme meaning no capacity restriction. On the other hand, the capacity can be set to 1 (we expect a discrete domain) meaning that only one activity can be processed at given time. Such resource is called unary and we are speaking about disjunctive scheduling. When the capacity is higher than 1, i.e., more activities can be processed together, then scheduling is called cumulative.

For each activity and each resource that can process the activity, we define an **activity capacity** consumed/produced by the activity. The activity capacity is simply an integer (we are working with a discrete domain), positive capacity means that the activity consumes the resource, negative capacity means that the activity produces the resource. So for each activity we have a group of couples *(Capacity, Resource)* describing the capacity requirement of the activity when processed by the *Resource*. We call this group an **alternative resource group**.

Until now, we expected that the activity is allocated to exactly one resource selected from the alternative resource group. This could be rather restrictive in some problems where more resources per activity are required. For example, a classroom, a teacher, and special equipment (like a projector) must be assigned to a course in timetabling [14, 17]. Thus, we can define **a set of alternative resource groups** for each activity [14]. Then, exactly one resource is selected from each group to process the activity. Note that the duration parameter and time windows are unique for the activity, i.e., even if several resources process the activity, duration of the activity is identical at each selected resource.

Resource capacity restricts the number of activities that can be processed at the same time. In the following example, we show that we need an additional mechanism restricting which activities can be processed together. Let us assume there is a bath that can galvanise metal items. The capacity of the bath is large enough to include several items but there is a restriction that only items of the same metal can be galvanised together. To solve this problem, we can introduce a **compatibility constraint**. There is a set of activities that can be processed by a given resource (this set can be derived from the alternative resource groups). We assign a type to each activity in this set; note that the activity may have different types in different resources. Now, the compatibility constraint says that only the activities with the identical type can be processed at the same time (in the above example, type is equivalent to metal).

Activities can be allocated to the resource arbitrary, we call it **parallel processing** (Figure 4 part a). However, sometimes a more restricted alignment of the activities is required, namely the activities that are processed together must start at the same time and they must end at the same time. Then we are speaking about **batch processing** (Figure 4 part b)[2].

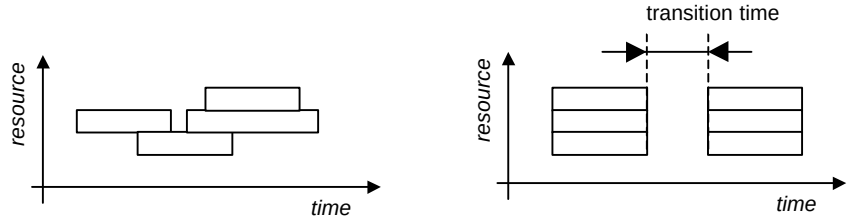

Fig. 4. Parallel vs. batch processing (rectangles describe activities).

[2] Our definition of batch processing corresponds to p-batching problems in traditional scheduling [7] with the restriction that we require all the activities in the batch to have identical duration. In s-batching problems, the batch is defined by a sequence of activities.

Compatibility constraint described in the previous paragraph plays an important role in batch processing. Type of all activities processed together in a single batch can be seen as a type of the batch. We can then define a special **transition time** between subsequent batches, sometimes called a set-up time. This transition time is dependent on type of both batches so a simple table can describe the transition times between each pair of types. If the transition is not allowed then the supreme is used as a transition time. Note also that the transition time from A to B can be different from the transition time from B to A, i.e., a complete table is necessary.

In Table 2, we show what is the difference between scheduling and timetabling when resource features are compared.

	Scheduling	**Timetabling**
Renewable resources	yes	yes
Consumable resources	yes	no
Capacity constraint	yes	often unary
Compatibility constraint	new	no
Batches and transitions	new definition	no

Table 2. Comparison of scheduling and timetabling using resources (AI planning is not included as it does not use resources; planning with time and resources has similar nature as scheduling).

4.3 Tasks

Activities in the problem are often dependent on each other, e.g., they are grouped in a task (in traditional scheduling we use the notion of job). Task describes the relations among the involved activities, e.g., the precedence relations. We propose to define a task as a directed AND-OR graph called the **task graph**: vertices are marked by activities and arcs describe precedence (and other) relations. AND branching means that a given activity must be preceded/followed by all the connected activities; OR branching means that one of the predecessors/successors must be selected, i.e., OR branches describe alternatives. A **precedence relation** defined by an arc says than the first activity must be finished before the second activity starts. Also, such relation can be more constrained, e.g., exact amount of time between the activities is specified (see Figure 5). To keep the task graph connected we introduce a special node called **root** that has no activity assigned (see Figure 5). The root node is connected to selected activities such that it is a part of all alternative solution sub-graphs. In the root we can keep some global information about the task, e.g., the maximal duration between the start of the earliest activity in the task and the end of the latest activity[3].

The OR branches introduce alternatives in the definition of the task. To fulfil (solve) the task we have to select a sub-graph of the task graph that includes the root node and represents one of the alternatives (in the standard meaning of AND-OR graphs).

[3] This, sometimes called a validity time, is useful to model problems in food industry etc.

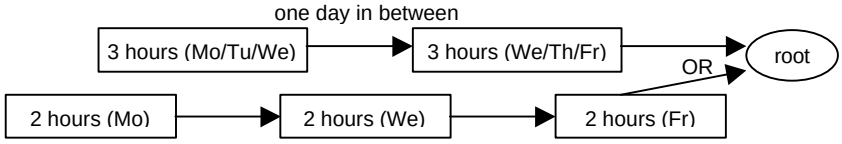

Fig. 5. Task in timetabling may express alternative composition of lectures in the course, e.g., course with 6 hours per week can be taught as two 3 hours lectures taught on Monday+Wednesday, Tuesday+Thursday, or Wednesday+Friday or three 2 hours lectures taught on Monday, Wednesday, and Friday.

The structure of the task graphs can identify to which category a particular problem belongs. In timetabling, the task graph is typically represented by a single activity (course/meeting) or a short sequence of activities. In Figure 5, we show a more complex structure of the task graph that contains two alternatives. When this case appears in practice, one of the alternatives is usually selected before we start solving the problem. Here we present a more general concept where an alternative can be selected during the course of problem solving.

In traditional scheduling, the task graph usually consists of a sequence of actions (operations). More precisely, in job-shop and flow-shop scheduling, every task consists of a sequence of activities (see Figure 6, part a). In open-shop scheduling, there are no precedence relations among the activities and so a task graph can be expressed as a root node connected to the activities (see Figure 6, part b). Note that this connection is an AND-branching – in traditional scheduling (as well as in traditional timetabling) there are no OR-branches, i.e. no alternatives in the structure of the task. Typically the alternatives are resolved during planning, so traditional scheduling/timetabling is free of alternatives in the task definition[4].

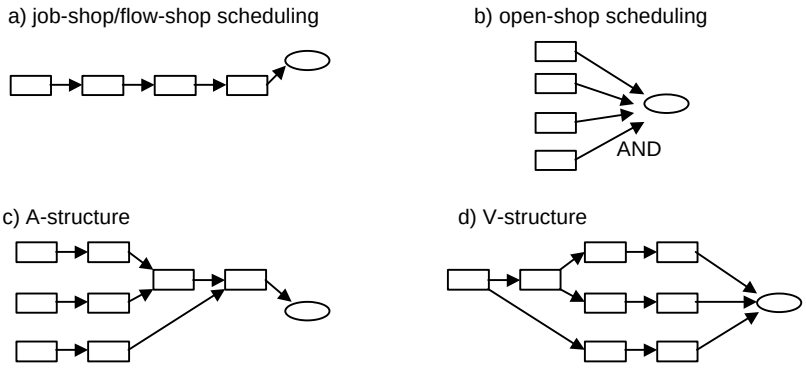

Fig. 6. Basic structures of the task in traditional scheduling.

[4] There are alternatives in traditional scheduling/timetabling, for example alternative resources to process a given activity. It is different from the structure of alternative activities in the task, though.

In addition to the basic structures of the task, i.e., a sequence of activities, and a set of activities with no precedence relations, there exist two more basic structures of the task derived from the real-life production: A-structure (Figure 7, part c) and V-structure (Figure 7, part d). Task with the **A-structure** describes a production of a single product from several parts (say construction of a car). Task with the **V-structure** describes the processing of a raw material to several products (e.g., in petroleum refinery). Note that no alternatives appear in all above structures. In traditional scheduling, A-structure and V-structure is defined over the tasks (jobs) rather than over the activities (operations) and even more complex structures consisting of parallel and serial compositions are supported there [8]. Still all these structures contain neither alternatives (OR-branches) nor cycles. Because our model supports both these features, we believe it to be more general than traditional scheduling models. In Figure 7 we show a more realistic task graph motivated by real production scheduling that includes both alternative production sequences and some form of cycling.

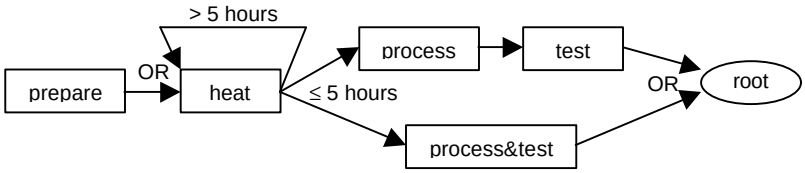

Fig. 7. Task in production scheduling describes the production process. Notice the two alternative branches in production and a cycle to re-heat the product.

In planning, the task graph is usually not given explicitly but it is described via e.g. STRIPS rules. This is because the whole task graph for planning problems could be huge, so the planning problem is to generate and search a narrow part of the task graph and to find a solution sub-graph there. In the Graphplan algorithm [6], a planning graph similar to our task graph is defined. In the planning graph, the activities interleave with propositions and the resulting plan consists of a sequence of activities while our task graph supports a more complex structure of the plan. Still we can say that in some sense our definition of the task graph corresponds to a solution of the planning problem that contains alternatives.

There exist scheduling algorithms that can tackle the problems containing alternatives in the task graph. In [4], an algorithm for scheduling alternative activities was proposed. This algorithm can be used if the task graph is given explicitly, there are not too many alternatives, and there are no cycles. If the number of alternatives is large in comparison to the final plan then this approach is less efficient. Moreover, if there are cycles in the graph then a more dynamic solving approach is required. In [3], we proposed such an approach based on a slot representation for solving problems described using a resource-oriented model.

4.4 Objectives

The proposed model consists of activities that are allocated to resources and that are grouped in tasks. A group of activities is selected from each task graph and these activities are allocated to resources. Resource constraints (capacity, etc.) as well as task constraints (precedence etc.) must be satisfied. We call this problem **a feasibility problem** because the goal is to find a feasible plan/schedule/timetable. We should also mention that we expect that the tasks are independent at this stage, i.e., there is no direct relation among them. The constraints among activities of different tasks are derived from sharing the same resource only. General relations among the activities of different tasks can be modelled via their integration into a joint task.

In many cases, the problem is not to find any schedule but a good schedule. This type of problem is usually called an **optimisation problem**. In traditional scheduling, there exist several objective functions to describe a quality of the schedule, e.g., minimising makespan, minimising tardiness, etc. (see [8] for details). Timetabling typically differs from scheduling in the objective function. For example we are expected to maximise the number of fulfilled tasks or to maximise satisfaction of soft constraints [17]. The proposed model is ready to support various objective functions; the description of them is out of scope of this paper however.

5 Conclusions and Future Works

This paper describes neither a new planning/scheduling/timetabling algorithm nor a particular problem from these areas. We rather concentrate on extracting key common features of planning, scheduling, and timetabling and on encapsulating these features into a unified model. The main contribution of this paper is presenting different concepts in a single framework. We generalised some modelling ideas behind resources and we believe that our concept of task as an AND-OR graph is new to scheduling and timetabling.

Our intention is not to propose another academic model but our modelling approach is motivated by real-life problems. Still we want to describe these problems formally, which gives us a tool for further studies of the real-life problems. Thus, our subsequent research will cover fitting the proposed model into existing frameworks for planning, scheduling, and timetabling. Our ambition is to design a formal descriptive language for specifying various real-life problems at the border of planning and scheduling. We also plan to show how other real-life problems can be modelled using our framework. The ultimate goal is to propose a mechanism for solving the problems defined in our model, e.g., by partitioning them into sub-problems that can be solved using existing techniques.

6 References

[1] Barták R.: On the Boundary of Planning and Scheduling: A Study. In Proceedings of the 18th Workshop of the UK Planning and Scheduling SIG, Manchester, pp. 28-39, 1999.

[2] Barták R.: Dynamic Constraint Models for Planning and Scheduling Problems. In New Trends in Constraints, LNAI 1865, Springer Verlag, pp. 237-255, 2000.

[3] Barták R.: A Slot Representation of the Resource-Centric Models for Scheduling Problems. In Proceedings Dynamic Constraint Models for Planning and Scheduling Problems. In Proceedings of the ERCIM Workshop on Constraints, Padova, 2000.

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

[5] Binh Do M. and Kambhampati S.: Solving planning-graph by compiling it into CSP. In Proceedings of AIPS 2000, pp. 89-91, 2000.

[6] Blum A. and Furst M.: Fast Planning Through Planning Graph Analysis. In Artificial Intelligence 90, pp. 281-300, 1997.

[7] Brenner M.: A Formal Model for Planning with Time and Resources in Concurrent Domains. In Proceedings of IJCAI-01 Workshop Planning with Resources, Seattle, 2001.

[8] Brucker P.: Scheduling Algorithms. Springer Verlag, 2001.

[9] Erol K., Hendler J., and Nau D.: UMCP: A Sound and Complete Procedure for Hierarchical Task-Network Planning. In Proceedings of 2nd International Conference on AI Planning Systems, pp. 249-254, 1994.

[10] Ghallab M., Howe A., Knoblock C., McDermott D., Ram A., Veloso M., Weld D., Wilkins D.: PDDL - The Planning Domain Definition Language, Tech Report CVC TR-98-003/DCS TR-1165, Yale Center for Computational Vision and Control, 1998.

[11] Kingston J.H.: Modelling Timetabling Problems with STTL. In Proceedings of The Practice and Theory of Automated Timetabling, LNCS 2079, Springer Verlag, pp. 309-321, 2001.

[12] Koehler J.: Planning under Resource Constraints. In Proceedings of 13th European Conference on Artificial Intelligence, Brighton, pp. 489-493, 1998.

[13] Laborie P.: Algorithms for Propagating Resource Constraints in AI Planning and Scheduling: Existing Approaches and New Results. In Proceedings of 6th European Conference on Planning, Toledo, 2001.

[14] Müller T., Barták R.: Interactive Timetabling. In Proceedings of the ERCIM Workshop on Constraints, Prague, 2001.

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

[16] Nareyek A.: AI Planning in a Constraint Programming Framework. In Proceedings of 3rd International Workshop on Communication-Based Systems, 2000.

[17] Rudová H.: Soft Scheduling. In Proceedings of the ERCIM Workshop on Constraints, Prague, 2001.

[18] Smith D.E, Frank J., and Jónsson A.K.: Bridging the Gap Between Planning and Scheduling. In Knowledge Engineering Review, 15(1), pp. 61-94, 2000.

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

[20] Van Beek P. and Chen, X.: CPlan: A Constraint Programming Approach to Planning. In Proceedings of AAAI-99, pp. 585-590, 1999.

[21] Wren A.: Scheduling, Timetabling and Rostering - A Special Relationship. In Proceedings of The Practice and Theory of Automated Timetabling, LNCS 1153, Springer Verlag, pp. 46-76, 1996.