An Artificial Intelligence Planner for Satellite Operations

MD. R-Moreno [(1)], D. Borrajo[(2)], D. Meziat1 [(1)]

[(1)]Departamento de Automática. Universidad de Alcalá (Madrid). Spain
Carretera Madrid-Barcelona, Km. 33,600
28871 Alcalá de Henares (Madrid) Spain.
Email: {mdolores,meziat}@aut.uah.es

[(2)] Departamento de Informática. Universidad Carlos III de Madrid
Avda. de la Universidad, 30
28911 Leganés (Madrid). Spain.
Email: dborrjo@ia.uc3m.es

INTRODUCTION

Planning and scheduling are closely related areas with which the AI community has been long concerned. The first one deals with finding plans to achieve some goals from an initial state; that is, a sequence of activities that will modify the initial state into another that satisfies the goals. The second one refers to the allocation of available resources to known activities over time in order to produce schedules that respect temporal relations and resource capacity constraints. They can also optimise a set of objectives, such as minimising tardiness, minimising work in process, maximising resource allocation or minimising cycle time. Due to these differences, planning and scheduling have been traditionally solved separately using rather different methods.

Depending on the problem complexity some domains allow a strict separation between planning and scheduling. For example, in a travelling domain, one can first plan the actions to follow in the train trips to later assign the trip start and end points and the departure and arrival times at each station according to the desired timetable. 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. In the same example, it could happen that two trains need to use the same track. The schedule must allocate time intervals for the use of the track to avoid the trains colliding.

One way of integrating both tasks within the same tool consists on adding representation and reasoning capabilities on resource and temporal information. This has been done in systems such as IxTeT [7], O-Plan2 [11] or HSTS [8]. In our case, we wanted to explore the possibility of using a standard non-linear domain independent planner, PRODIGY [12], and study the possibility of using it directly for integrating planning and scheduling. This planner does not have 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 variables, coding functions to obtain variables values and the use of control rules to prune the search, we have successfully integrated planning and scheduling in a satellite control domain. This domain needs to integrate planning and scheduling for setting up nominal operations to perform in three satellites during the year.

The paper is structured as follows: section two introduces the satellites type operations that have to be performed during the year. Section three presents how the planner handles the problem. Then, a formal description of the type of knowledge represented in the planner is described. Finally related work is outlined and future work and conclusions are drawn.

SATELLITE OPERATIONS

This section gives a brief overview of the HISPASAT ground scheduling operations for its three satellites. HISPASAT is a Spanish multi-mission system in charge of satisfying national communication needs. It also supplies capacity for digital TV in Europe, America and North Africa, TV image, radio and other signals, and special communications for defence purposes. It is the first European satellites system to offer transatlantic capacity that allows simultaneous cover between South and North America.

Every operation in orbit must have explicit engineering instructions. These instructions provide a guide for technicians to consider the work accomplished. The operations engineering group generates this documentation every year *by hand* and o*n paper*. Later, this documentation is revised and verified. Due to the increasing number of satellites, actually three: 1A, 1B and 1C and shortly the 1D satellite, a program that handles and validates the operations for engineering support is needed.

Scheduling the nominal operations for on orbit control can be viewed as a constraint satisfaction problem where each satellite has its own restrictions depending on the different events that can occur over time. We could identify three categories of planned operations, according to the flexibility to schedule them (hard vs. soft constraints):

- Operations which are driven by external events and shall start or be completed at a fixed time such as Moon Blindings, Sun Blindings or Eclipses of the Sun by the Earth or by the Moon. These hard constraints are represented as pre-conditions of the operators.
- Operations that are part of a given strategy and should be performed at specific dates. However, some flexibility exists to modify the strategy. This is the case of Manoeuvres, Localisation Campaigns or Batteries Reconditioning. Although they are not hard constraints, we have coded functions in the pre-conditions to guide the planner for preferring some dates over others having in mind the restrictions imposed in some pre-conditions.
- Operations that should be performed within a wide period of time, and a significant flexibility exists for scheduling them as in Steerable Antenna Maintenance. These are the softest constraints and they are represented as control rules for preferring some dates over others.

Now we will describe what these three types of operations imply on defining planning knowledge in terms of operators, initial state and goals.

In the type of operations that belong to the first category, the external (i.e. eclipses and blindings) and seasonal (i.e. solstice and equinox) events are foreseen several weeks before the year starts. All these events are represented in the initial conditions of the problem. In the case of external events, we need to include the affected satellite, the start and end times, and for the seasonal events just the start time. Examples of these operations are `CHANGE-TO-MODE2/4` - performed every time a moon blinding occurs or `CPE-MODE-SUMMER` - performed in spring equinox. As goals, we have to define the number of times that the moon blindings occur during the year (thirty-six for year 2001) or the possible modes: summer and winter for CPE mode, so the planner generates the appropriate satellite operations. The operations in this first category do not force the addition of temporal constraints to the operators effects (for more details on how to add temporal constraints see section entitled *Type of knowledge represented in the planner*).

Most of the operations in the second category (and some in the third) are scheduled having in mind the date when a South Maneuver was performed. They are performed every two weeks at an hour corresponding to the *secular drift* direction. In the initial conditions we have the satellite affected by the secular drift, and the date and time in the year during which the Maneuver is going to be performed. Table 1 shows a South Maneuver task description. Given that the maneuvers are forbidden during moon or sun blindings they have to be moved ahead twenty-four or forty eight-hours (in case two moon blindings appear in following days) from the secular drift time. The two maneuvers that follow the first one must be also moved ahead the same number of hours. The rest of the operations in this category start/finish some hours before/after the start/end of the South Maneuver. Some operations cannot be performed if in that week a Maneuver has been scheduled, while others must be performed during Maneuvers.

Table 1. South Maneuver task. It is the ninth of the twenty-six South Maneuvers that will be performed during year 2001.

Name	South Maneuver
Description	Follows the two weeks keeping cycle, operating the satellite in one of every two weeks.
Requirements	On Monday or Tuesday at an hour corresponding to the secular drift. A West and East maneuver must follow it.
Constraints	It can not be performed with Moon or Sun Blinding.
Secular drift	30/04/01 18:02:00
Duration	3 hours

For the operations in the third category, we only need to represent in the initial conditions the last time the operation was performed in each satellite. As with respect to the goals, we have to introduce the times that each operator will be performed during the year. The control rules can help to choose the date for each goal and also assign a given resource to an operation (for example a tank).

THE PLANNING AND SCHEDULING MODEL

PRODIGY [12] is an integrated intelligent architecture that has been used in a wide variety of domains: from artificial domains to real applications as logistics or robotics. The problem solver is a non-linear planner that uses a backward chaining 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. Three types of knowledge are the inputs of the planner.

First, the domain theory that contains all the operations represented by the operators. The Prodigy domain theory is based on the STRIPS representation originally proposed by Fikes and Nilsson [5]. In the STRIPS representation, a world state is represented by a set of logical formulae, the conjunction of which is intended to describe the given state. Then, an operator consists of pre-conditions (conditions that must be true to allow the action execution), and post-conditions or effects (composed of an add list and a delete list). The add list specifies the set of formulae that are true in the resulting state while the delete list specifies the set of formulae that are no longer true and must be deleted from the description of the state. Since this representation is quite restricted, it has been extended to allow disjunctive preconditions, conditional effects and universally-quantified preconditions and effects. Figure 1 shows one of the hundred operators that have been implemented in the HISPASAT domain.

```
(OPERATOR SOUTH-MANEUVER
  (preconds
   ( (<s> SATELLITE)
     (<t> TIMES)
     (<t1> TIMES)
     (<p> PERCENTAGE)
     (<n> DIRECTION)
     (<d> (and DATE (gen-from-pred (moon-blinding <s> <d> <n> <p> <t1>))))
     (<p1> (and PERCENTAGE (over-n <d> 'greater 40)))
     (<d1> (and DATE (is-south-maneuver <d> 3 hours 3 hours)))
     (<dst> (and DATE (subtract-time <d1> 24 hours)))
     (<dend> (and DATE (end-operation <dst> 3 hours))))
   (south-maneuver <s> <t> <d1>))
  (effects
    ()
     ((add (south-m <s> <t>))
      (add (south-man <s> <t> <dst>))))
```

Fig. 1. Operator corresponding to the South Maneuver task of Figure 1. If there is any moon blinding within three hours from the expected maneuver with intensity over 40%, the maneuver has to be moved ahead twenty-four hours. The end date of the task is calculated by the end-operation function.

One has first to define the variables that are used in the operator. For that, one needs to declare the type of each variable. In PRODIGY, types can be defined structured in a hierarchy. The type hierarchy forms a tree structure: each type is a subtype of another type and may have several subtypes. There is a special kind of types that are the infinite types that allow to represent continuos valued variables (i.e. DATE). In our domain, we have among others, the following types: SATELLITE, TIMES, PERCENTAGE, DIRECTION and DATE. The type SATELLITE will instantiate one of the three satellites available, DIRECTION can have the values north or south. The types TIMES and PERCENTAGE could have been defined as numbers, but we have chosen to declare them as discrete variables given that only a finite number of values for them are needed to be considered.

The PRODIGY function *gen-from-pred* generates a list of values to be possible bindings for the corresponding variable by using the information of the current state. In this particular case we use it to generate all the values that the numeric variable DATE can have. DATE represents “seconds since 1900 in GMT” (we have used Common Lisp as the programming language for functions given that PRODIGY is written in Lisp and this is the way Common Lisp handles time). The reason to use this format is for efficiency: it will be faster 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 HISPASAT. Other similar approaches fix the starting point of the computation, call it *time zero*, and schedule all the activities from that point [8, 11, 13].

The rest of functions that appear in Figure 1 have been coded for this particular domain although they are general for any domain with temporal restrictions as will be described later on. For instance, *is-south-maneuver* generates the date of the maneuver (if there is any) that overlaps within three-hours any moon blinding. If the blinding intensity is over 40%, the maneuver must be moved twenty-four hours ahead (the function *over-n* calculates if the percentage of the

moon blinding is over forty). The *subtract-time* function subtracts twenty-four hours from the date of the expected maneuver, and *end-operation* calculates the end of the operation from the start time and the duration; three hours in this case. We have a similar function *add-time* that adds some time to a date and also helps to define temporal constraints into the operators.
With respect to the pre-conditions of this operator, it only has one: *(south-maneuver <s> <t> <d1>)*. As effects, the predicates *south-m* and *south-man* that belong to the add list.
The second input to the planner is the problem, described in terms of an initial state and a set of goals to be achieved. Figure 2 shows a small fraction of the initial state, that is, all the events that will occur during the year. For moon blindings we need to know the satellite affected by the blinding (*1B*), the date in seconds during which it will take place (3188333400 = 3/1/2001 01:10:00), the direction (*south*) and the intensity (45% represented by *p-45*). Finally, the moon blinding chronological appearance, that is, the first time it appears is represented by *t1*, the second time *t2*, etc.
With respect to the goals, we need to perform twenty-six maneuvers during the year.

```
(state
  (and
     (moon-blinding-st  1B 3188333400 south p-45 t1)
     (moon-blinding-end 1B 3188334000 south p-45 t1)
     (spring-equinox 3194121600)
     (last-antenna-maint 1B 3172893395)
     ...
     (south-maneuver 1B t1 3194035800)
     (south-maneuver 1B t2 3195245400)  ...))

  (goal(and
         (south-m 1B t1) (south-m 1B t2)...
         (south-m 1B t25) (south-m 1B t26)))
```

Fig. 2. Some goals and initial conditions for the *SOUTH-MANEUVER* operator. The number of times that is performed during the year is twenty-six. We would have to add the initial state and goals for the rest of operations.

As a result, PRODIGY generates a plan with the sequence of operators that achieves a state (from the initial state) that satisfies the goals. More importantly, the plan has also in mind the temporal constraints among the operators. The obtained plan does not consider any optimisation with respect to resource use or availability, given that for HISPASAT it is enough to find a plan. However, the planner could obtain an optimal plan according to some criteria using the QPRODIGY version described in [2].
When there is more than one decision to be made at a 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. Figure 3 shows an example of a selection binding rule. It says that if the date is around autumn equinox and we can apply the operator *TANK-SWAPPING* then we should select the tank NTO1. Each satellite has four tanks. Any of them could be chosen for swapping but there is a recommendation to use a given tank in a period of the year. This control rule prunes the search tree and chooses the tank for that period of the year.

```
(Control-rule select-tank_NTO1
  (if (and (current-goal (swapped <s> <tank>))
           (current-ops (TANK-SWAPPING))
           (true-in-state (swapped <s> NTO3))
           (true-in-state (equinox-au <d>))))
  (then select bindings (( <tank> . NTO1))))
```

Fig. 3. Control Rule to select the tank NTO1 in autumn equinox.

TYPE OF KNOWLEDGE REPRESENTED IN THE PLANNER

In this section we give an overview of the scheduling concepts that we had to face in order to give a solution to the nominal operations of HISPASAT using a planner.

Time

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 allow to add constraints among and within operators. PRODIGY can handle the seven Allen primitive relations between temporal intervals [1]. We have grouped them in the following types (for pedagogical reasons we have reduced the syntax. Go to Figure 1 to see how numbers are generated using the function *gen-from-pred*):

- *Operation-A* has a duration: the duration is the time required for *operation-A* to execute. This value is used to calculate the end/start time of *operation-A*. This is represented using the function add-time (*DUR* is a number, integer or not, that represents the duration, and *TIME* represents if the duration is in seconds, minutes, hours, days or months):

```
Operation-A
  preconds: (<dst> (start-process <dst>))
            (<dend> (add-time <dst> DUR TIME))
  effects:  (started-A <dst>)
            (finished-A <dend>)
```

- The end time of *operation-A* is the start time of *operation-B*. The following Allen relations belong to this type: *operation-A* before *operation-B* and *operation-A* meets *operation-B*. Then, the elapsed time from the end of *operation-A* can be a value that can be constrained, in the general case by an interval [a, b]. The limits can be zero or positive numbers. The function *add-time-in-interval* returns all possible values in that interval. The way this is handled is as follows (the variable DUR^0 represents the duration, $ELAPSED^1$ and $ELAPSED^2$ represent the minimal and maximal values of the interval [a, b] and $TIME^0$, $TIME^1$, $TIME^2$ represent seconds, minutes, hours, days or months):

```
Operation-A
  preconds: (<dst> (start-process <dst>))
            (<dend> (add-time <dst> DUR TIME))
  effects:  (finished-A <dend>)
Operation-B
  preconds: (<d> (finished-A <d>))
            (<dst> (add-time-in-interval <d> ELAPSED1 TIME1 ELAPSED2  TIME2))
            (<dend> (add-time <dst> DUR0 TIME0))
  effects:  (started-B <dst>)
            (finished-B <dend>)
```

- The start time of *operation-A* is the start time of *operation-B*. The following Allen relations belong to this type: *operation-A* starts *operation-B* and *operation-A* equals *operation-B*. Then, the elapsed time from the start of *operation-A* can be a value that can be constrained by an interval [a, b] as the previous case. The way to represent this is as follows:

```
Operation-A
  preconds: (<dst> (start-process <dst>))
            (<dend> (add-time <dst> DUR TIME))
  effects:  (started-A <dst>)
            (finished-A <dend>)
Operation-B
  preconds: (<dst> (started-A <dst>))
            (<dend> (add-time-in-interval <dst> ELAPSED1 TIME1 ELAPSED2 TIME2))
  effects:  (started-B <dst>)
            (finished-B <dend>)
```

- The end time of *operation-A* is the end time of *operation-B*. The *operation-A* finishes *operation-B* belongs to this category. The case *operation-A* equals *operation-B* could have also been in this category. The start time of *operation-B* computed from the end of *operation-A* can be a value that can be constrained, in the general case by an interval [a, b]. The limits can be zero or positive numbers. The function *del-time-in-interval* returns all possible values in that interval. This is represented as follows:

```
Operation-A
  preconds: (<dst> (start-process <dst>))
            (<dend> (add-time <dst> DUR TIME))
  effects:  (finished-A <dend>)
Operation-B
  preconds: (<dend> (finished-A <dend>))
            (<dst> (del-time-in-interval <dend> ELAPSED1 TIME1 ELAPSED2 TIME2))
  effects:  (started-B <dst>)
            (finished-B <dend>)
```

- The start time of *operation-B* starts in a range from the start of *operation-A*, that is, *operation-A* overlaps *operation*-B. This is handle as follows:

```
Operation-A
  preconds: (<dst> (start-process <dst>))
            (<dend> (end-operation <dst> DUR TIME))
  effects:  (started-A <dst>)
            (finished-A <dend>)
Operation-B
  preconds: (<d> (started-A <d>))
            (<dst> (add-time-in-interval <d> ELAPSED1 TIME1 ELAPSED2 TIME2))
            (<dend> (add-time <dst> DUR0 TIME0))
  effects:  (started-B <dst>)
            (finished-B <dend>)
```

- The start time of *operation-B* starts after the start time of *operation-A* and its end time ends before the end time of *operation-A*. The Allen relation that belongs to this type is *operation-A* during *operation-B*. The way to represent the last category is (the variables $ELAPSED^3$ and $ELAPSED^4$ represent the minimal and maximal values of the interval [a, b] and $TIME^3$ and $TIME^4$ represent seconds, minutes, hours, days or months):

```
Operation-A
  preconds: (<dst> (start-process <dst>))
            (<dend> (end-operation <dst> DUR TIME))
  effects:  (started-A <dst>)
            (finished-A <dend>)
Operation-B
  preconds: (<d> (started-A <d>))
            (<d1> (finished-A <d1>))
            (<dst> (add-time-in-interval <d> ELAPSED1 TIME1 ELAPSED2 TIME2))
            (<dend> (del-time-in-interval <d1> ELAPSED3 TIME3 ELAPSED4 TIME4))
  effects:  (started-B <dst>)
            (finished-B <dend>)
```

Resources

In PRODIGY there is no provision for specifying resource requirements or consumption. But resources can be seen as variables that can have associated values through literals that refer to them. For example, if tank T1 has fuel with 30 litres in it, we would represent it as (has-fuel T1 30). Then, in the operators, we can restrict 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 through a functional expression. For instance, we can define a function that calculates the fuel consumed in each maneuver as a function of the available fuel and the angle of the satellite respect to the sun. The predicate *(position <satellite> <angle>)* is a pre-condition in the operator *Operation-Maneuver-S*:

```
Operation-Maneuver-S
  preconds: (<available> (available-fuel <satellite> <available>))
            (<fuel-consumed> (compute-consumed-fuel <available> <angle>))
            (position <satellite> <angle>)
  effects:  (performed-maneuver  <satellite>)
```

EXPERIMENTAL RESULTS

Tables 2 and 3 show the performance of Prodigy for the Hispasat domain. Both tables show the time performance in seconds, the number of goals, the number of nodes expanded, operations needed and the time in seconds for a node expansion. In the first experiment, we have used the satellite 1B increasing the number of satellites from two to five.

Table 2.Time in seconds for the 1B satellite.

Number of satellites	Time in seconds	Goals	Number of nodes	Operations	Time per nodes
2	1896	494	3890	878	0.48
3	4527	741	5833	1317	0.78
4	9194	988	7776	1756	1.18
5	16349	1235	9719	2195	1.68

Table 3 shows the results obtained for each satellite and the three satellites at the same time. The big difference in performance between satellite 1C and 1A & 1B is due to functions like *is-south-maneuver* (see Figure 1) that checks that a maneuver does not interfere with a moon blinding. In the 1C operators these functions are not coded because the software included in the satellite avoids this type of interferences. Also, this difference can be appreciated between satellites 1A and 1B: even if the number of goals in 1B is higher than 1A, the satellite 1A has two more moon blindings (36) that can affect the maneuvers. This is translated into more check time for each maneuver goal.

Table 3.Time in seconds for each satellite and the three together.

Satellite	Time in seconds	Goals	Initial conditions	Number of nodes	Operations	Time per nodes
1A	548	215	93	1837	411	0.30
1B	517	247	91	1947	439	0.26
1C	120	221	160	1269	303	0.09
1A-1B-1C	4000	683	344	5045	1153	0.79

All these results have been done in a Pentium III 800 MHz and 256 Mb under Windows using a demo version of Allegro CL. The same results in a Linux distribution take half of the time. The number of operators in the Hispasat domain is of one hundred.

RELATED WORK

Several planning systems have addressed the applications of limited resources and time considerations. The planner NONLIN+ [10] maintains information about the duration of the activities and the ability of representing limited resources. SIPE [14] allows the declaration of the use of resources in the operators. DEVISER [13] could also handle consumable resources. These planners have been designed with the purpose of explicitly handling some types of resources. In the IPP planner [6], based on Graphplan [3], the search algorithm has been modified to combine the ADL search algorithm to handle logical goals together with *interval arithmetics* to handle resource goals. This is the main difference with PRODIGY that has been designed as a general planning and learning system rather than a scheduler. But this is not inconvenient to represent this type of knowledge and to reason about it without modifying the search algorithm.

Other more modern approaches have integrated planning and scheduling in the same system as HSTS by instantiating state-variables into a temporal database [8]. Also the O-PLAN2 integrates planning, scheduling, execution and monitoring within a system that also uses an activity based plan representation [11]. Other systems are temporal-based planners, such as IxTeT, that uses a graph search algorithm and an interval action representation [7]. They all differ from our approach in that they had to (re-)implement the planners to handle scheduling information, while we have used a standard planner and defined a series of simple time handling functions that allow to introduce resources in an intuitive way almost equivalent to the expressive power of other tools.

Also ASPEN [4] uses an AI planner with a richer representation than ours in activities that easily allow the introduction of start times, end times, duration and use of one or more resources. Some algorithms that this planner uses to solve the problem belong basically to the scheduling area (i.e., the schedule iterative repair algorithm). On the other hand, our planner uses specific algorithms of the planning area that perfectly handle the scheduling problem.

CONCLUSIONS AND FUTURE WORK

Satellites provide a challenging domain for applying AI techniques. Due to the use of time and resources, scheduling techniques have been traditionally applied. We present how the deliberative planner PRODIGY can solve scheduling problems thanks to its capability of defining numerical variables, coding functions and the use of control rules. The functions allow the user to obtain variables values, calculate the duration, specify resources constraints and relationships among operators. The control rules help to reduce the search and then the time spent obtaining the solution. The planning techniques provide a successful solution to the ground scheduling operations during the year for the three satellites in Orbit.
Actually we are developing the interface and we estimate that the system will be operational at the end of the year.
Further research will explore hybrid strategies as the integration of PRODIGY with a scheduler and comparing the efficiency between the two systems. We also want to use SHAMASH [9] a Knowledge Base System that handles time and resources constraints to analyse its behaviour related to the others systems.

ACKNOWLEDGEMENTS

We want to thank all the HISPASAT Engineering Team for their help and collaboration shown during the developing time of this project, especially Arseliano Vega, Pedro Luis Molinero and Jose Luis Novillo. This work was partially funded by the CICYT project TAP1999-0535-C02-02.

REFERENCES

[1] Allen J.F. Towards a general theory of action and time. Artificial Intelligence, 23 (2): 123-154, 1984.
[2] Borrajo D., Vegas S. and Veloso M. Quality-based Learning for Planning. Working notes of the IJCAI'01 Workshop on Planning with Resources. IJCAI Press. Seattle, WA (USA). August 2001.
[3] Blum and M. Furst, Fast Planning Through Planning Graph Analysis, Artificial Intelligence, 90:281--300 (1997).
[4] Chien S., Rabideau G., Knight R., Sherwood R., Engelhardt B., Mutz D., Estlin T., Smith B., Fisher F., Barrett T., Stebbins G. and Tran D. ASPEN Automating Space Mission Operations using Automated Planning and Scheduling. SpaceOps 2000, Toulouse, France, June 2000.
[5] Fikes R. E, Nilsson N. J. STRIPS: a new approach to the application of theorem proving to problem solving . In Artificial Intelligence, 2:189-208.
[6] Koehler J., Nebel B., Hoffmann J., and Dimopoulus Y. Extending Planning Graphs to an ADL subset. Proc ECP-97. 1997.
[7] Laborie P. and Ghallab M. Planning with sharable resource constraints. In Proceedings 14th Int. Joint Conference on AI, 1995.
[8] Muscettola, N. HSTS: Integrating Planning and Scheduling. In Intelligent Scheduling, Ed. M. Zweben & M.S. Fox. pp 169-212. Morgan Kauffmann, 1994.
[9] Sierra-Alonso A., Aler R. and Borrajo D. Knowledge-Based Modelling of Processes. Working notes of the IJCAI'99 workshop on Intelligent Workflow and Process Management: The New Frontier for AI in Business. Stockholm, Sweden 1999. IJCAI Press.
[10] Tate A. and Whiter A.M. Planning with Multiple Resource Constraints and an Application to a Naval Planning Problem. First Conference on the Applications of AI. Denver, Colorado, USA. December 1984.
[11] Tate A., Drabble B. and Kirby R. O-Plan2: An Open Architecture for Command, Planning, and Control. In Intelligent Scheduling, ed. M. Zweben & M.S. Fox. pp 213-240. Morgan-Kauffmann, 1994.
[12] Veloso M., Carbonell J., Perez A., Borrajo D., Fink E., and Blythe J. Integrating planning and learning: The PRODIGY architecture. Journal of Experimental and Theoretical AI, 7: 81-120, 1995.
[13] Vere S. Planning in Time: Windows and Durations for Activities and Goals. IEEE Trans. On Pattern Analysis and Machine Intelligence. PAMI-5, nº3, pp. 246-267. 1983.
[14] Wilkins D.E. Practical Planning Extending the classical AI planning paradigm. Morgan-Kaufmann. 1988.