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@ai.sri.com, marie@ai.sri.com

To appear in *AI Magazine*, Spring 2001, volume 22, number 1.

Abstract

We are interested in solving real-world planning problems and, to that end, argue for the use of domain knowledge in planning. We believe that the field must develop methods capable of using rich knowledge models in order to make planning tools useful for complex problems. We discuss the suitability of current planning paradigms for solving these problems. In particular, we compare knowledge-rich approaches such as hierarchical task network (HTN) planning to minimal-knowledge methods such as STRIPS-based planners and disjunctive planners (DPs). We argue that the former methods have advantages such as scalability, expressiveness, continuous plan modification during execution, and the ability to interact with humans. However, these planners also have limitations, such as requiring complete domain models and failing to model uncertainty, that often make them inadequate for real-world problems.

In this paper, we define the terms *knowledge-based* (KB) and *primitive-action* (PA) planning, and argue for the use of KB planning as a paradigm for solving real-world problems. We next summarize some of the characteristics of real-world problems that we are interested in addressing. Several current real-world planning applications are described, focusing on the ways in which knowledge is brought to bear on the planning problem. We describe some existing KB approaches, and then discuss additional capabilities, beyond those available in existing systems, that are needed. Finally, we draw an analogy from the current focus of the planning community on disjunctive planners to the experiences of the machine learning community over the past decade.

Keywords: Planning, execution monitoring, knowledge-based planning, hierarchical task network

Introduction

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 current planning systems. Real-world problems have been found to require more expressive representations and capabilities than are needed for the "standard" set of benchmark planning problems (blocks world, Towers of Hanoi, simplified logistics, and the like) or for the problems used in the 1998 and 2000 Artificial Intelligence Planning and Scheduling Conference planning competitions (McDermott 2000; Long 2000; Bacchus *et al.* 2000).

Past research in AI planning can be roughly divided into two camps: systems that take a minimalist approach to domain knowledge, and systems that focus on leveraging as much domain knowledge as possible. Techniques in the first set generally restrict themselves to domain models consisting of STRIPS-style descriptions of primitive actions. We refer to these methods in the first set as *primitive-action* (PA) planning techniques, since they construct plans from descriptions of the actions that can appear in the final plan.[1] A currently popular form of primitive-action planning is *disjunctive planning*, which uses primitive action descriptions to encode and solve propositional representations of planning problems. (A recent survey of current directions in AI planning (Weld 1999) focuses almost exclusively on this style of planning.)

Techniques in the second set are based on a philosophy of using whatever domain knowledge is available to solve the planning problem. These systems are characterized by the use of multiple types of domain knowledge and complex domain models to support their reasoning processes. This knowledge may include task and goal structures, additional constraints, search control techniques, and interacting with humans when necessary to make use of their expertise. We refer to these techniques as *knowledge-based* (KB) planning methods.

In this paper, we argue that in order to scale up to complex problems, multiple types of knowledge must be explicitly encoded in understandable structures, and that planning algorithms must be able to use this explicit knowledge ef-

[1] The term "domain-independent planning" has sometimes been used in the literature to describe these systems, but this term is a misnomer, since the action descriptions do in fact constitute a form of domain knowledge. In this paper, we use the term "domain-independent" to refer to any planner that is designed to be generically applicable to any domain encoded in the proper form. Using this definition, KB planning methods that operate on explicit domain descriptions—as opposed to systems that are hardwired for particular domains—are as "domain-independent" as PA planning systems.

fectively.

One possible approach to achieving this knowledge encoding would be to augment primitive-action planners with additional knowledge. Indeed, researchers have begun to investigate how these systems can be augmented with domain-specific, hand-encoded control rules (see sidebar). However, only certain types of knowledge can be captured in these rules, and they are difficult and time-consuming to construct. The result is that this approach to encoding knowledge is not scalable to large, complex problems. It seems unlikely that hand-coded control rules will be a sufficient approach to knowledge modeling for the types of problems that arise in the real world.

KB planning methods have limitations as well. On the one hand, current KB methods are not knowledge-based enough: as we will discuss later in the paper, they do not incorporate many types of knowledge that are important for real-world problems. On the other hand, they are *too* knowledge-based for some problems: the inference entailed by modeling many aspects of the planning problem can be computationally expensive, with the result that KB methods may not be the most efficient approach to solving certain types of constraint satisfaction subproblems within a larger planning problem. Therefore, we believe that solving large, complex problems will require both the development of new KB methods and the integration of primitive-action methods with these new techniques.

KB planning and primitive-action planning represent two ends of a continuum: a planner may be more or less knowledge-based, depending on the range of knowledge it uses, and how effectively it uses it. Integrated systems may ultimately provide the best of both worlds. However, we argue that KB methods can solve problems that primitive-action methods cannot, because of the greater expressivity and more natural representations of KB planning.

The remainder of the paper is organized as follows. We first describe some common characteristics of real-world planning problems that are not solvable by current primitive-action methods, and we argue that these methods are unlikely to extend to these problems. We describe some real-world planning problems that have been addressed by current KB planners. We then discuss how multiple types of knowledge and capabilities are exploited in existing KB planners. However, current KB techniques represent only a small step in the direction of the level of KB planning that we envision. We next argue that achieving goal-directed behavior in a complex, dynamic world will require reasoning about the consequences of future actions. This is turn will entail the use of much more knowledge and richer knowledge models than those used in today's KB planners. We discuss forms of knowledge that current KB planners do not use, and give some examples of problems that today's 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 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 and oil spill planning applications supports these conclusions. Here we describe some of the specific capabilities that are needed to solve real-world problems: numerical reasoning, concurrent actions, 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. An example of the latter is the goal of obtaining a certain quantity of resource that must be assembled from smaller aggregations, such as getting enough boom from several warehouses to contain an oil spill, or enough soldiers or equipment for a military operation. In practice, disjunctive planners (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 disjunctive planners, and some 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 disjunctive planners. Extensions to the Graphplan algorithm to handle conditional effects are given in (Kambhampati, Parker, & Lambrecht 1997) and (Guéré & Alami 1999), but applications of Graphplan are already limited by computational efficiency and 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. This approach appears to outperform

[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 the solution of only toy problems.

a straightforward splitting of each action into (a possibly exponential number of) STRIPS operators, at least on large problems. 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. It is also hard to specify when a situation warrants breaking rules or ignoring certain information, yet such situations are common in real life. 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. Figure 1 shows a possible architecture for a system incorporating both planning and execution. The intention is not to promote this architecture, but to show some of the added complication introduced by executing a plan. The planner and executor have specialized knowledge bases to support their respective roles, and must also share common ontologies, models of actions, and knowledge about the world. The executor requires a rich model of the relationships among tasks in the plan and of possible outcomes and contingencies. The executor generally operates at a faster tempo than the planner and attempts to provide effective responses to changing conditions, which requires rapid replanning and possibly responding to time-critical situations without invoking the planner (perhaps using procedures from the monitoring knowledge).

The Plan Initializer/Synchronizer in Figure 1 prepares plans for execution. This module may generate plan-specific monitors to efficiently monitor conditions over certain intervals. It must also synchronize the transfer of control from the original plan to a newly modified version of the plan, even though the original plan has continued execution during the replanning process. In many domains, large quantities of information about the world state (for example, from sensor networks) are constantly arriving, but only small portions may affect the current plan. The Information Manager filters the incoming data, passing the relevant parts to the Execution Manager.

Because there is no dependency structure in DP plans, monitoring them is difficult. In addition, the disjunctive planning 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. Because KB planners record dependencies and goal structure, KB replanning techniques can often modify an executing plan in the face of new requirements (Myers 1999; Wilkins *et al.* 1995).

Finally, realistic problems involve enormous search spaces, so scalability is essential. Vast strides have been made in the size of problems solved by disjunctive planners, such as solving instances with 10^{16} to 10^{19} configurations. However, these large disjunctive planning 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 (Wilkins & Myers 1998; Wilkins & Desimone 1994).

Real-World Applications

We describe several existing planning applications that include many of the characteristics of hard real-world problems that we are interested in exploring. These applications use a variety of knowledge-based planning techniques, and represent a starting point for research into real-world planning methods. They also highlight the need for further work in many of the areas we have discussed. Another survey of real-world planning applications can be found in (Knoblock 1996).

Unlike toy problems, real-world problems generally cannot be completely modeled, particularly when plans are executed in the real world. Therefore, validating the correctness of a planner raises many challenging issues. The NASA domains described in this section have been fielded, and the planners have been subjected to extensive validation procedures. Generally, validation involves empirical tests for carefully selected test cases (against simulators, test harnesses, or the real world). Results may be checked automatically against a set of correctness requirements that have been carefully defined and validated by human experts. A description of such testing and the issues involved can be found in (Smith, Feather, & Muscettola 2000).

Oil-Spill Crisis Response Planning

Oil-spill incident response is a race against time, to contain or remove oil before it damages the shore. Planning begins by entering the specifics of a spill incident—location, time of day, spill rate, and so on—and then forecasting (using legacy systems) the spill trajectory, considering the uncertainty in its spreading caused by wind and waves. This forecast determines which environmentally sensitive shore sectors the oil will hit, and when.

The Spill Response Configuration System (SRCS) helps the U.S. Coast Guard (USCG) estimate the adequacy of the amounts and locations of cleanup equipment in its coastal oil-spill incident response plans (Agosta & Wilkins 1996). SRCS determines adequacy by building plans that meet a range of spill scenarios and then evaluating the plans. Previous approaches used approximate rules to estimate equipment needs. By automating the planning process, SRCS enables users to plan and evaluate a range of detailed responses to a range of spill scenarios, enabling the USCG to more accurately estimate its needs.

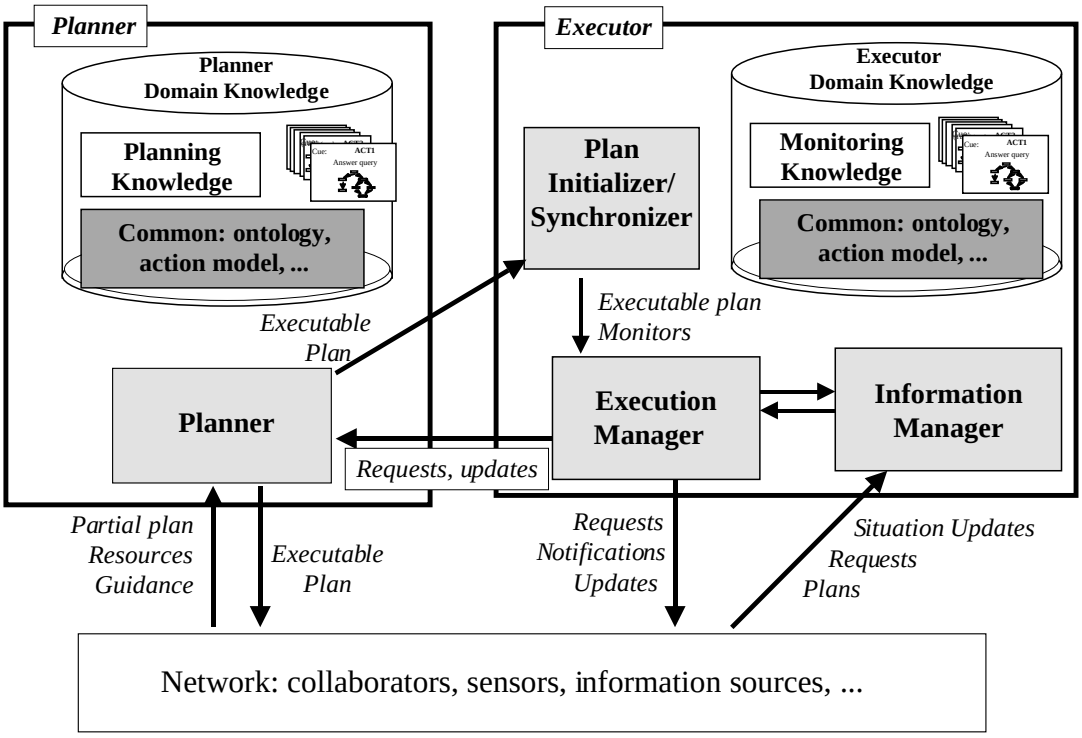

Figure 1: Possible architecture for a system that incorporates both planning and execution. The planner sends plans to the executor, which monitors the executing plan and sends requests and situation updates to the planner.

The planner works from the spill-trajectory forecast, together with geographic information, such as the sectors into which the region is divided and the USCG requirements for protection of these areas. In addition, the planner works with a database of the quantities and capabilities of available equipment and resources, which are often located over a large geographical area with varying time and transportation costs.

SRCS integrates simulation, evaluation, map display, and scheduling tools with SIPE–2. The planner, which uses a knowledge base of oil-spill response HTN operators, and the scheduler work interactively with the user to generate a plan consisting of equipment deployment and employment actions. The actions in this plan must satisfy constraints determined by the projected oil dispersal pattern, equipment cleanup capabilities and transport times, and environmental protection requirements. SRCS is intended to be used for configuration planning—advance planning to prepare for likely incidents—rather than for real-time planning as an incident unfolds.

This application requires extensive use of metric (numerical) goals, primarily resource and temporal reasoning. The temporal reasoning involves deadlines and concurrency. One important use of resource reasoning is the accumulation of a certain level of some resource at a certain place and time. An example is a goal to provide several thousand feet of oil-containment boom to protect a sensitive area. This goal must typically be met by transporting several shipments of boom from different locations around the state or country.

Most of the user's interaction with SRCS is mediated by the map interface, implemented in a commercial geographical information system. The user thus can immediately see both the extent of the spill and where resources are employed at various times.

Plans are evaluated on the degree to which they achieve the overall objective of cleaning up the spilled oil. In many spills, much of the oil will escape, no matter how much equipment is available, because of the difficulty of operations and speed of spread due to the weather. Furthermore, for any spill, SRCS can generate many possible plans, and users can partially or completely sacrifice a sector cleanup goal if they believe equipment that would have been assigned to a sector better serves the overall goals by being used elsewhere.

Because there are many feasible plans that vary widely in their degree of success, SRCS includes an evaluation model for finding good plans. The plan is used as an input to this model, along with the projected flows determined by the trajectory model. The evaluation model accounts for the quantities of oil contained and removed in each sector, for each period. From this accounting, it can calculate measures of plan merit, such as the final fraction of oil removed by the plan.

Space Applications

Several planning domain models have been developed by NASA researchers[3] for studying a range of space applications of AI planning systems. Three of these domains are described here; they highlight the need for planning systems that can represent and reason about complex activities, resources, and interactions.

DATA-CHASER DATA-CHASER flew aboard the space shuttle Discovery on mission STS-85 as a Hitchhiker payload with the International Extreme Ultraviolet Hitchhiker Bridge (IEH-2) in August 1997 (Figure 2). This mission used automated planning and scheduling techniques to decrease mission commanding effort by 80% while increasing science return (i.e., efficiency of instrument utilization) by 40% (as compared to manual sequence generation) (Chien *et al.* 1999).

DATA-CHASER consisted of three co-aligned instruments that collected data in the far and extreme ultraviolet wavelengths. These instruments obtained images of the sun that correlated solar activity with radiation flux, associating this flux with individual active regions of the sun. The irradiance data could be sent to the ground system using low-rate (available 90% of the time, at 1200 bps) or medium-rate (available when scheduled, at 200 kbps) transmission. The payload was capable of receiving commands sent from the ground system when uplink was available. The DATA module contained the science instruments themselves. The CHASER module (Figure 3) contained the planning and scheduling system that managed the shuttle resources in order to accomplish the mission successfully.

Shuttle resources are shared by multiple missions, and their availability is subject to change every 12 hours (the frequency at which NASA changes shuttle flight plans). These resources include access to uplink and downlink channels and time windows when the instruments are allowed to operate. In addition, DATA-CHASER had thermal constraints that limited the duration of payload exposure to the sun and environmental constraints that restricted the state and activities of the payload when shuttle contamination events occurred. Therefore, DATA-CHASER's planner needed to interoperate with the shuttle flight plan to enforce numerous resource constraints.

DATA-CHASER posed a challenge for automated scheduling techniques because of its complex resource and power management requirements. The scheduler needed to identify an optimal data collection schedule, while adhering to the resource constraints. In addition, scientists wanted to be able to perform dynamic scheduling during the mission. For example, the summary data might indicate the presence of a solar flare. If this occurred, scientists could change their requirements and goals, for example, raising the priority on certain instruments, or providing longer integration

[3]These descriptions were generously provided by Steve Chien of NASA's Jet Propulsion Lab; however, the authors take responsibility for the final text. The NASA photographs are used with permission.

times. These new goals could require a different schedule of activities.

Citizen Explorer The Citizen Explorer (CX-1) satellite project is a small satellite built and operated by the Colorado Space Grant Consortium at the University of Colorado at Boulder, Colorado. CX-1 is scheduled to launch as a secondary payload aboard a Delta-II launch vehicle in November 2000. The science mission of CX-1 will focus on obtaining geographical coverage of ozone, aerosols, and ultraviolet radiation measurements using both on-board and ground-based science instruments. CX-1 mission operations will include ground-based automated planning using the ASPEN KB planning system (Willis, Rabideau, & Wilklow 1999).

CX-1 operations require managing resources such as the spectrophotometer (Speck) science instrument, battery power, solar array power and the solid-state disk. There are also several data collection modes that must be scheduled based on the spacecraft's orbital location. A typical daily CX-1 operations scenario includes Speck data collection, engineering health and status data collection, data downlink to operations ground stations and to participating schools, command set uplink, updates to the on-board executive control database, and updates to ephemeris data. Interactions between limited power availability and limited downlink opportunities (due to ground station placement, orbital constraints, onboard memory limitations, and the transmission power costs) make mission operations a complex optimization problem.

Antarctic Mapping Mission The Modified Antarctic Mapping Mission (MAMM) used RadarSAT, a Synthetic Aperture Radar (SAR) satellite operated by the Canadian Space Agency, to gather interferometry information covering the Antarctic continent from September to December 2000. The ASPEN automated planner (Chien *et al.* 2000) was used to develop and verify the MAMM mission plan, resulting in a reduction from one work year of planning effort in the first Antarctic Mapping Mission to around eight work weeks for MAMM. The mission plan was executed flawlessly aboard RadarSAT during the operation.

RadarSAT is in a 100-minute polar orbit around the earth. One mapping cycle, consisting of 356 orbits, takes about 24 days, and must end by positioning RadarSAT in the same position and trajectory at at the start of the cycle. During each of its three mapping cycles, RadarSAT collected images of the entire continent of Antarctica, with significant redundancy around interesting regions, such as dynamic areas around the coastline and fast-moving ice flows. These three sets of images were used to construct interferometry data, which will allow scientists to determine the surface flow of the continent.

The downlink scheduling problem was complex and highly constrained. Five ground stations were used to downlink information from RadarSAT. There are two methods of capturing an image: real-time, where the image is captured and simultaneously downlinked to a receiving station, and recorded, where the image is captured on the On-Board Recorder (OBR). The OBR has a capacity of approximately 916 seconds. Data on the OBR must be downlinked to a receiving station when RadarSAT is visible from the station. Each ground station is only visible for downlinking up to 15 minutes per orbit. The only station capable of receiving real-time downlinks from MAMM was MacMurdo Ground Station, which is located in Antarctica. Furthermore, the availability of each downlinking station could change: for example, if MacMurdo was shut down by weather, all imaging had to be recorded and downlinked to other stations.

Many additional operational constraints complicated the scheduling problem. For example, real-time imaging and OBR downlink can occur simultaneously, but real-time imaging and recording images to the OBR can not. The SAR imager can be on for at most 32 minutes per orbit. Each imaging activity has to be at least one minute long, including the eight-second intervals before and after imaging. In order to use the OBR, there is a 13-second spin-up time and a 2.5-second spin-down time, although if two images occur less than 30 seconds apart, the OBR continues to record. The OBR cannot play back until it has recorded all 916 seconds, and then it must play the entire tape back with no pause. There are also delays associated with switching from recording to playback mode, connecting to and disconnecting from a ground receiver, and calibrating this connection between transmissions. In the final schedule, there were approximately 819 imaging activities per cycle, of which one-third were recorded and two-thirds were real time.

The most challenging planning issue for MAMM was to ensure that all of the images were captured within the operational constraints, and that all of the data was downlinked successfully, within the downlink constraints. Because the availability of resources could change during the cycle, rapid replanning in the event of such changes was critical.

Military Air Campaign Planning

In air campaign planning, a human planner is typically given a set of high-level political and military goals (for example, "Protect U.S. citizens and forces from hostile attack") and refines the goals that are attainable (wholly or in part) by the employment of air power into more specific goals. This process iterates until each goal is directly attainable by the execution of a mission. A group of identical aircraft acting in concert performs a mission. Each mission consists of a mission type, a time and place, a type of aircraft, munitions, and the number of sorties required to execute the mission. Thus, a mission might be expressed as "Four F-15Cs to escort strike package P to target T on day D+1." Mission planning details such as flight path and altitude profile are at a lower level of granularity and may be left until later in the planning process. Support missions, which include refueling and reconnaissance, must be planned, as they must compete with combat missions for

resources such as aircraft and fuel.

There are often multiple ways to refine goals into subgoals. These refinements reflect the different strategies and tactics that are available. Available options are constrained by the situation, which includes local geography, the enemy's characteristics and capabilities, restrictions imposed by political authority, and the availability of aircraft, fuel, crews, and other resources.

The Multiagent Planning Architecture (MPA) was used to demonstrate automated planning capability within the air campaign planning domain (Wilkins & Myers 1998). This application starts with a high-level military objective, "achieve air superiority," and expands the plan down to the level of individual missions and their support missions. The planner works from a knowledge base of planning operators (encoded specifically for the planner), which encode air campaign tactics and strategy for goals from achieving air superiority down to mission-level goals. There are often multiple ways to refine goals into subgoals. Thus, the HTN operators at multiple abstraction levels encode what it means to "achieve air superiority," a concept that would be difficult to express in primitive-action planners.

The planner has access to an extensive knowledge base of available assets (including aircraft and munitions), which was downloaded from existing military databases. Each aircraft has a dozen or more properties that affect its suitability for missions, such as speed, range, crew requirements, and munitions. Constraints on these properties appear in the HTN operators. The planner also has access to the results of the (human-conducted) intelligence analysis of the situation, which the planning operators use to focus the planner on enemy strengths, weaknesses, and other salient aspects of the situation.

The air campaign planning application, like oil-spill response, requires extensive use of metric goals, such as deadlines, resource usage, and resource accumulation. Combat missions and their support missions must compete for use of pooled resources such as fuel, aircraft, and munitions. Concurrency is important as dozens or hundreds of missions must all take place at the same time. Capacity analysis is used to determine the number of missions that a given pool of resources can support.

Evaluation of plans in this domain is complex and has not been automated. Some simple measures can be computed such as whether deadlines are met, and what percentage of desired targets are attacked given available resources. In MPA, the Air Campaign Simulator (Cohen, Anderson, & Westbrook 1996) from the University of Massachusetts provided many Monte Carlo simulations of plans, and the results were presented to the user through visualization tools. Many variables could be viewed, including levels of destruction of the targets and attrition of assets for both friend and foe. However, only humans can evaluate some of the more complex effects such as political costs and benefits.

Using Knowledge in Knowledge-Based Planning

We describe some of the uses made of domain knowledge in current KB planners. These features may be candidates for extending non-KB planners, and in some cases such extensions are currently being explored. Kautz and Selman (1998) identify three kinds of planning knowledge: knowledge about the domain, knowledge about good plans, and explicit search-control knowledge. KB 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 (see the discussion of expressiveness below).

KB Planning

Our intent is not to provide a comprehensive survey of KB planning approaches in this paper, although they are sometimes ignored in other planning surveys (Weld 1999). Instead, we mention several examples of KB planners and draw our examples of knowledge use from them. Hierarchical task network (HTN) planning is the most studied and well understood of the KB methods. The best-known knowledge-intensive applications of HTN are SIPE–2 and O-Plan.

Smith, Frank, and Jonsson (2000) have identified a common framework that is emerging from the NASA work: the use of interval representations for actions and propositions, and constraint-satisfaction techniques for reasoning about these intervals. They refer to this as the constraint-based interval approach. More recent KB approaches include Ozone, Remote Agent Experiment Planner/Scheduler (RAX-PS), and ASPEN. Each of these is described briefly here.

Ozone (Smith, Lassila, & Becker 1996; Becker & Smith 2000), a planning and scheduling toolkit, is centered on a knowledge-intensive modeling of the problem domain. A model is specified in terms of basic types of entities, operations, resources, demands, and products. Ozone provides knowledge-structuring primitives for each of these, including several specialized resource classes. Operations can be organized hierarchically to model processes at different levels of detail.

ASPEN (2000) automates planning and scheduling for space mission operations. It provides, among other capabilities, an expressive constraint modeling language, a language for representing plan preferences, constraint reasoning systems, and a graphical interface for visualizing plans in mixed-initiative systems. These capabilities are used to model many forms of knowledge, including spacecraft operability constraints, flight rules, spacecraft hardware models, science experiment goals, and operations procedures. High-level activities can be decomposed into lower-level activities using ASPEN's activity hierarchies.

RAX-PS (Jonsson *et al.* 2000) generates plans that could be safely executed on the Deep Space One spacecraft. The plans achieve high-level goals that are provided as inputs to the planner, while satisfying resource constraints and

complex flight safety rules. As in ASPEN, large amounts of knowledge are encoded about spacecraft resources, constraints, and procedures. RAX-PS also provides a rule language for the search controller that can be used to help avoid inefficient searches. RAX-PS incorporates specialized knowledge about the development of plan fragments from "planning experts," which are generally legacy software systems or other specialized software.

SIPE–2 (Wilkins 1990; Wilkins *et al.* 1995) is a domain-independent HTN planner that models various types of domain knowledge. For example, SIPE–2 includes languages to represent activities at multiple levels of abstraction (HTN operators, also known as methods or schemas), knowledge about a user's preferences (Myers 1996) (which are expressed as advice to the planner), search-control knowledge, and knowledge about plan repair during execution. 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 (which are mentioned in constraints), 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.

O-Plan (Tate, Drabble, & Kirby 1994) is a domain-independent HTN planner with the ability to encode extensive domain knowledge, including temporal constraints, object/variable constraints, resource constraints, goal structure, and condition types. Plug-in constraint managers can be used to extend or modify system capabilities. O-Plan's agenda mechanism provides flexible control of the planning and execution process. Applications include space station assembly and the control of a simple satellite.

In the following sections, we mention features of these KB planning systems that are good candidates for extending primitive-action planners, particularly disjunctive planning techniques.

Expressiveness

It has been known for some time that HTN formalisms are more expressive than the STRIPS formalism used by most primitive-action planners, roughly analogous to the additional expressivity of context-free grammars over right-linear (regular) grammars (Erol, Hendler, & Nau 1994a).

In practice, the gap in expressiveness is very wide. In the problems addressed by current KB planners, actions can occur concurrently and have different durations. Goals can include temporal deadlines and constraints, maintenance conditions, and accumulation of metric quantities of some entity. Goals and actions can be at multiple levels of abstraction. Metric resource constraints must be satisfied. All of these aspects are problematic in the STRIPS formalism.

The KB approaches mentioned provide languages for expressing the types of goals and constraints mentioned above, making them suitable for complex domains. Figure 4 shows a hypothetical knowledge-based planning system, illustrating the range of domain knowledge, inputs, and outputs that may be required for planning in real-world domains. General domain knowledge includes knowledge about actions, tactics, and strategies, at multiple abstraction levels, as well as situation assessment information, knowledge about resources, world knowledge, and so forth. Problem-specific inputs for a particular planning session may include goals and assumptions, constraints, additional resources that may be brought to bear, and advice or guidance from the user.

Encoding activities at multiple *abstraction levels* is crucial in many complex problems. The higher levels can model various solution methods and constraints on the goal and plan structure, which can be required by the domain or desired for efficient search. The high-level goals in KB planners may only be expressible in terms of primitive actions as disjunctions of thousand or millions of possible final states (corresponding to all possible plans that achieve the high-level goal). Multiple levels may be necessary for user interaction or to support different planning interactions for different levels of management. In time-critical domains, multiple abstraction levels may be required to quickly produce a plan within the available time. ASPEN, Ozone, O-Plan, and SIPE–2 all support hierarchical descriptions at multiple abstraction levels.

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.

All the KB planners mentioned can *reason about numbers*, a capability that is crucial in nearly all their applications. For example, a planning variable may be constrained to refer to a runway with length greater than 9,000 feet, multiple-capacity 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 (often concurrent) actions contribute to the accumulation. For example, 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 nearly all complex problems. All of the KB planners mentioned here have temporal constraint reasoners. In the HTN planners, they are plug-in modules that can be replaced by external temporal reasoners. For example, SIPE–2 has two different modes for reasoning about time. The most general allows specification of any of the 13 "Allen relations" between any two nodes. The temporal constraints are solved separately from the other constraints by passing them to Tachyon (Arthur & Stillman 1992).

Situation-dependent effects of actions are deduced by a causal theory in SIPE–2, but not supported by O-Plan. Such effects have proven their use in practice—without them, the

number of operators can grow exponentially in complex domains. As mentioned earlier, Smith and Weld have developed factored expansion to address this problem. Correcting an error in previous publications, SIPE–2 does recalculate these deductions (and always has) when new actions or ordering links are added to the plan before the action in question.

Finally, calls can be made to "planning experts," which are specialized software modules, including legacy software. RAX-PS uses such experts in the development of plan fragments. In SIPE–2, 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. These techniques allow encoding of knowledge in arbitrary domain-specific LISP code, for knowledge that cannot easily be modeled in the planner's formalism, and for sophisticated numerical calculations.

Correctness

Erol, Handler, and Nau (1994b) gave a formal definition of HTN planning, and also analyzed its complexity (1994a). Since then, many HTN planners have been proven correct and complete (for example, SHOP (Nau *et al.* 1999)). Planning and execution in real-world domains generally cannot be completely modeled. Although particular properties of a system can be formally verified, system validation must often rely on empirical methods, which raises many challenging issues (Smith, Feather, & Muscettola 2000). Defining evaluation criteria and correctness requirements for empirical tests is another type of knowledge that must be specified.

User Interaction

In many real-world domains, plans and actions have far-reaching effects, not all of which are modeled within the planner's formalism. For example, political consequences of actions may be important in choosing a plan, but difficult or impossible to model formally. Thus, it is often necessary to have an interactive planner that allows a human expert to guide plan development. In addition, experienced human planners can guide the search effectively, and are often reluctant to give control to an automated system in any case.

Hierarchical knowledge like that used by KB planners often models the world in the same way that human users do, using the same abstractions (generally provided by the human experts themselves). For example, in the air campaign planning domain, higher-level goals include achieving air superiority and breaching the enemy's air defenses. Human users use these same abstractions and ontology, so they can naturally, for example, advise the system on how to breach air defenses or drill down to see how the air defenses were breached in an effort to understand the plan. 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, disjunctive planning approaches model 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.

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. ASPEN, Ozone, O-Plan, and SIPE–2 all provide a GUI to aid in generating plans, viewing complex plans, and following and controlling the planning process. The GUI can also be used to view information relevant to planning decisions.

SIPE–2 provides hyperlinked descriptions of plans and plan objects in a web server. Particularly useful for visualizing the plan derivation and structure is the ability to view a tree rooted at the most abstract goals and selectively drill down through abstraction levels for selected goals (such as breaching air defenses). When the plan contains thousands of nodes, selective drill-down is often a user's preferred method for understanding it.

Several of the KB techniques mentioned provide flexible and powerful interactive planners. For example, the user may be able to interact with the planning process at many levels of detail, and may direct the planner to solve certain parts of the problem automatically. Under interactive control in SIPE–2, the user can determine (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 (Wilkins 1999). The user can also control or influence the plan development process using the Advisable Planner (Myers 1996), which allow users to 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

It is sometimes argued that the knowledge used by HTN planners is "simply search-control knowledge," rather than part of the problem statement. (We have argued above that KB planners encode much more than just search-control knowledge.) 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.

Ozone, SIPE–2, and other systems represent invariant object properties in a hierarchical ontology, which describes the classes to which an object belongs, and allows for inheritance of properties. The ontology encodes a large amount of knowledge, and the planner can reason more efficiently about this knowledge because it knows that the relationships cannot change as actions are performed.

O-Plan and SIPE–2 have, for a long time, separated various classes of constraints for efficiency, for example, solv-

ing temporal constraints and constraints on static attributes of objects separately. O-Plan pioneered the use of modular, specialized constraint solvers used at certain intervals. This separation of constraints is an example of an HTN idea that has migrated to disjunctive planners in recent work showing that performance of disjunctive planners can be improved by separating out resource reasoning to prevent thrashing (Srivastava & Kambhampati 1999).

How soon to make commitments in the search will depend on the search strategy being used and problem being solved. SIPE–2 and O-Plan have developed techniques to exploit the least-commitment approach (Myers & Wilkins 1998; Tate, Drabble, & Kirby 1994; Wilkins 1990). For example, in SIPE–2 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. (Early instantiation might result in a poor choice and failure to find a solution leading to a possible exploration of a large search space.) Because uninstantiated variables increase computational complexity, domain-specific knowledge can be used to require early instantiation of variables (by software "planning experts") at particular points in the planning process. Human experts often know when certain instantiations can be done without adversely affecting solution quality (Myers & Wilkins 1998). Thus, this knowledge can improve performance without sacrificing quality.

In some KB systems, 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, as well as in procedural reasoning systems (Georgeff & Ingrand 1989).

HTN systems often rely on *plan critics* that find conflicts or flaws in a plan. Plan critics can be invoked after some number of plan modifications rather than after every modification, thus reducing computational costs during plan expansion. Examples of plan critics include finding resource conflicts, failed preconditions or unsatisfiable constraints. In SIPE–2, domain knowledge can be used to increase or decrease the frequency of plan critic application (Wilkins 1990).

Finally, Ozone, ASPEN, and O-Plan include specialized resource classes. The system can efficiently implement specialized reasoners for these classes, instead of trying to represent, for example, multiple-capacity resources in the underlying planning formalism.

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 primitive-action planners, although some of them, such as handling uncertainty, are the subject of ongoing research (Boutilier, Dean, & Hanks 1999; Onder & Pollack 1999; Majercik & Littman 1998; Smith & Weld 1998; Weld, Anderson, & Smith 1998; Kushmerick, 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 sorts 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.

Erol, Handler, and Nau (1994b) showed that if a domain is completely modeled, then an HTN planner can provide a guarantee that the plans it produces are correct with respect to this domain model. However, for realistic domains, evaluation criteria other than correctness and plan length will have to be factored in explicitly. Interacting effectively with humans will be essential because we will never model every possible issue that might affect a planning decision. Humans often have evaluation criteria that cannot be captured precisely, and have common-sense knowledge that allows them to determine appropriate actions in unusual situations that were unforeseen when the domain was modeled. Evaluating the plans that are produced should be evaluated in the same way that plans produced by humans are evaluated:

for example, scoring performance against a simulator or the real world, or having human experts evaluate the plans by hand.

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, Williams and Nayak (1996) used a purely propositional representation. However, it seems unlikely that most interesting problems will be amenable to such an approach, and other NASA applications have required richer representations (Chien *et al.* 1996).

All types of systems mentioned above (including both disjunctive planners and the KB planners mentioned) ignore the qualification problem and have simplistic solutions to the frame problem (Pollock 1998). These issues must be addressed by future KB planners. Among other problems with traditional planners, Pollock shows that new perceptions that contradict old assumptions cause difficulty. Pollock's system handles these problems more robustly, although it does not currently appear scalable to real-world problems. Unsurprisingly, significantly more knowledge must be encoded, such as knowledge about causation, defeasibility, and when causation can be "undercut".

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, these 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 against which researchers could test new ideas. On the other hand, they tended to stifle research that did not 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

Adding Knowledge to Disjunctive Planners

Much of the effort of the planning community is currently focused on improving the performance of disjunctive planners, which embody a form of primitive-action planning. 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. These systems all use STRIPS-style planning knowledge to represent a planning problem, and then transform the problem into a propositional form that can be solved using efficient graph manipulation or constraint satisfaction techniques.

To impact realistic problems, we predict that disjunctive planners 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 knowledge incorporation is already starting to occur. For example, there is initial work on adding knowledge about the 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 disjunctive planners. 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 disjunctive planner may have to be carefully chosen for the problem being solved and the algorithm being used.

KB approaches are rightly criticized for the expense of modeling a new domain. However, we conjecture that building computationally efficient encodings for disjunctive planners 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). disjunctive planners often start with STRIPS-based encodings, further restricting the types of actions that can be encoded, and causing a possible exponential explosion in the number of operators when context-dependent effects are not permitted.

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 Bayesian networks. Bayesian network 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 disjunctive planning approaches, they use a propositional representation, and do not address many of the other challenges posed by realistic learning problems. As with disjunctive planning approaches, the rush of enthusiasm over Bayesian network 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 disjunctive planning 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 disjunctive planners overlook completely. As we discussed earlier, dijunctive planning researchers within the planning community are starting to look toward 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 disjunctive planning 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 do not yet have a good handle on formulating or solving. Results may be more difficult to achieve or quantify for the latter problems, but that does not mean we should not 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 (p.624), "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 primitive-action approaches are almost exclusively puzzle-style problems (or "real-world" problems that have been reformulated as puzzles).

We favor using primitive-action methods to solve puzzle-style subproblems that can be handled by constraint satisfaction engines in acceptable time. However, AI planners also need to provide support for the commonsense reasoning aspects of the problem so that plans can be used to guide behavior while embedded in a complex, dynamic environment. We have argued that incorporating knowledge, encoded in understandable structures, into the planning process is the most promising way to provide these abilities. HTN planning methods are better suited than disjunctive planners for such problems because HTN systems can interact with humans effectively, use more expressive representations, and can make use of domain knowledge to scale up to complex problems. However, HTN methods still have significant limitations, and we have argued that one must use still more knowledge (both in quantity and in quality) than HTN planners do in order to solve the hardest problems.

Although primitive-action 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 disjunctive planning approaches, and the associated evaluation techniques and "puzzle-style" problem suite, to overly influence the field, making it more difficult for advanced KB planning 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, Steve Chien and Foster Provost for contributing their ideas to this paper, and to Steve Chien for parts of the section on space applications.

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. Pittsburgh, PA: American Association for Artificial Intelligence, Menlo Park, CA.

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

Corporate Research and Development Center, Schenectady, NY.

Bacchus, F.; Kautz, H.; Smith, D. E.; Long, D.; Geffner, H.; and Koehler, J. 2000. The fifth international conference on artificial intelligence planning and scheduling: Planning competition. http://www.cs.toronto.edu/aips00.

Becker, M. A., and Smith, S. F. 2000. Mixed-initiative resource management: The AMC barrel allocator. In *Proc. of the 2000 International Conference on AI Planning and Scheduling*, 32–41. Breckenridge, CO: American Association for Artificial Intelligence, Menlo Park, CA.

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, D.; 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.

Chien, S.; Rabideau, G.; Willis, J.; and Mann, T. 1999. Automating planning and scheduling of shuttle payload operations. *Artificial Intelligence* 114(1):239–255.

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. 2000. ASPEN—automating space mission operations using automated planning and scheduling. In *SpaceOps 2000*.

Cohen, P.; Anderson, S.; and Westbrook, D. 1996. Simulation for ARPI and the Air Campaign Simulator. In Tate, A., ed., *Advanced Planning Technology: Technological Achievements of the ARPA/Rome Laboratory Planning Initiative*, 113–118.

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

Erol, K.; Hendler, J.; and Nau, D. S. 1994b. UMCP: A sound and complete procedure for hierarchical task-network planning. In *Proc. of the 1994 International Conference on AI Planning Systems*.

Georgeff, M. P., and Ingrand, F. F. 1989. Decision-making in an embedded reasoning system. In *Proc. of the 1989 International Joint Conference on Artificial Intelligence*, 972–978.

Ginsberg, M. L. 1996. Do computers need common sense? In *Proc. of 5th International Conference on Knowledge Representation and Reasoning*, 620–626.

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.

Jonsson, A.; Morris, P.; Muscettola, N.; and Rajan, K. 2000. Planning in interplanetary space: Theory and practice. In *Proc. of the 2000 International Conference on AI Planning and Scheduling*, 177–186. Breckenridge, CO: American Association for Artificial Intelligence, Menlo Park, CA.

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

Kambhampati, S. 1997. Challenges in bridging plan synthesis paradigms. In *Proceedings of the 15th International Joint Conference on Artificial Intelligence*, 44–49.

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. Pittsburgh, PA: American Association for Artificial Intelligence, Menlo Park, CA.

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.

Knoblock, C. 1996. AI planning systems in the real world. *IEEE Expert* 11(6):4–12.

Kushmerick, N.; Hanks, S.; and Weld, D. 1994. An algorithm for probabilistic least-commitment planning. In *Proc. of the 1994 National Conference on Artificial Intelligence*, 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.

Long, D. 2000. The AIPS-98 planning systems competition. *AI Magazine* 21(2):13–33.

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. Pittsburgh, PA: American Association for Artificial Intelligence, Menlo Park, CA.

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. Pittsburgh, PA: American Association for Artificial Intelligence, Menlo Park, CA.

McDermott, D. 2000. The 1998 AI Planning Systems competition. *AI Magazine* 21(2):35–55.

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)*, 112–123. Cambridge, MA: Morgan Kaufmann Publishers Inc., San Francisco, CA.

Myers, K. L. 1999. CPEF: A continuous planning and execution framework. *AI Magazine* 20:63–70.

Nau, D.; Cao, Y.; Lotem, A.; and noz Avila, H. M. 1999. SHOP: Simple hierarchical ordered planner. In *Proc. of the 1999 International Joint Conference on Artificial Intelligence*, 968–983.

Onder, N., and Pollack, M. E. 1999. Conditional, probabilistic planning: A unifying algorithm and effective search control mechanisms. In *Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99)*, 577–584. AAAI Press.

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):127–132.

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.

Smith, B.; Feather, M.; and Muscettola, N. 2000. Challenges and methods in validating the remote agent planner. In *Proc. of the 2000 International Conference on AI Planning and Scheduling*, 254–263. Breckenridge, CO: American Association for Artificial Intelligence, Menlo Park, CA.

Smith, D. E.; Frank, J.; and Jonsson, A. K. 2000. Bridging the gap between planning and scheduling. *Knowledge Engineering Review* 15(1).

Smith, S.; Lassila, O.; and Becker, M. 1996. Configurable, mixed-initiative systems for planning and scheduling. In Tate, A., ed., *Advanced Planning Technology: Technological Achievements of the ARPA/Rome Laboratory Planning Initiative*, 235–241.

Srivastava, B., and Kambhampati, S. 1999. Scaling up planning by teasing out resource scheduling. In *Proc. of European Conference on Planning*.

Tate, A.; Drabble, B.; and Kirby, R. B. 1994. O-Plan2: an open architecture for command, planning, and control. In Fox, M., and Zweben, M., eds., *Intelligent Scheduling*. Morgan Kaufmann Publishers Inc., San Francisco, CA. 213–239.

Weld, D. S.; Anderson, C. R.; and Smith, D. E. 1998. Extending Graphplan to handle uncertainty and sensing actions. In *Proc. of the 1998 National Conference on Artificial Intelligence*, 897–904. AAAI Press.

Weld, D. S. 1999. Recent advances in AI planning. *AI Magazine* 20(2):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.; Myers, K. L.; Lowrance, J. D.; and Wesley, L. P. 1995. Planning and reacting in uncertain and dynamic environments. *Journal of Experimental and Theoretical AI* 7(1):197–227.

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

Wilkins, D. E. 1999. *Using the SIPE-2 Planning System: A Manual for Version 6.0*. SRI International Artificial Intelligence Center, Menlo Park, CA.

Williams, B. C., and Nayak, P. P. 1996. A model-based approach to reactive self-configuring systems. In *Proc. of the 1996 National Conference on Artificial Intelligence*, 971–978. AAAI Press.

Willis, J.; Rabideau, G.; and Wilklow, C. 1999. The Citizen Explorer scheduling system. In *Proceedings of the IEEE Aerospace Conference*.

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.

Dr. David E. Wilkins is a Senior Computer Scientist in the Artificial Intelligence Center at SRI International. A fellow of the American Association of Artificial Intelligence (AAAI), he received his Ph.D. in computer science from Stanford University in 1979. His current research focuses on continuous planning and execution monitoring, mixed-initiative planning, and multiagent planning. He is interested in the application of these technologies to real world problems. His e-mail address is wilkins@ai.sri.com.

Dr. Marie desJardins is a Senior Computer Scientist in the Artificial Intelligence Center at SRI International. Her current research projects focus on distributed planning and negotiation, machine learning, and information management. Other research interests include probabilistic reasoning, decision theory, and intelligent tutoring systems. desJardins received her Ph.D. in computer science-AI from the University of California at Berkeley in 1992. Her email address is marie@ai.sri.com.

Figure 2: The space shuttle Discovery lands at Kennedy Space Center after successfully completing mission STS-85. *Photo by National Aeronautics and Space Administration (taken by Bionetics). Used with permission.*

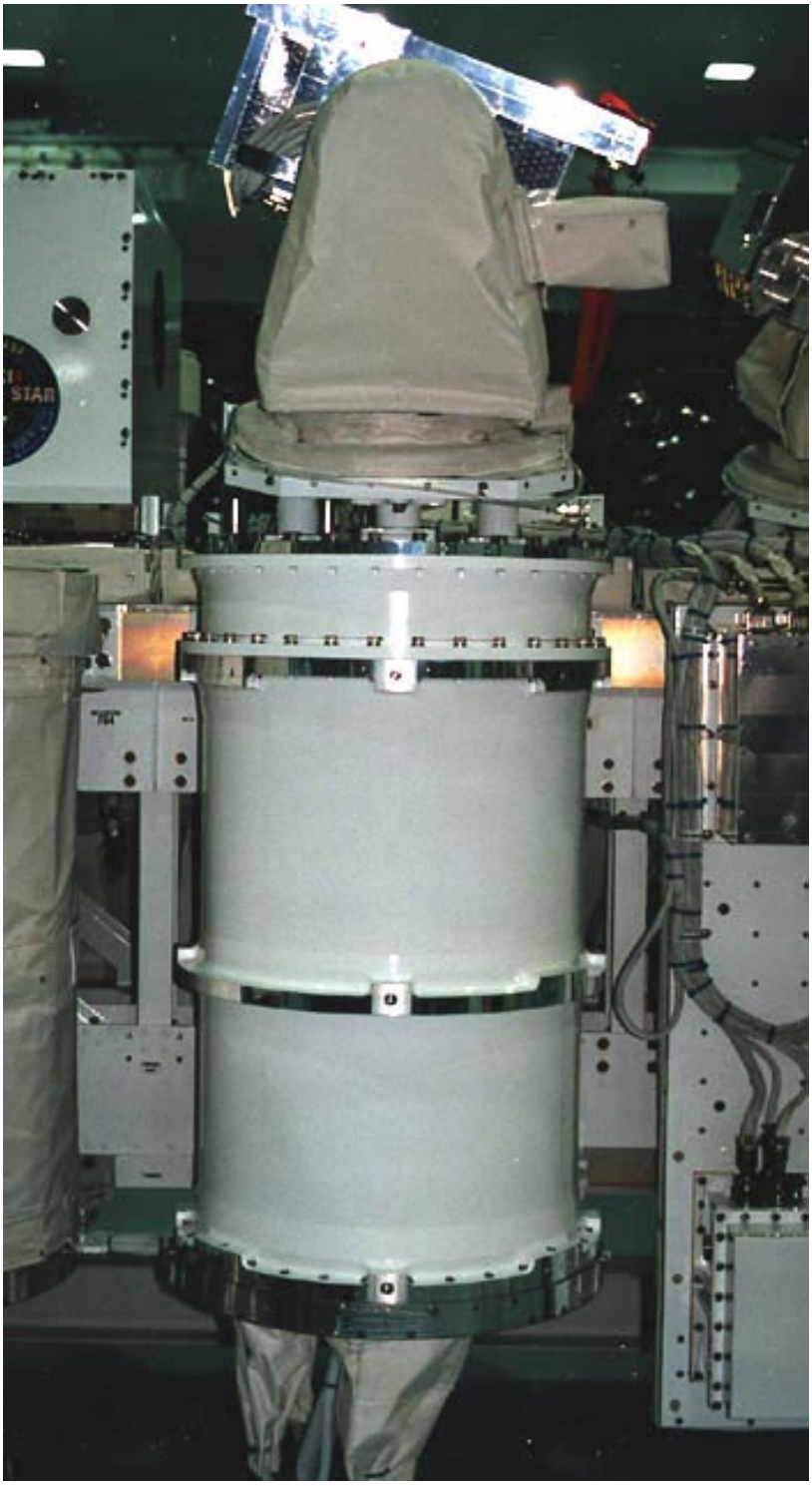

Figure 3: The CHASER module of the DATA-CHASER planning and scheduling system that flew aboard the space shuttle Discovery in August 1997. *Photo by National Aeronautics and Space Administration. Used with permission.*

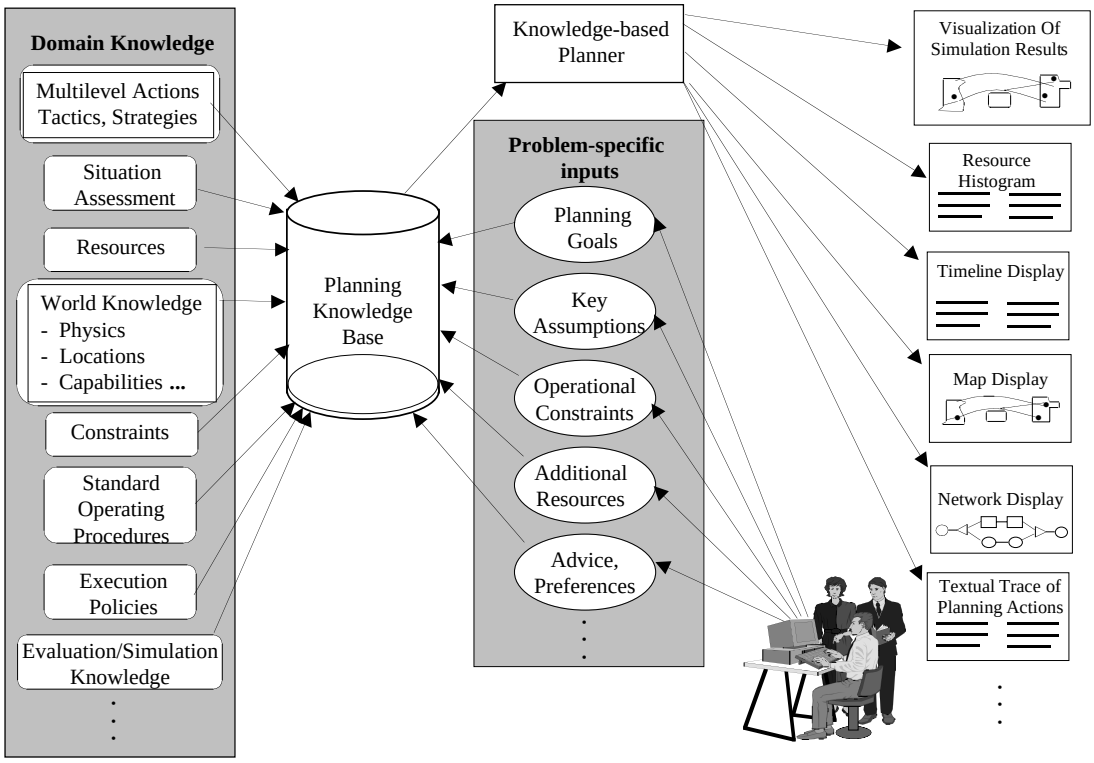

Figure 4: Potential sources of knowledge and modes of interaction for a hypothetical knowledge-based planning system.