The Role of Different Solvers in Planning and Scheduling Integration

Federico Pecora[1] and Amedeo Cesta[1]

Planning and Scheduling Team
Institute for Cognitive Science and Technology
Italian National Research Council
{pecora, cesta}@ip.rm.cnr.it, http://pst.ip.rm.cnr.it

Abstract. This paper attempts to analyze the issue of planning and scheduling integration from the point of view of *information sharing*. This concept is the basic bridging factor between the two realms of problem solving. In fact, the exchange of each solver's point of view on the problem to be solved allows for a synergetic effort in the process of searching the space of states. In this work, we show how different solving strategies cooperate in this process by varying the degree of integration of the combined procedure. In particular, the analysis exposes the advantage of propagating sets of partial plans rather than reasoning on sequential state space representations. Also, we show how this is beneficial both to a component-based approach (in which information sharing occurs only once) and to more interleaved forms of integration.

1 Introduction

The reason for the richness of research and debate in the area of planning and scheduling integration is that the solving techniques for planning and scheduling problems are fundamentally different. On one hand, a planner reasons in terms of causal dependencies: by performing logical deduction in a predefined logic theory, it is capable of deriving sets of actions which have to be performed in order to achieve a preset goal. On the other hand, scheduling involves the propagation of constraints, which in turn determines the domains in which the searched assignment of variables is to be found.

Both planning and scheduling are, basically, search algorithms. Planners and schedulers implement efficient techniques for (1) representing the search space and (2) finding solutions in the search space. Thus, research in planning and scheduling technology alike is directed towards, on one hand, finding data structures which represent the problems, and evolving the techniques with which these data structures are generated and explored on the other.

The goal of this paper is to assert some basic principles related to planning and scheduling integration. In particular, we formulate the idea of an integrated system in terms of *information sharing*, that is the level of synergy between the planning and the scheduling solving cores. These considerations will enable us to identify the solving algorithms that are best fit for planning and scheduling integration.

2 Integrating Planning and Scheduling

Planning and scheduling address complementary aspects of problems, and this makes them a natural couple. As a consequence, research in the field of integrated planning and scheduling has recently produced some interesting results.

Many approaches to planning and scheduling integration have been proposed. Some deal with time and resources contextually with causal reasoning, that is they expand a strictly causal solving technique (such as GRAPHPLAN) with data structures capable of modeling time and/or resources. Some examples of this approach are given in [8, 9, 11, 17].

Another approach to integrating planning and scheduling capabilities in a single solving tool is to implement the two solving paradigms separately, and then to link them in order to solve the two aspects of the problem with two distinct solving techniques. This, which may seem the most realistic approach, presents some major difficulties since it requires the definition of forms of *information sharing* between the two subsystems. Indeed, in this paper we would like to assert the need for an in-depth comprehension of some fundamental aspects related to this form of integration of planning and scheduling. In particular, we will try to answer the following questions:

When do we need an integrated reasoner? This equates to identifying which sorts of problems require such combined reasoning capabilities. Thus, we will try to single out the characteristics of these problems in terms of whether they are fit for causal reasoning (what we have called planning) or time- and resource-related reasoning (scheduling).

Which planning and scheduling technology is to be used? This issue is clearly a very open one. The goal of this paper is to assert some fundamental principles of planning and scheduling integration. This involves the theorization of what we have called a Naïve Component-Based Approach (N-CBA), in which there is an explicit distinction between the two aspects of problems. We will see that the considerations which can be made in the N-CBA conform to those made in a more in-depth analysis of strongly interwoven planning and scheduling integrations.

In the following sections we will address the two questions contextually, by investigating the role of two solving ideas in the context of the N-CBA and of more tightly integrated approaches.

3 A Naïve Component-Based Approach

Probably the most immediate form of integration we can think of is to serialize a planner and a scheduler (hence the name Naïve Component-Based Approach, or N-CBA), as depicted in figure 1. A planner is given a problem specification along with a domain description *which is purified with respect to time- and resource-related constraints*. These constraints are accommodated after the planning procedure has taken place, i.e. has produced a Partially Ordered Plan (POP). The

plan adaptation procedure is responsible for inserting this additional information. A scheduler is then employed in order to obtain a completely instantiated solution of the initial planning problem.

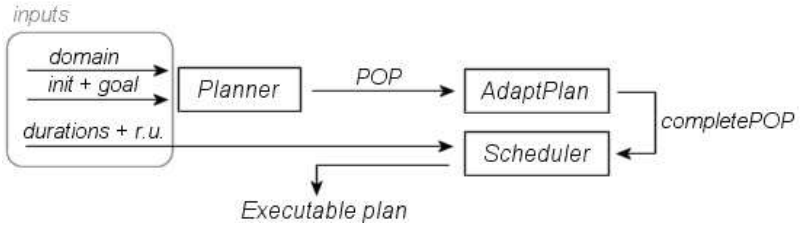

Fig. 1. Architecture of the N-CBA.

This approach is strongly intuitive, but we believe it is instrumental for the comprehension of the phenomena underlying the integration of the two processes. In fact, the *choice of components* for the serialized planning and scheduling system exposes very clearly the relative fitness of particular planning and scheduling solving strategies. In other words [1], studying both processes in a separate way has the effect of improving both their performance in the serialized setting and the performance of a "truly" integrated system (in which planning and scheduling are more tightly interwoven) which makes use of these solving algorithms.

It should be clear that the N-CBA is certainly not the best way to build an integrated reasoner, since its efficiency relies very strongly on *how separable* the two aspects of the problem are. This requirement is often not realistic, since the degree of inter-dependency between causal and time/resource-related constraints in the problem are much higher. Nonetheless, it should be clear in a moment why we have chosen to start our analysis on this very simple architecture.

The N-CBA has the nice property of delivering a very clear picture of what is going on during problem resolution. In particular, this architecture exposes the following interesting characteristics of planning and scheduling integration:

Information sharing The necessity of both a planning and a scheduling solver derives from the fact that they infer different types of information. Thus, it is necessary for the two solvers to exchange *their* view of the problem which is being solved, so as to take advantage of the different perspectives from which the reasoning takes place.

Deductive bottlenecks When it comes to solving challenging problems, we are interested in understanding exactly where the difficulties are in the deductive process. In a tightly coupled integration this may be difficult to verify. The analysis of a more loosely coupled architecture like the N-CBA may give interesting indications as to which difficulties can be encountered by the single solving strategies, thus on one hand biasing the choice of the type of strategy, and on the other exposing the necessary adaptation measures to be taken on the strategy.

Let us first of all focus on the information sharing aspect of the N-CBA, which constitutes the effective binding of the planning and scheduling subsystems in any integrated approach.

3.1 Information Sharing

The point of integrating planning and scheduling solvers is that they address complementary issues. More specifically, the responsibility of the two solving algorithms is to *validate partial plans with respect to the particular constraints they propagate*. In other words, integrated planning and scheduling systems solve problems by mutually validating the decisions which are taken during partial plan construction with respect to the causal and the time/resource-related constraints.

In general, information sharing between causal and time/resource solvers implements a sort of mutual "advice-giving" mechanism. In this context, the N-CBA is furnished with the most basic form of this type of information sharing, in which the scheduler validates the entire plan, rather than making use of a tight synergy at the partial plan level. Clearly, the ideal form of an information sharing mechanism is one in which this mutual interrogation occurs at every decision point, and, indeed, CSP approaches seem to be moving in that direction.

But even the N-CBA has a rather interesting property, namely that it gives us an indication as to which solving strategies support information sharing mechanisms. To this end, let us see which characteristics the single components of an integrated planning and scheduling system should have.

3.2 Solving Strategies

What we have said so far is instrumental in understanding which characteristics we have to look for in order to theorize and implement a more tightly integrated planning and scheduling architecture.

Problems in which scheduling is necessary are typically such that the solution contains concurrent actions (at least at the logical level). In the N-CBA, the scheduling phase is concerned with enforcing the "executability" of a partially ordered plan, so we can be sure that on the causal level there will be a relatively high degree of concurrency to deal with.

The first thing one should notice is that we have taken for granted the use of a PO planner rather than a Total Order (TO) planner as the first component in the N-CBA. This is clearly necessary, given the fact that scheduling is seen as a post-processing phase. Alternatively, the option would have been to cascade a de-ordering phase after the TO planner, in order to establish the concurrency among the actions in the plan for the scheduling phase.The relative convenience of adopting such a strategy is questionable since the computational aspects of de-ordering and re-ordering plans are not trivial. Depending on the criterion with which the de/re-ordering is done, this problem can be NP-Hard [1].

On the other hand, a more realistic approach to planning and scheduling integration is to allow the two solvers to share information in a more constant

fashion during problem solving. In other words, if we see the N-CBA as an approach in which information sharing occurs only in *one* point, namely when the planning phase is done, the most sophisticated form of integration we can think of is one in which such information is exchanged *during* the problem solving. In order to achieve this, the best we can do is to provide the means for each solver to verify *at each decision point* the effects of its decision from the other solver's point of view. Let us see how such an interwoven integration can be realized with a TO solver.

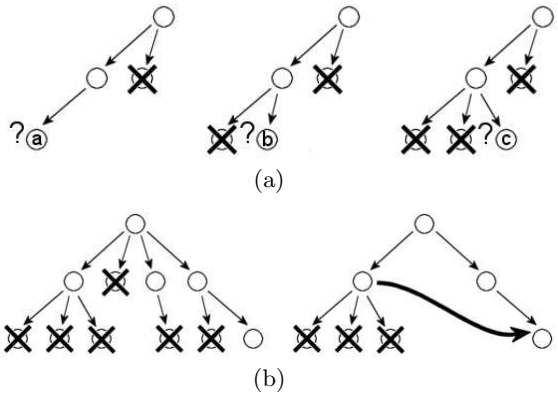

Fig. 2. Searching the state space (a) and backtracking vs. de-ordering (b).

TO planners are typically implemented as Heuristic State Space Search (HS^3) algorithms [4, 10, 18]. The state space representation of such planners is very explicit with respect to the decision points of the causal planning phase. Every node represents a "choice" for the planning algorithm, where it relies on a heuristic which gives the solver an indication as to which node to branch over in the following step. For instance, figure 2(a) depicts three successive decisions over the nodes of the state space (nodes represent states while edges represent the actions which lead to them).

In the figure, nodes a, b and c are successively considered with respect to the heuristic and, in an integrated reasoner, also with respect to time and resource feasibility. Now, let us suppose that the subtree which has node c as root is pruned because of time or resource considerations. There are two options for the solver (see figure 2(b)): one is to backtrack in order to explore other regions of the state space, hoping to encounter another node which is time- and resource-feasible. But a more "educated guess" is to de-order the partial plan obtained up to node c. In fact, since the partial plan failed at node c because of time- and resource-related constraints, there is a certain probability that an alternative ordering of the actions taken so far (one that respects the causal constraints of the domain) could enable the planning to go on. Indeed, this approach is adopted with a certain degree of success in [20].

Notice, though, that we would again have to perform de-ordering. It was to be expected that planning taking into account time and resources is inevitably more expensive computationally that just planning in causal terms. But let us see another way of achieving high levels of integration.

The principal pitfall of the TO-based strategy is that in order for the planner to *see* the time- and resource-feasible node, it must somehow be aware of the existence of the alternative evolution of the state, and this, in turn, requires either de-ordering or backtracking.

So the natural consequence of this is that there is at least one reason for which such a state representation does not seem to match with planning and scheduling integration: the propagation of causal constraints does not take into account all possible evolutions of the state. In other words, the causal decision making during the process of planning is strong to the point that it is retractable only by performing an expensive "roll-back".

It appears obvious that an integrated planning and scheduling architecture must be theorized so as to give *equal importance to both types of deductive processes*. To put it more simply, the reason for which an action can be non-applicable during plan synthesis can be traced back to both causal- *and* time/resource-related considerations. It is clear that the only way of ensuring equal dignity to both solvers is to propagate *sets* of possible choices with respect to causal and time/resource feasibility.

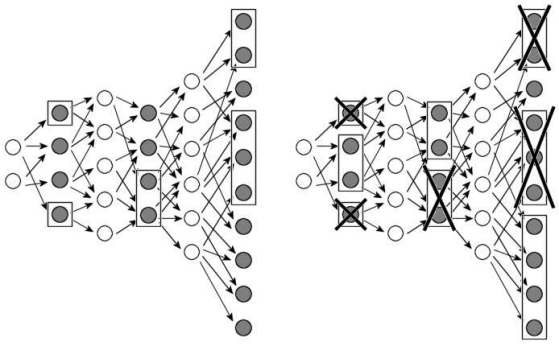

Fig. 3. Alternate plans are propagated in the planning graph.

In contrast with HS^3 planners, GRAPHPLAN-based planners [3] maintain a planning graph data structure, which represents the *union* of all applicable courses of action given the initial state and the applicable actions. As noted in [12], a planning graph is a sequence of alternated predicate and action levels which correspond to a unioned representation of a state space search tree. The basic solving strategy of GRAPHPLAN-based planners consists of propagating pairs of action and predicate levels in successive steps. At each step, the

planning graph is "searched" for a solution[1]. This representation of the search space is rather different from the tree structure in HS[3] planners for one reason in particular: it contemplates *concurrent* actions. In other words, while a tree representation of the state space is fundamentally sequential, a planning graph is capable of representing in a compact way the parallel execution of multiple actions. This means that if some actions which are necessary in order to achieve the goal state do not depend on each other, then they will be present in the solution in the same *logical step*, the solution being a *partial order* of actions.

The planning graph state space representation used in GRAPHPLAN-based planning is far more effective from this point of view. A planning graph is a data structure which maintains, as the planning goes along, *all* admissible partial plans (and also non-admissible ones due to the propagation of only pair-wise mutex constraints [12]). Indeed, a planning graph represents the *space of plans* rather than the space of states. Thanks to this, information sharing in a tightly coupled integration based on a GRAPHPLAN-like solver is much more realistic. As depicted in figure 3.2, the exclusion of a partial plan does not require expanding the state space representation, since it already "contains" all alternative partial plans up to the current point of the solving process. Of course, singling out another candidate partial plan is still NP-Hard. But, as we will see in the next section, there is one reason for which this seems to be more advantageous.

3.3 BLACKBOX and HSP

While GRAPHPLAN itself can be considered out of its league among the best competition planners, planners such as BLACKBOX [13] and IPP [14], which are based on the GRAPHPLAN paradigm, certainly are not. Our tendency to think that GRAPHPLAN-based solvers seem to be better suited for planning and scheduling integration is motivated by the comparison of BLACKBOX and HSP (a well-known heuristic search planner [4]) on three significant domains taken from the AIPS Planning Competition: Freecell, Rovers and Driverlog.

The experimental base we have considered shows that there is an interesting separation of classical planning domains in two categories: those which can be solved by HS[3] planners, and those which are more suited for GRAPHPLAN-based planners. Figures 4(a) and 4(b) show some planning details for problems on the three mentioned domains.

Freecell is a familiar solitaire game found on many computers, involving moving cards from an initial tableau, constrained by tight restrictions, to achieve a final suit-sorted collection of stacks. The problems in this domain show quite a good variety of difficulty, which makes this game rather challenging for most planners.

The first trend we should notice is that BLACKBOX performs very badly on the Freecell domain. This should not be surprising, since the domain design for this card game is certainly not a parallel one: there is one player, and all actions it

[1] BLACKBOX extracts solutions by employing SAT algorithms [15, 16] on a CNF formula obtained from the planning graph.

Problem	BlackBox	HSP
roverprob1234	117	–
roverprob4213	102	–
roverprob6232	104	–
roverprob2435	7882	–
roverprob1423	583614	–
roverprob5146	17868	–
roverprob1425	15482	–
roverprob4135	–	–
roverprob5142	–	–
roverprob5624	–	–
FreeCell2-4	119957	50
FreeCell3-4	1224730	230
FreeCell4-4	–	470
FreeCell5-4	–	1800
FreeCell6-4	–	6630

(a)

Problem	BlackBox	HSP
DLOG-2-2-2	29	10
DLOG-2-2-3	777	10
DLOG-2-2-4	134	20
DLOG-3-2-4	1192	30
DLOG-3-2-5	5805	20
DLOG-3-3-5	123	30
DLOG-3-3-6a	1550	30
DLOG-3-3-7	6056	50
DLOG-2-3-6a	8255	90
DLOG-2-3-6b	9427	80
DLOG-2-3-6c	14541	200
DLOG-2-3-6d	–	1890
DLOG-2-3-6e	–	470
DLOG-3-3-6b	98677	4550
DLOG-4-4-8	–	21960

(b)

Fig. 4. Details for the BlackBox and HSP planning instances for problem instances on the Rovers and Freecell (a) domains, and on the Driverlog (b) domain — time unit is milliseconds.

decides to carry out are necessarily sequential (with some, very few exceptions). With some difficulties, BlackBox managed to solve the first two problems, but finally tossed the towel when it came to tackling the more difficult ones. On the other hand, HSP is capable of solving all problems of the Freecell domain rather easily. Notice also that HSP overwhelmingly outperformed BlackBox on all problems.

The Rovers domain models a multi-robot system for exploring an unknown extraterrestrial environment, in which soil-sampling, imaging and data transmission tasks have to be carried out. The problem solving instances described in figure 4(a) show very clearly that this particular domain is rather suitable for planning with BlackBox, which solved most problems rather fast.

For the sake of completeness, we should mention that the same problems are extremely difficult to solve for HSP, which could not cope with any of the them[2] before timing out. Dually with respect to the Freecell problems, we can see that a high degree of concurrency is advantageous for solution extraction, thus making BlackBox quite competitive while planners such as HSP, which do not model resources, are left choking in the dust.

We have seen sets of problems in two domains, which somehow represent the "extremes" of classical planning: Freecell is a one-player card game, in which

[2] For the first two problems, HSP is not capable of finding a solution and reports this within a few seconds. This is due to the fact that, in this mode of operation, HSP does not ensure completeness.

practically no parallelism is possible, while the Rovers domain models a highly concurrent application for multiple robots.

This albeit brief experimental evaluation would not be complete without something that resembles a compromise. The well-known Driverlog domain seems to fit the role, namely for its mixed characteristics: on one hand, tasks can be concurrent, such as multiple drivers and trucks involved in the delivery of goods; on the other hand, the domain specifies that trucks have to be loaded and unloaded, and drivers have to reach, board, drive and disembark their trucks. In short, this makes the Driverlog domain not clearly suitable for either of the planning approaches. It turns out, in fact, that most problems are solvable by both BlackBox and HSP, the latter having a slight advantage.

Let us analyze the above domains with respect to their planning and scheduling characteristics. Freecell is a rather "pure" example of a planning domain, given the fact that all actions (moves) are motivated by logical deduction given the situation and the goal or subgoal in a certain logical step. Problems in the Freecell domain do not contemplate any time- or resource-related constraints, nor for that matter is it possible to have two or more actions in parallel since the game is strictly sequential. On the other hand, problems in the Rovers domain are typically scheduling problems in that they could be specified in terms of duration of the actions and their usage of on-board instruments.

A significant trait of the three domains is *parallelism* (vs. *sequentiality*). Clearly, a planning problem specification which allows for multiple tasks to be executed in parallel generally produces *contention peaks* on resources, a problem which is naturally addressed by a scheduler. Thus, such planning problems pose a scheduling problem composed of partially ordered, causally dependent actions which must be instantiated in time so as to satisfy resource capacity constraints.

In the N-CBA it is clear that the preferred planner would be BlackBox, since it performs better than HSP on the problems for which it makes sense to apply an integrated planning and scheduling approach. This result seems to go in the same direction as the considerations we made earlier for tightly integrated systems, namely that in domains which require high concurrency, it is most likely that resource-related considerations are required while planning; because of this, since alternative action orderings are more promptly available thanks to the structure of the planning graph, we can make a case for investigating a Graphplan-based approach to planning and scheduling integration.

4 The N-CBA as a Lower Bound

So far we have given a glimpse of the issues which have to be taken into consideration in the theorization of an integrated planning and scheduling solution. From the previous considerations we can derive that not all planning paradigms are well suited for an integrated approach.

An integrated planning and scheduling reasoner built according to the N-CBA is reported in [19]. In this section we will give an overview of this implementation of the N-CBA which makes use of established off-the-shelf components

for the planning and scheduling modules. The choice of these components (or better, of the planning system) is strongly grounded on the previous considerations. Our aim is to assert a basic loosely-coupled system according to the N-CBA, which can be to some extent taken as the "best we can do" with the most primitive form of information sharing.

The planning system which has been used is BlackBox, a Graphplan-based planner which combines the efficiency of planning graphs with the power of SAT solution extraction algorithms [15, 16]. This planner has done very well in recent planning competitions, and in the category of those planning paradigms which seem fit for planning and scheduling integration, it can certainly be considered to be one of the best choices. The scheduler which has been selected for use in this implementation of the N-CBA is O-Oscar [7], a versatile, general purpose CSP solver which implements the ISES algorithm [6].

The result is an architecture which is effectively capable of solving quite a number of problems, and which has been extensively used in the development of a preliminary Active Supervision Framework [19] for the RoboCare project. It is not trivial to notice that even this primitive form of integration has some important advantages:

- the only necessary additional development is the adaptation procedure, in other words, the information sharing mechanism;
- assuming the separability of causal and time/resource-related problem specifications, the system performs quite well, especially with respect to plan quality;
- it is a general purpose tool, thanks to the PDDL planner input, and to the high expressivity of the scheduling problem specification [5, 2];

The performance details of this implementation of the N-CBA are outside the scope of this paper. These issues are extensively reported in [19], in which the N-CBA has been used to solve problems on a multi-agent domain for a health care institution.

Clearly, a more tightly integrated system is often necessary, especially because causal and time/resource-related aspects of the domain are often strongly dependent. Nonetheless, it is interesting to notice how the basic properties of solving ideas and their relative adherence to the philosophy of integration are somehow independent from *when* the information sharing occurs (at partial plan level, at every decision point and so on), rather they depend on *which* information is shared between the two solving engines. It is clear already in the N-CBA that the most effective solution seems to be that of adopting strategies which are capable of propagating plans rather than states, in order to ensure fast reactivity when it comes to taking decisions.

In the previous sections we have motivated the use of a Graphplan-based planner in conjunction with a CSP scheduling tool. The most primitive approach to planning and scheduling integration is clearly the N-CBA, an approach in which information sharing occurs only once during the solving process. Given the coarseness of this integrated solving strategy, our opinion is that more sophisticated implementations of integrated solvers should be compared to the

N-CBA[3]. In fact, the overall performance of an integrated architecture can only get better with higher occurrences of information sharing events between the two solvers.

5 Conclusions and Future Work

In this paper we have presented an analysis of the issues involved in planning and scheduling integration. Many approaches to this problem have been developed, yet some fundamental aspects still remain to be explored fully. The work we have presented is motivated by the fact that while the independent research in planning and scheduling has evolved very rapidly, the study of the combined approach is still trying to deal with some basic issues.

The intent of this paper is to analyze some of these issues starting from the N-CBA, an approach to integration which implements the simplest form of information sharing. Integrated architectures in general can be distinguished by two parameters: *when* information sharing is carried out, and *what* information is shared between the two solvers. We have shown how higher degrees of integration are obtainable from the N-CBA by "toggling" the first parameter, where the optimal strategy is to cross-validate causal and time/resource-related aspects of the problem at every decision point.

The study of the N-CBA is interesting because it exposes some characteristics which are common to all forms of integration. In particular, we have focused on which solving strategies are more applicable in an integrated solver. In this context, we have found that both in the N-CBA and in more tight integrated approaches, a desired property of the solvers is to propagate sets of possible choices with respect to causal and time/resource feasibility.

The analysis of the N-CBA has shown how a variety of useful indications can be drawn with respect to the general issue of planning and scheduling integration: thanks to these considerations, we are aiming at furthering the study in this direction, looking in particular at CSP approaches to planning.

Acknowledgments

This research is partially supported by MIUR (Italian Ministry of Education, University and Research) under project RoboCare (A Multi-Agent System with Intelligent Fixed and Mobile Robotic Components). The Authors are part of the Planning and Scheduling Team [PST] at ISTC-CNR and would like to thank the other members of the team for their continuous support.

References

1. Bäckström, C. Computational Aspects of Reordering Plans. *Journal of Artificial Intelligence Research 9* (1998), 99–137.

[3] Assuming the two aspects of the problem are distinguishable in the benchmark domains.

2. Bartusch, M., Mohring, R. H., and Radermacher, F. J. Scheduling Project Networks with Resource Constraints and Time Windows. *Annals of Operations Research 16* (1988), 201–240.
3. Blum, A., and Furst, M. Fast Planning Through Planning Graph Analysis. *Artificial Intelligence* (1997), 281–300.
4. Bonet, B., and Geffner, H. Planning as heuristic search. *Artificial Intelligence 129*, 1–2 (2001), 5–33.
5. Brucker, P., Drexl, A., Mohring, R., Neumann, K., and Pesch, E. Resource-Constrained Project Scheduling: Notation, Classification, Models, and Methods. *European Journal of Operations Research* (1998).
6. Cesta, A., Oddi, A., and Smith, S. A Constrained-Based Method for Project Scheduling with Time Windows. *Journal of Heuristics 8*, 1 (2002), 109–135.
7. Cesta, A., Oddi, A., and Susi, A. O-Oscar: A Flexible Object-Oriented Architecture for Schedule Management in Space Applications. In *Proceedings of the Fifth International Symposium on Artificial Intelligence, Robotics and Automation in Space (i-SAIRAS-99)* (1999).
8. Currie, K., and Tate, A. O-Plan: The Open Planning Architecture. *Artificial Intelligence 52*, 1 (1991), 49–86.
9. 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).
10. Hoffmann, J., and Nebel, B. The FF Planning System: Fast Plan Generation Through Heuristic Search, 2001.
11. Jonsson, A., Morris, P., Muscettola, N., Rajan, K., and Smith, B. Planning in Interplanetary Space: Theory and Practice. In *Proceedings of the Fifth Int. Conf. on Artificial Intelligence Planning and Scheduling (AIPS-00)* (2000).
12. Kambhampati, S., Parker, E., and Lambrecht, E. Understanding and Extending Graphplan. In *Proceedings of ECP '97* (1997), pp. 260–272.
13. Kautz, H., and Selman, B. Unifying SAT-Based and Graph-Based Planning. In *Workshop on Logic-Based Artificial Intelligence, Washington, DC, June 14–16, 1999* (College Park, Maryland, 1999), J. Minker, Ed., Computer Science Department, University of Maryland.
14. Koehler, J., Nebel, B., Hoffmann, J., and Dimopoulos, Y. Extending Planning Graphs to an ADL Subset. In *Proceedings of ECP '97* (1997), pp. 273–285.
15. McAllester, D., Selman, B., and Kautz, H. Evidence for Invariants in Local Search. In *Proceedings of the Fourteenth National Conference on Artificial Intelligence (AAAI'97)* (Providence, Rhode Island, 1997), pp. 321–326.
16. Moskewicz, M., Madigan, C., Zhao, Y., Zhang, L., and Malik, S. Chaff: Engineering an Efficient SAT Solver. In *Proceedings of the 38th Design Automation Conference (DAC'01)* (2001).
17. Muscettola, N., Smith, S., Cesta, A., and D'Aloisi, D. Coordinating Space Telescope Operations in an Integrated Planning and Scheduling Architecture. In *IEEE Control Systems, Vol.12, N.1* (1992), pp. 28–37.
18. Pearl, J. *Heuristics: Intelligent Search Strategies for Computer Problem Solving.* Addison-Wesley Longman Publishing Co., Inc., 1984.
19. Pecora, F., and Cesta, A. Planning and Scheduling Ingredients for a Multi-Agent System. In *Proceedings of UK PLANSIG02 Workshop, Delft, The Netherlands* (2002).
20. R-Moreno, M., Oddi, A., Borrajo, D., Cesta, A., and Meziat, D. Integrating Hybrid Reasoners for Planning & Scheduling. In *Proceedings of UK PLANSIG02 Workshop, Delft, The Netherlands* (2002).