A Call for Knowledge-based Planning

David E. Wilkins and Marie desJardins
Artificial Intelligence Center, SRI International
333 Ravenswood Ave., Menlo Park, CA 94025
{wilkins, marie}@ai.sri.com

Abstract

Much of the effort of the planning community is currently focused on improving the performance of disjunctive planners (DPs). We are interested in solving real-world planning problems and, to that end, argue for the use of domain knowledge in planning. Hierarchical task network (HTN) planners use more knowledge than DPs and have advantages such as scalability, expressiveness, continuous plan modification during execution, and the ability to interact with humans. We argue, however, that the field must develop methods capable of using even richer knowledge models than those used in HTNs (and therefore DPs) in order to make planning tools useful for complex problems. While we applaud the development of faster DP systems and their use for planning subproblems, it may not be best for the field to focus so many resources on techniques that solve only a narrow subset of the problems that are faced in real-world domains.

Introduction

Much of the effort of the planning community is currently focused on improving the performance of disjunctive planners (DPs). Kambhampati (1997) defines disjunctive planners as planners that retain the current planset without splitting its components into different search branches. This family of planners includes Graphplan (Blum & Furst 1995), SATPLAN (Kautz & Selman 1996), and their derivatives, all of which use STRIPS-style planning knowledge to derive and solve propositional representations of planning problems.

Most planners described in AAAI-99 and IJCAI-99 are DPs that could solve only simple, small toy problems from the standard corpus used for testing disjunctive planners. (By contrast, the planning applications in IAAI-99 used more expressive representations and methods.) The 1998 AIPS planning competition also focused considerable effort on simple STRIPS-encoded toy problems that were solvable (or nearly so) by DPs. Unfortunately, the difficulties of defining a common notation for more complex problems and knowledge led to the demise of a proposed HTN track. A recent survey of advances in AI planning (Weld 1999) focused entirely on DP methods, reflecting the current emphasis on this style of planning.

We are interested in solving real-world planning problems, and believe that doing so will require techniques that are more expressive and provide a wider range of capabilities than DPs. In particular, we argue for the use of *knowledge-based planning* (KBP)—that is, methods that use available domain knowledge to solve the planning problem, including interacting with humans when necessary to make use of their expertise. A planner may be more or less knowledge-based, depending on the range of knowledge it uses, and how effectively it uses it. We argue that KBP can solve problems that DPs in their current form cannot, because of the greater expressivity and more natural representations of KBP.[1]

We present some characteristics of real-world palnning problems that are not solvable by current DP approaches, and we argue that DPs are unlikely to extend to these problems. We summarize the features of an HTN planner to show the types of knowledge and capabilities that are exploited in an existing KBP system. However, current HTN planners are only a small step in the direction of the level of KBP that we envision. We next argue that achieving planful behavior in a complex, dynamic world will require the use of much more knowledge and richer knowledge models than those used in HTN planners. We discuss forms of knowledge that HTN planners do not use, and give some examples of problems that today's HTN planners are not able to solve. Finally, we draw some lessons from the history of the machine learning research community that are analogous to the current trends in the planning community.

Characteristics of Real-World Planning Problems

Real-world problems have been found by many researchers to require more expressive representations and capabilities than those provided by current AI planning systems. Chien *et al.* (1996) conclude from their experience with multiple NASA applications that "current plan representations are

[1]Our comments apply not only to DPs, but to any planner that cannot address hard, embedded problems, whether for computational reasons or a lack of expressiveness. We address our remarks to DPs as they are currently the most studied systems with that property.

impoverished." They discuss the requirements of an operational context in which users must interact with the system, and must be able to understand and modify the plans produced by the planner. Our experience with military planning applications supports these conclusions. In this section, we describe some of the specific capabilities that are needed to solve real-world problems: numerical reasoning, parallelism, context-dependent effects, interaction with users, execution monitoring, replanning, and scalability.

Reasoning with numbers is essential in every realistic domain that we have studied. Common needs for numbers are time, sharable resources having a specific capacity, continuous resources available in limited quantities, and goals of accumulation. In practice, DPs have difficulty handling problems involving reasoning about numbers.[2] In most existing non-HTN AI planners, the need for numerical reasoning is reduced by assuming that sharable resources have infinite capacity, and that continuous resources are unlimited (Srivastava & Kambhampati 1999).

Realistic domains may have dozens of (perhaps necessarily) parallel activities, as activities of various agents are coordinated. Parallelism can cause computational problems for DPs, and many systems produce only sequential plans.

Realistic domains often have numerous context-dependent effects, which can cause an exponential explosion in the number of STRIPS operators needed. This problem is being addressed to some extent in DPs. Extensions to the Graphplan algorithm to handle conditional effects are given in (Kambhampati, Parker, & Lambrecht 1997) and (Guéré & Alami 1999), but neither paper discusses the time or space complexity of the algorithms. Other approaches have also been tried, perhaps the most promising being factored expansion, in which an action with conditional effects is split into new actions called "components," one for each conditional effect. The cost is added complexity in the planning algorithms involving "tricky" extensions (Anderson, Smith, & Weld 1998).

Interacting with people is a critical aspect of real-world planning. Realistic problems are embedded in the world, and generally do not have precisely defined boundaries or evaluation functions. Thus, most interesting planning problems will be difficult or impossible to model fully. For example, criteria for plan evaluation often cannot be quantified, such as when the political consequences of a military or media action are crucial. In such cases, a human user must be able to guide the planner and evaluate the plans produced, allowing the planning system to take advantage of the user's expertise

In the real world, the goal of planning is not simply to build the plan, but to use it to control actions in the world. Therefore, realistic planning systems must support execution monitoring and continuous plan modification during execution. Because there is no dependency structure in DP plans, monitoring them is difficult. In addition, the DP approach is very brittle in the face of changing problem requirements, and any change in the environment may result in the planning system having to start from scratch.

Finally, realistic problems involve enormous search spaces, so scalability is essential. Vast strides have been made in the size of problems solved by DPs, such as solving instances with 10^{16} to 10^{19} configurations. However, these large DP problems are still representations of toy problems, such as a logistics problem with 9 packages, 5 trucks, 2 airplanes, and 15 locations (Kautz & Selman 1998). Simply increasing the number of locations to a realistic number will make even these toy problems unsolvable. In contrast, HTN planners can generate plans in domains with thousands of objects and hundreds or thousands of actions.

Using Knowledge in HTN Planners

Hierarchical task network (HTN) planning is one example of a KBP approach that is more knowledge-based than DPs. It has been known for some time that HTN formalisms are more expressive than the STRIPS formalism favored by many DPs, roughly analogous to the additional expressivity of context-free grammars over right-linear (regular) grammars (Erol, Hendler, & Nau 1994).

HTNs naturally model the world in the same way that human users do in many domains, using the same abstractions. This modeling approach enables users to control and understand the planning process and the resulting plans. (Note, however, that current HTN planners still leave much to be desired in terms of interactive planning.) In contrast, the DP approach models the planning problem as millions of conjunctive normal form expressions, making it difficult for users even to understand the planner's reasoning process, much less intervene to modify or guide it.

In this section, we describe some of the uses made of domain knowledge in a particular HTN planner, SIPE–2. These features may be candidates for extending non-HTN planners. Kautz and Selman (1998) identify three kinds of planning knowledge: knowledge about the domain, knowledge about good plans, and explicit search-control knowledge. In addition, HTN planners are also concerned with other types of knowledge, such as knowledge about interacting with the user, knowledge about a user's preferences, and knowledge about plan repair during execution. SIPE–2 has formalisms that allow encoding of these types of knowledge, in either HTN operators (also known as methods or schemas), advice, or other declarations.

It is sometimes argued that the knowledge used by HTN planners is "simply search-control knowledge," rather than part of the problem statement. However, if the goal is to solve realistic planning problems, then intelligent, principled search control that takes advantage of knowledge about the domain is precisely what is needed. This knowledge can often be naturally and efficiently captured in HTN operators, where much of the context is implicit and therefore need not be expressed or checked during each attempted application.

[2]While simple, finite arithmetic could be added to DPs, the combinatorics would generally explode. Another approach is given by Wolfman and Weld (1999), who describe a system that combines SATPLAN with an incremental Simplex algorithm for solving linear inequalities. A useful extension, but combinatorics allow solution of only toy problems.

HTN Capabilities: SIPE-2

SIPE–2 (Wilkins 1990) is a domain-independent HTN planner that uses more knowledge and has richer capabilities (such as numerical constraints and resource models) than the current DPs. Example applications include containing oil spills (Agosta & Wilkins 1996), planning air campaigns for the Air Force (Wilkins & Myers 1998; Lee & Wilkins 1996), and joint military operations planning (Wilkins & Desimone 1994). In the latter applications, the domain knowledge includes 100 to 200 operators, around 500 objects with 15 to 20 properties per object, and a few thousand initial predicate instances. Plans can include up to several hundred actions—several thousand if all abstraction levels are counted—usually having numerous parallel activities.

Expressiveness SIPE–2 provides a powerful approach to representing and reasoning about planning problems that makes it suitable for complex domains.

SIPE–2's HTN operators are encoded at multiple *abstraction levels*. The higher levels can model various solution methods. SIPE–2 operators can *dynamically generate a set of goals* at planning time, a capability that has been extensively used. For example, a defend goal can be generated for every currently known threat. *Situation-dependent effects* of actions are deduced by a causal theory. Such effects have proven their use in practice — without them, the number of operators can grow exponentially in complex domains.

SIPE–2 can *reason about numbers*, a capability crucial in nearly all of our applications. For example, a planning variable may be constrained to be a runway with a length greater than some number, sharable resources have a specific capacity, and continuous resources are available in limited quantities. In many application domains, it is necessary to *accumulate a certain quantity* of some resource, or achieve a certain level of effect, such as obtaining a sufficient length of boom to surround an oil spill. Such goals are not accomplished by a single action; rather, several actions contribute to the accumulation. Thus, SIPE–2 determines when a set of actions (that individually produce some amount of the resource in question) together achieve an accumulation goal.

Temporal reasoning is important in many domains. SIPE–2 has two different modes for reasoning about time. The most general allows specification of any of the thirteen Allen relations between any two nodes. The temporal constraints are solved separately from the other constraints by passing them to Tachyon (Arthur & Stillman 1992).

Finally, calls can be made to *arbitrary domain-specific* LISP *code*, for knowledge that cannot easily be modeled in the HTN formalism, and for sophisticated numerical calculations. Functions on planning variables may compute an instantiation (e.g., the duration of a flight) and procedural attachment on predicates may compute whether a condition is true.

User Interaction Because HTN plans and domain knowledge can be complex, a powerful graphical user interface (GUI) is essential. Without natural pictorial representations of the knowledge and plans, it would be nearly impossible for a human to understand them. SIPE–2 provides a GUI and a web server to aid in generating plans, viewing complex plans as graphs, and following and controlling the planning process.

SIPE–2 provides a flexible and powerful interactive planner. In our applications, this feature is critical because experienced human planners can guide the search effectively, and are reluctant to give control to an automated system. The user may interact with the planning process at many different levels of detail, and may direct the planner to solve certain parts of the problem automatically. Under interactive control, the user can control (among other things) when and how resources are allocated, which operators to select, which goal to expand next, how to instantiate planning variables, and how to resolve conflicts.

The user can also control or influence the plan development process using the Advisable Planner (Myers 1996). Users can direct the planning process by providing high-level guidance that influences the nature of the solutions generated. Advice consists of task-specific properties of both the desired solution and the problem-solving process to be used for a particular problem.

Constraints and Efficiency A sort hierarchy represents invariant object properties, describes the classes to which an object belongs, and allows for inheritance of properties. The sort hierarchy encodes large amounts of knowledge in our applications, and the planner can reason more efficiently about this knowledge because it knows the relationships cannot change as actions are performed.

SIPE–2 uses the least-commitment approach to variable binding. Constraints are placed on variables by domain knowledge in the operators (e.g., a particular truck must have a capacity greater than 100). Instantiations are not chosen until sufficient constraints accumulate to identify a unique acceptable value. Because uninstantiated variables increase computational complexity, domain-specific knowledge can be used to specify when early instantiation of variables can be done without adversely affecting solution quality (Myers & Wilkins 1998).

Predicates can be declared as functional in certain arguments, allowing a dramatic speedup, which has been documented experimentally (Myers & Wilkins 1998). Functional predicates are of particular importance to reasoning about locations in planning systems, and have proven valuable in nearly every application of SIPE–2.

The system relies on *plan critics* that check for and correct conflicts in plans, reducing computational costs during plan expansion. Examples of plan critics include the temporal reasoner and checking constraints on planning variables. These critics are applied by default after each *planning level*, i.e., an expansion of the whole plan to a greater level of detail, but the frequency can be increased or decreased as appropriate for the problem domain (Wilkins 1990).

Knowledge Beyond HTN

Despite the power of HTN planning systems, and their demonstrated ability to address real-world planning problems, they have limitations that make them inadequate for many problems of interest. In particular, HTN planners:

- require complete (except for anticipated incompleteness) and certain knowledge about the world
- model the effects of actions as deterministic, fully understood outcomes
- assume that the planner controls all agents that cause changes in the world state
- require significant effort in domain modeling and knowledge acquisition for complex problems
- cannot perform or incorporate complex or decision-theoretic evaluations of plan quality
- ignore the qualification problem
- use simplistic frame problem solutions that prevent drawing the most appropriate conclusions when contradictory (perceptual) information arrives (Pollock 1998)
- do not consider risks and utilities
- do not use knowledge and probabilities to handle uncertainty
- are brittle (may not work if the problem changes slightly).

These limitations are shared by the other major families of planners (DPs and causal-link planners), although some of them, such as handling uncertainty, are the subject of ongoing research (Boutilier, Dean, & Hanks 1999; Majercik & Littman 1998; Smith & Weld 1998; Weld, Anderson, & Smith 1999; Kushmerik, Hanks, & Weld 1994).

In addition to these limitations, planning systems that could solve interesting problems in a complex, dynamic world will need capabilities that represent a fundamental shift in how we think about planning problems. An ideal system would be able to behave like humans do in these sort of environments; in particular, it would have to:

- exhibit creativity, devising new actions that can solve a problem or shorten a plan
- use analogy to transfer solutions from other problems
- effectively interact with humans to use their knowledge in decisions
- behave intelligently in the face of conflicting or incomplete information.

We believe that these capabilities will require more knowledge, including background knowledge of other domains and of how the world works. Evaluation criteria other than correctness and plan length will have to be factored in explicitly. Also, interacting effectively with humans will be essential because we will never model every possible issue that might affect a planning decision.

Of course, not all interesting problems have these characteristics, and in any given case, it may be possible to formulate the problem in such a way as to remove the need for these capabilities. For example, in developing the Burton planner (not a DP planner), Williams and Nayak (1996) used a purely propositional representation. However, it seems unlikely that all interesting problems will be amenable to such an approach, and other NASA applications have required richer representations (Chien *et al.* 1996).

Knowledge in DP vs KBP

To impact realistic problems, we predict that DPs will have to incorporate the types of knowledge used by HTN planners, as well as knowledge to overcome the limitations of HTN approaches that we have discussed. It is encouraging that this is already starting to occur. For example, there is initial work on adding knowledge about temporal extent of actions to SATPLAN encodings (Smith & Weld 1999), and on encoding HTN method knowledge for satisfiability solvers (Mali & Kambhampati 1998).

However, while HTN planners can generally make effective use of additional knowledge, the same is not necessarily true of DPs. Additional knowledge encoded as axioms may increase the size of the problem with redundant axioms, and make the problem harder to solve. Initial experiments indicate that whether added knowledge helps or hurts may depend on the particular combination of knowledge, problem, and algorithm (Kautz & Selman 1998). For example, the "point of diminishing returns from the addition of axioms would be sooner reached for stochastic search than for systematic search" (Kautz & Selman 1998). Thus, the knowledge added to a DP may have to be carefully chosen for the problem being solved and the algorithm being used.

KBP approaches are rightly criticized for the expense of modeling a new domain. However, we conjecture that building computationally efficient encodings for DPs of complex planning domains is no easier than building HTN models. Many encoding issues are still under study, even for toy domains (Kautz & Selman 1999; Brafman 1999; Mali & Kambhampati 1998).

Lessons from Machine Learning

In every research community, there is an ongoing tension between well defined and more ambitious problems. On the one hand, if a field focuses on small, well understood problems, with well defined algorithmic properties and evaluation metrics, then a set of benchmark problems can be formulated to facilitate formal and empirical analysis and comparison of competing methods. On the other hand, many of the interesting challenges posed by realistic applications have broader implications and less well understood properties, and the problems are more difficult to define crisply and to evaluate.

Several years back, the machine learning community established a repository of benchmark problems to evaluate machine learning systems. Naturally, these problems all had commonalities: most used an attribute-vector representation; most consisted of "sets of instances" with no background knowledge. In practice, they could be used only to evaluate predictive accuracy on propositional, supervised learning algorithms. Despite these limitations, however, it became the de facto standard that papers submitted to the International Conference on Machine Learning (ICML) had to include an evaluation on these benchmark problems.

In some ways, this set of benchmarks, and the emphasis on evaluation, were good for the community: they forced researchers to think about metrics and about comparing their systems to other systems, and they provided a baseline of performance that researchers could test new ideas against. On the other hand, they tended to stifle research that didn't fit neatly into the problem space defined by the benchmark problems. Applications-oriented researchers reported that it was difficult or impossible to get their papers accepted to the leading ML conferences (Provost & Kohavi 1998). Meanwhile, more and more papers appeared showing minor tweaks and incremental improvements to existing algorithms (but they showed "statistically significant" improvements on the benchmark algorithms!)

As a result, there are now many well understood and effective methods for propositional supervised learning—and there has been much less progress in other areas of machine learning, such as incorporating background knowledge, feature engineering, relational learning, interactive learning techniques, visualization of learned knowledge, and complex evaluation criteria.

Most recently, there has been an explosion of interest in learning Bayes nets. Bayes net learning and inference techniques have appealing computational properties that are analogous to those of DP approaches: they efficiently capture certain types of problem structure, and significantly speed up certain types of inference over previous methods. However, like DP approaches, they use a propositional representation, and do not address many of the other challenges posed by realistic learning problems. As with DP approaches, the rush of enthusiasm over Bayes net techniques has threatened to overshadow the fact that despite their computationally attractive properties, they still solve only a small subproblem within the overall field of machine learning.

Similarly, in the planning community, there is a danger that by focusing too much attention and effort on DP methods and the problems they solve, we risk losing the ability to recognize other kinds of contributions and advances. In particular, if the benchmark of performance becomes solely how many blocks our planners can stack, and how fast they can do it, then it will become increasingly difficult to recognize and learn from research that performs well along other dimensions—or that addresses problems that DP systems overlook completely. As we discussed earlier, DP researchers within the planning community are starting to look towards extending their systems to incorporate richer forms of knowledge. This is a trend that we applaud, and that we hope will continue, but it is not enough to simply broaden the uses of DP systems: we need to be open to completely different approaches and paradigms as well.

While improving the speed of solving problems we know how to formulate precisely is a valuable research activity, so is continuing to investigate problems that we don't yet have a good handle on formulating or solving. Results may be more difficult to achieve or quantify for these the "hard problems," but that doesn't mean we shouldn't be working on them.

Conclusion

Ginsberg (1996) has pointed out that the SATPLAN approach is successful because it solves the "puzzle" part of a problem, and overlooks any commonsense reasoning aspects of the true problem. In Ginsberg's view, commonsense reasoning is the heuristic process by which we reduce extremely complex problems to NP-hard or simpler problems for which search is feasible. Which aspects of a problem to pay attention to, frame and context assumptions, and default strategies for organizing complex activities are all aspects of commonsense reasoning. As Ginsberg puts it, "It is Kautz and Selman who are solving the commonsense aspects of the problem; their 'planner' is solving the puzzle-mode kernel of the problem instead of the problem itself." Indeed, the problems solved by DP approaches are almost exclusively puzzle-style problems (or "real-world" problems that have been reformulated as puzzles).

We favor using DP to solve puzzle-style subproblems that can be represented as satisfiability problems and solved in acceptable time. DP may well be the appropriate method for those aspects of the overall problem. However, AI planners also need to provide support for the commonsense reasoning aspects of the problem so that plans can be used to guide planful behavior while embedded in a complex, dynamic environment. We have argued that incorporating knowledge into the planning process is the most promising way to provide these abilities.

Although DP methods are clearly useful approaches for solving certain subproblems, it is important for the field as a whole to continue to look at a wider range of problems. There is a danger of allowing the current popularity of DP approaches, and the associated evaluation techniques and "puzzle-style" problem suite, to overly influence the field, making it more difficult for advanced KBP methods to find an audience.

Acknowledgments This research was supported by Contract F30602-95-C-0235 with the Defense Advanced Research Projects Agency, under the supervision of Air Force Research Lab – Rome. Thanks to Dana Nau and Foster Provost for contributing their ideas to this paper.

References

Agosta, J. M., and Wilkins, D. E. 1996. Using SIPE-2 to plan emergency response to marine oil spills. *IEEE Expert* 11(6):6–8.

Anderson, C.; Smith, D. E.; and Weld, D. 1998. Conditional effects in Graphplan. In *Proc. of the 1998 International Conference on AI Planning Systems*, 44–53.

Arthur, R., and Stillman, J. 1992. Tachyon: A model and environment for temporal reasoning. Technical report, GE Corporate Research and Development Center.

Blum, A., and Furst, M. 1995. Fast planning through planning graph analysis. In *Proceedings of the 14th International Joint Conference on Artificial Intelligence*, 1636–1642. Morgan Kaufmann.

Boutilier, C.; Dean, T.; and Hanks, S. 1999. Decision-theoretic planning: Structural assumptions and computational leverage. *Journal of Artificial Intelligence Research* 11:1–94.

Brafman, R. I. 1999. Reachability, relevance, resolution and the planning as satisfiability approach. In *Proc. of the 1999 International Joint Conference on Artificial Intelligence*, 976–981.

Chien, S.; Hill-Jr, R.; Wang, X.; Estlin, T.; Fayyad, K.; and Mortensen, H. 1996. Why real-world planning is difficult: A tale of two applications. In Ghallab, M., and Milani, A., eds., *Advances in AI Planning*. IOS Press. 287–298.

Erol, K.; Hendler, J.; and Nau, D. S. 1994. HTN planning: Complexity and expressivity. In *Proc. of the 1994 National Conference on Artificial Intelligence*, 1123–1128.

Ginsberg, M. L. 1996. Do computers need common sense? In *Conference on Knowledge Representation and Reasoning*.

Guéré, E., and Alami, R. 1999. A possibilistic planner that deals with non-determinism and contingency. In *Proc. of the 1999 International Joint Conference on Artificial Intelligence*, 996–1001.

Kambhampati, S.; Parker, E.; and Lambrecht, E. 1997. Understanding and extending Graphplan. In *European Conference on Planning*.

Kambhampati, S. 1997. Challenges in bridging plan synthesis paradigms. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*.

Kautz, H., and Selman, B. 1996. Planning as satisfiability. In *Proceedings of the 10th European Conference on Artificial Intelligence*, 359–363. Wiley.

Kautz, H., and Selman, B. 1998. The role of domain-specific knowledge in the planning as satisfiability approach. In *Proc. of the 1998 International Conference on AI Planning Systems*, 181–189.

Kautz, H., and Selman, B. 1999. Unifying SAT-based and graph-based planning. In *Proc. of the 1999 International Joint Conference on Artificial Intelligence*, 318–325.

Kushmerik, N.; Hanks, S.; and Weld, D. 1994. An algorithm for probabilistic least-commitment planning. In *Proc. of AAAI-94*, 1073–1078.

Lee, T. J., and Wilkins, D. E. 1996. Using SIPE-2 to integrate planning for military air campaigns. *IEEE Expert* 11(6):11–12.

Majercik, S., and Littman, M. 1998. Maxplan: A new approach to probabilistic planning. In *Proc. of the 1998 International Conference on AI Planning Systems*, 86–93.

Mali, A. D., and Kambhampati, S. 1998. Encoding HTN planning in propositional logic. In *Proc. of the 1998 International Conference on AI Planning Systems*, 190–198.

Myers, K. L., and Wilkins, D. E. 1998. Reasoning about locations in theory and practice. *Computational Intelligence* 14(2):151–187.

Myers, K. L. 1996. Strategic advice for hierarchical planners. In Aiello, L. C.; Doyle, J.; and Shapiro, S. C., eds., *Principles of Knowledge Representation and Reasoning: Proceedings of the Fifth International Conference (KR '96)*. Morgan Kaufmann Publishers.

Pollock, J. L. 1998. Perceiving and reasoning about a changing world. *Computational Intelligence* 14(4):498–562.

Provost, F., and Kohavi, R. 1998. Guest editors' introduction: On applied research in machine learning. *Machine Learning* 30(2/3).

Smith, D., and Weld, D. 1998. Conformant Graphplan. In *Proceedings of the 15th National Conference on Artificial Intelligence*, 889–896. AAAI Press.

Smith, D. E., and Weld, D. S. 1999. Temporal planning with mutual exclusion reasoning. In *Proc. of the 1999 International Joint Conference on Artificial Intelligence*, 326–333.

Srivastava, B., and Kambhampati, S. 1999. Scaling up planning by teasing out resource scheduling. In *ECP-99*.

Weld, D. S.; Anderson, C. R.; and Smith, D. E. 1999. Extending Graphplan to handle uncertainty and sensing actions. In *Proceedings of the 15th National Conference on Artificial Intelligence*, 897–904. AAAI Press.

Weld, D. S. 1999. Recent advances in AI planning. *AI Magazine* 93–123.

Wilkins, D. E., and Desimone, R. V. 1994. Applying an AI planner to military operations planning. In Fox, M., and Zweben, M., eds., *Intelligent Scheduling*. Morgan Kaufmann Publishers Inc., San Francisco, CA. 685–709.

Wilkins, D. E., and Myers, K. L. 1998. A multiagent planning architecture. In *Proc. of the 1998 International Conference on AI Planning Systems*, 154–162.

Wilkins, D. E. 1990. Can AI planners solve practical problems? *Computational Intelligence* 6(4):232–246.

Williams, B. C., and Nayak, P. P. 1996. A model-based approach to reactive self-configuring systems. In *Proceedings of the 13th National Conference on Artificial Intelligence*. AAAI Press.

Wolfman, S. A., and Weld, D. S. 1999. The LPSAT engine and its application to resource planning. In *Proc. of the 1999 International Joint Conference on Artificial Intelligence*, 310–316.