Investigating the Integration of Planning and Scheduling: A First Step

M[a] Dolores Rodrguez-Moreno[1], Angelo Oddi[2], Daniel Borrajo[3], Amedeo Cesta[2], and Daniel Meziat[1]

[1] Departamento de Automática. Universidad de Alcalá.
Carretera Madrid-Barcelona, Km. 33,600. 28871 Alcalá de Henares (Madrid), Spain.
{mdolores, meziat}@aut.uah.es
[2] ISTC-CNR-Italian National Research Council
Viale Marx 15, I-00137 Rome, Italy.
{oddi, cesta}@ip.rm.cnr.it
[3] Departamento de Informática. Universidad Carlos III de Madrid.
Avda. de la Universidad, 30. 28911 Leganés (Madrid), Spain.
dborrajo@ia.uc3m.es

Abstract. Integrating Planning and Scheduling is becoming an increasing topic in Artificial Intelligence due to real application needs. Some problems allow a strict separation between planning and scheduling. The way to solve the problem is to assign time and resources to the planned activities. But in other cases there is an indirect temporal dependency with other states that can not be taken into account if we follow the first approach. Techniques from both fields must be merged.
In this paper, we present several approaches to integrate planning and scheduling that have emerged from the developpement of a tool that sets up nominal operations in the three satellites of the HISPASAT company.

1 Introduction

Planning and scheduling have always been two very related tasks. Planning generates a plan (sequence or parallelization of activities) such that it achieves a set of goals and satisfies a set of domain constraints. Scheduling is an optimisation task where limited resources are allocated over time among both parallel and sequential activities such that makespan and/or resources usage are minimised.

Both fields have strong and weak points. From one hand we have the planning systems with a rich representation of the problem description (usually following the so called STRIPS representation). However, this type of representation has problems for dealing with variables with infinite values such as information about time and resources. Usually, they use a discrete model of time in which all actions are assumed to be instantaneous and uninterruptible. Also, there is not an explicit language that allows to represent the basic Allen primitive relations between temporal intervals [1]. With respect to resources there is not a way to handle cumulative resources. On the other hand, scheduling systems can perfectly handle temporal reasoning and resource consumption, but they cannot

produce the needed precedence relations among activities given that they lack an expressive language to represent the activities.

From this perspective, by combining scheduling and planning systems synergistically these weaknesses can be solved [11]. The planning systems could deal with time and resources thanks to the scheduling search procedure that interleaves refinement solution (assigning values to variables) and constraint propagation (computation of implications of these assignments to others variables and elimination of inconsistent values). On the other direction, the planning system can supply to the scheduler the language and the precedence constraints for the activities.

As a first step on merging both types of tasks, we have explored the possibility of using a standard non-linear domain independent planner, PRODIGY [20], and using it directly for integrating planning and scheduling. This planner does not have an explicit model of time representation nor a declarative way for specifying resource requirements or consumption. However, thanks to its capability of representing and handling continuous values variables, coding functions to obtain variables values and the use of control rules to prune the search, we have successfully integrated them in a satellite control domain [17]. This domain needs to integrate planning and scheduling for setting up nominal operations to perform in three satellites during the year. Similar approaches that allow planners to also reason about temporal and resource usage are IxTeT [12] or HSTS [15].

A second approach we have explored consists on using the O-OSCAR scheduler system [5,8] in combination with the QPRODIGY planner [20,3].[4] For a given problem, PRODIGY generates a plan as a sequence of activities to be executed. Then, the O-OSCAR scheduler obtains a viable temporal and resource solution. A similar approach can be found in [19], though they did not use a planner that is able to handle by itself some types of temporal information. Therefore their planner is not able to avoid generating some types of temporal invalid plans.

As a future step we propose to integrate the temporal and resource reasoning of the scheduler in a parallel execution with the planner, by exchanging relevant information. This research would follow a similar approach to that reported in [11]. The main difference relies on the planner. Since ours allows to use some type of temporal/resource usage reasoning together with some optimisation capabilities, we can control/vary the amount of reasoning that the planner will perform.

The paper is structured as follows. Section 2 presents how the planner handles the problem. Next, the integration of the planner and the scheduler is presented. Finally conclusions are drawn and future work is outlined.

2 Handling time and resources using a planner

PRODIGY [20] is an integrated architecture that has been used in a wide variety of domains: from artificial domains to real applications such as satellites control. The problem solver is a non-linear planner that uses a backward chaining

[4] QPRODIGY is a planner based on PRODIGY that is able to handle optimisation metrics.

means-ends analysis search procedure with full subgoal interleaving. The planning process starts from the goals and adds operators to the plan until all the goals are satisfied. The inputs of the planner are:

1. The domain theory that contains all the actions represented by the operators. To define them, one has first to define the variables that are used in the operator. For that, one needs to declare the type of each variable, that can be either finite or infinite. Finite types can be defined structured in a hierarchy as in PDDL2.1 [10], although the PRODIGY syntax also allows boolean operations on types. For infinite types PRODIGY allows coded functions to generate a list of values to be possible bindings for the corresponding variable by using the information of the current state of the search. In Figure 1, these functions are represented separatedly from the domain description. This has been done for two purposes: to emphasize the fact that this type of knowledge is important for our research purposes (as we will describe in next sections); and for uniformity with the rest of literature in planning with respect to the meaning of domain description (usually a set of operators plus a hierarchy of types).
   As an example, in the satellite domain, all data is represented in seconds since 1900 in GMT, se there is an infinite type TIME that represents this type of values and is used in the operators. The reason to use this format is for efficiency: it is faster for the planner to generate the bindings of one number variable instead of generating values for the six usual time dependent variables (corresponding to the year, month, day, hours, minutes and seconds). Also GMT is the reference zone time for the satellite company HISPASAT.
2. The second input to the planner is the problem, described in terms of an initial state and a set of goals to be achieved. In the satellite domain, the initial state describes all the events that will occur during the year such as moon blindings, sun blindings, eclipses, etc. With respect to the goals, the planner has to obtain a plan to perform all the maintenance operations along the year.
3. When there is more than one decision to be made at decision points, the third input to the planner, the control knowledge (declaratively expressed as control rules) could guide the problem solver to the correct branch of the search tree avoiding backtracking. There are three types of rules: selection, preference or rejection. One can use them to choose an operator, a binding, a goal, or deciding whether to apply an operator or continue sub-goaling.

As a result, a Total Order Plan is generated. Figure 1 shows the inputs and output of PRODIGY. In the case of our study in the satellite domain, the plan contains details in each operator instance about time and resource assignment. The time representation of PRODIGY is a discrete model of time, in which all actions are assumed to be instantaneous and uninterruptible. There is no provision for allowing simultaneous actions. However, the coded functions that can be used in the preconditions of operators allow to add constraints among and within

operators and therefore the seven Allen primitive relations between temporal intervals can be handled.

Fig. 1. PRODIGY inputs and output.

As an example, the *A meets B* primitive is shown in Figure 2. The function `add-time` calculates the end time of the operation. `DUR` is a number, integer or not, that represents the duration, and `TIME` defines the time units used (seconds, minutes, hours, days or months). `gen-from-pred` is a PRODIGY function that allows to bind the corresponding variable with all possible values that make the function's argument (a literal) match a corresponding literal in the state. Therefore, `(<dstA> (gen-from-pred (starts-at <dstA>)))` will match the literal `(starts-at <dstA>)` with the current state literals in order to assign to variable `<dstA>` all the matching values.

```
A
variables: (<dstA> (gen-from-pred (starts-at <dstA>)))
           (<dendA> (add-time <dstA> DUR_A TIME_A))
effects:   (finished-A <dendA>)

B
variables: (<dstB> (gen-from-pred (finished-A <dendA>)))
           (<dendB> (add-time <dstB> DURB_B TIME_B))
effects:   (started-B <dstB>)
           (finished-B <dendB>)
```

Fig. 2. The *A meets B* Allen primitive.

When operator `A` is applied, it adds to the state the fact that it has finished at some point in time (`<dstA>`+$\texttt{DUR}_A$ in $\texttt{TIME}_A$ time units). Operator `B` binds to its variable `<dstB>` that time point, so it knows when it should start its execution.

In PRODIGY there is also no provision for specifying resource requirements or consumption. Resources can be seen as variables that can have associated values through literals that refer to them. One solution consists on restricting in the operators the set of values that can be assigned to the variable that represents the resource.

To represent capacity, we can use the scalar quantity model. The capacity constraints of a resource with uniform capacity can be calculated using coded functions as in time. To know more about the time and resource model we have used, we refer to [17].

3 Using a planner and a scheduler

The first approach that we have described in the previous section to integrate planning and scheduling within the same tool presents some problems when dealing with time and resources such as low reuse of the coded functions, that is, it is domain dependent. In this section we present two approaches that explicitly use a scheduler, O-OSCAR [5,8], in combination with the planner to perform the same integration. We first describe at a high level the scheduler, and then describe the integration approaches.

3.1 The scheduler: O-OSCAR

O-OSCAR is a scheduler that follows a Constraint Satisfaction approach [14] to solve complex multi-capacity temporal problems. The release used in this paper is able to solve project scheduling problems with time windows by using the ISES algorithm [7]. Its basic solving method uses a two layered problem representation:

1. The ground layer, or ground-CSP, that only represents the temporal aspects of the problem in the form of a quantitative temporal constraints network [9];
2. The higher layer is a meta-CSP, where resource conflicts are represented and reasoned about. In this layer resource constraints are super-imposed, giving rise to a set of resource conflicts that are solved with a number of sequencing decisions [6,7].

The search proceeds by iteratively performing the following steps:

- Propagation of temporal consequences through the ground-CSP to compute currents bounds for all temporal variables.
- Computation of the meta-CSP identifying the set of resource capacity violations implied by the temporal network.
- Selection of a conflict in the meta-CSP according to variable ordering heuristic.
- Resolution of the conflict by imposing a new precedence constraint in the ground-CSP. This is done using a value ordering heuristic that preserves temporal flexibility.

The result is an efficient greedy procedure for generating feasible solutions to the Resource Constrained Project Scheduling Problem with time windows (RCPSP_{max}). This basic one-pass procedure is then inserted in a random restart optimization cycle that incrementally searches for better solutions by sampling new solutions on problems with reduced temporal horizons.

3.2 PRODIGY and O-OSCAR

A second approach to solve the complete planning and scheduling problem consists of integrating in sequence the planner PRODIGY and the scheduler O-OSCAR. Figure 3 shows the inputs and outputs of this approach. In this case, we remove from the domain all time-related information, letting O-OSCAR take advantage of handling the temporal information. PRODIGY generates a sequence of atemporal operations that describes the precedence relations among activities. Then, the plan is given to O-OSCAR that assigns a start and end time for each operator. A parser must be built to translate the atemporal plan into the O-OSCAR inputs.

Fig. 3. Sequential approach of PRODIGY and O-OSCAR.

As an example, the main inefficient operations for solving problems in the satellite domain with PRODIGY alone were checking the moon blindings within manoeuvre operations. Using the second approach, we can now handle it by eliminating the function that calculates the blindings. Then, we model them as a binary resource, i.e. sun availability, with two values: 0 if it is available and 1 when it is not available due to any type of blindings. As a result, O-OSCAR tries to locate the manoeuvre operations where the sun resource is available.

The disadvantages of this approach are:

- Not every precedence ordering between plan steps in a Total Order plan is necessary for maintaining its consistency. In order to solve this, a set of TO plans can be compressed into a structure known as a *Partially Ordered set of steps*, along with a set of constraints on these steps [18,21]. Another way of avoiding this, would be directly using a partial-order planner, such as UCPOP [16]. However, we wanted to maintain the rich representation language of PRODIGY (that allows infinite types, for instance) together with

its flexibility to easily inspect the search trees, define control knowledge, or define new alternative control strategies (through the use of functions called handlers) [4].
- The lack of communication between the two systems: in case of a failure in the scheduler, a new solution has to be generated from scratch so the computational cost is high.
- There is no way to represent explicitly in the Prodigy language, or in PDDL2.1, resources, temporal constraints between activities or define a makespan in the solution.

3.3 PRODIGY, a TO-PO algorithm, and O-OSCAR

To overcome the first drawback of the last approach, we convert the Total Order output given by PRODIGY into a Partial Order plan. A PO plan represents a set of TO plans, where each TO is a linear sequence of steps in the PO such that the PO relation in the latter is not violated by the sequence. We have followed the algorithm in [21] that takes advantage of the given total ordering of the plan, by visiting at each step, only earlier plan steps. The algorithm is sketched in Figure 4.

Procedure **Build Partial Order(TO-Plan)**
Input:
TO-Plan: A totally ordered plan (being n the number of steps in the plan) and the start operator with preconditions set to the initial state.
Output: A partially ordered plan shown as a directed graph

1. **for** i=n down-to 1 **do**
   (a) **for** each *precond*∈Preconditions(op_i) **do**
   Find an operator op_j in plan that has as an effect *precond*
   Add directed edge from op_j to op_i)
   (b) **for** each *del*∈Delete-Effects(op_i) **do**
   Find all operators that have as a precondition (some of the, a) delete effects of op_i
   Add directed edge from these operators to op_i
   (c) **for** each add∈Primary-Adds(op_i) (if it appears either in the goal or in the subgoaling chain of a goal proposition) **do**
   Find all operators that delete any of the primary adds of op_i
   Add directed edge from these operators to op_i
2. Remove Transitive Edges in the graph
   Every directed edge e connecting op_i and op_j is removed if there is another path that connects the two vertices

Fig. 4. Algorithm that transforms a TO into a PO.

This greedy algorithm constructs an entirely new PO, analysing the action conditions, and using the original TO to guide the greedy strategy. It may fail to

produce a minimally constrained deordering as explained in [2] because of step 1(a) in the algorithm. From all possible operators in the plan that achieve an effect needed as precondition of another operator op_j later in the plan, it always selects the closest one before op_j. Since, there may be other operators earlier in the plan having the same effect and being a better choice, the algorithm is not optimal. However, given that PRODIGY (as most current planners) is not an optimal planner either (unless all alternatives are searched for, which can be actually done in PRODIGY), then the non-optimality of the TO-PO algorithm is not a crucial point.

Fig. 5. Sequential approach of a PO solution of PRODIGY and O-OSCAR.

This approach is shown in Figure 5. As it is also the case of the last one, it shares the high computational cost and the lack of an explicit representation for time and resources, but there is a little bit more communication between the planner and the scheduler. The algorithm can be extended to look for different deorderings depending on the information that the scheduler gives back. This means there are two meta level backtracking points: another deordering from the TO-PO algorithm, or another plan from the planner in case previous alternatives fail.

4 Conclusions and future work

In this paper we have presented an initial step to integrate planning and scheduling and some thoughts on possible further developments. First we have used a classical planner to reason about time and resources for a satellite domain. Then, the temporal and resource information has been eliminated from the domain, and the output generated by the planner is used as an input for a scheduler. The current step is quite preliminary and mainly aimed at putting together different architectures and to understand the problems involved in their integration. The experimental results mainly show that the approach works but further work is needed to gain effectiveness.

One aspect that has not been addressed yet concerns the feedback from the scheduler to the planner to be used to influence the planner choices. Along this

line we are currently pursuing a thighter integration schema to interleave the refinement solution inside the search tree of PRODIGY. In particular we are using QPRODIGY [3] in the aim of taking advantage of the quality metrics that this version is able to use. The sketchy idea is to have two systems evolving in parallel; the one of PRODIGY and a second with the time and resource constraint representation contained in the scheduler. Information is continuously exchanged between these two representations and in particular is used to influence the PRODIGY search to prune choices that lead to either temporal or resource inconsistency.

Further directions that are currently pursued concern the knowledge engineering aspects. From the experience gained in the HISPASAT project and studying the weak points of the integration, we are developing a graphical domain construction and validation tool for PDDL2.1 language as it has being done in the GIPO environment [13]. The language has been extended having in mind resources definition, temporal constraints between activities and the possibility to minimise/maximise the makespan or resources usage.

Acknowledgements

We want to thank all the HISPASAT Engineering Team for their help and collaboration shown during the developing time of the HISPASAT project, especially Arseliano Vega, Pedro Luis Molinero and Jose Luis Novillo. The first author also wants to thank the ISTC-CNR group for their help during the visit to CNR. This work was partially funded by the CICYT project TAP1999-0535-C02-02, the Spanish MCyT research network TIC2001-4936-E and a bilateral coordinated project funded by Spanish and Italian Foreign Affairs Departments. Cesta and Oddi work is partially supported by ASI (Italian Space Agency) project ARISCOM.

References

1. Allen, J. Towards a general theory of action and time. *Artificial Intelligence 23* (1984), 123–154.
2. Bäckström, C. Computational Aspects of Reordering Plans. In *Journal of Artificial Intelligence Research* (1998), vol. 9, Morgan Kaufmann Pub. Inc., pp. 99–137.
3. Borrajo, D., Vegas, S., and Veloso, M. Quality-based learning for planning. In *Working notes of the IJCAI'01 Workshop on Planning with Resources. IJCAI Press. Seattle, WA (USA).* (2001).
4. Carbonell, J. G., Blythe, J., Etzioni, O., Gil, Y., Joseph, R., Kahn, D., Knoblock, C., Minton, S., Pérez, A., Reilly, S., Veloso, M., and Wang, X. PRODIGY4.0: The manual and tutorial. In *Tech. Rep. CMU-CS-92-150, Department of Computer Science, Carnegie Mellon University* (1992).
5. Cesta, A., Cortellessa, G., Oddi, A., Policella, N., Severoni, F., and Susi, A. Reconfigurable constraint-base architecture as a support for automating mission planning. In *ESA Workshop on On-Board Autonomy. (AG Noordwijk, The Netherlands).* (2001), pp. 219–226.

6. Cesta, A., Oddi, A., and Smith, S. Profile Based Algorithms to Solve Multiple Capacitated Metric Scheduling Problems. In *Proceedings of the Fourth Int. Conf. on Artificial Intelligence planning Systems (AIPS-98)* (1998).
7. Cesta, A., Oddi, A., and Smith, S. A Constrained-Based Method for Project Scheduling with Time Windows. *Journal of Heuristics 8* (2002), 109–136.
8. Cesta, A., Oddi, A., and Susi, A. O-Oscar: A flexible object-oriented architecture for schedule management in space applications. In *Proc. of the Fifth International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS-99)* (1999).
9. Dechter, R., Meiri, I., and Pearl, J. Temporal Constraint Networks. *Artificial Intelligence 49* (1991), 61–95.
10. Fox, M., and Derek, L. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *In AIPS'02 competition* (2002).
11. Garrido, A., and Barber, F. Integrating Planning and Scheduling. In *Applied Atificial Intelligence* (2001).
12. Ghallab, M., and Laruelle, H. Representation and Control in IxTeT, a Temporal Planner. In *Proceedings of the Second International Conference on AI Planning Systems (AIPS-94)* (1994).
13. McCluskey, T. L., Richardson, N. E., and Simpson, R. M. An Interactive Method for Inducing Operator Descriptions. In *Proc. of AIPS'02 Workshop on Planning for Temporal Domains* (2002), pp. 151–159.
14. Montanari, U. Networks of Constraints: Fundamental Properties and Applications to Picture Processing. *Information Sciences 7* (1974), 95–132.
15. Muscettola, N. HSTS: Integrating Planning and Scheduling. In *Intelligent Scheduling*, M. Zweben and M. Fox, Eds. Morgan Kaufmann, 1994.
16. Penberthy, J. S., and Weld, D. S. UCPOP: A sound, complete, partial order planner for adl. In *In Proceedings of KR-92.* (1992), pp. 103–114.
17. R-Moreno, M. D., Borrajo, D., and Meziat, D. An artificial intelligence planner for satellites operations. In *ESA Workshop on On-Board Autonomy. (AG Noordwijk, The Netherlands).* (2001), pp. 233–240.
18. Regnier, P., and Fade, B. Complete determination of parallel actions and temporal optimization in linear plans of action. In *European Workshop on Planning* (1991), pp. 110–111.
19. Srivastava, B., and Kambhampati, R. Scaling up planning by teasing out resource scheduling. In *In Proceedings of the Fifth European Conference on Planning (Durham, United Kingdom), S. Biundo and M. Fox, Eds.* (1999).
20. Veloso, M., Carbonell, J., Prez, A., Borrajo, D., Fink, E., and Blythe, J. Integrating planning and learning: The prodigy architecture. In *Journal of Experimental and Theoretical AI* (1995), vol. 7, pp. 81–120.
21. Veloso, M., Pérez, A., and Carbonell, J. Nonlinear Planning with Parallel Resource Allocation. In *Proceedings of the DARPA Workshop on Innovative Approaches to Planning, Scheduling, and Control* (1990), Morgan Kaufmann Pub. Inc., pp. 207–212.