AI Magazine Volume 24 Number 2 (2003) (© AAAI)

Learning-Assisted Automated Planning

Looking Back, Taking Stock, Going Forward

Terry Zimmerman and Subbarao Kambhampati

■ This article reports on an extensive survey and analysis of research work related to machine learning as it applies to automated planning over the past 30 years. Major research contributions are broadly characterized by learning method and then descriptive subcategories. Survey results reveal learning techniques that have extensively been applied and a number that have received scant attention. We extend the survey analysis to suggest promising avenues for future research in learning based on both previous experience and current needs in the planning community.

In this article, we consider the symbiosis of two of the most broadly recognized hallmarks of intelligence: (1) *planning*—solving problems in which one uses beliefs about actions and their consequences to construct a sequence of actions that achieve one's goals—and (2) *learning*—using past experience and precepts to improve one's ability to act in the future. Within the AI research community, machine learning is viewed as a potentially powerful means of endowing an agent with greater autonomy and flexibility, often compensating for the designer's incomplete knowledge of the world that the agent will face and incurring low overhead in terms of human oversight and control. If we view a computer program with learning capabilities as an agent, then we can say that learning takes place as a result of the interaction of the agent and the world and observation by the agent of its own decision-making processes. Planning is one such decision-making process that such an agent might undertake, and a corpus of work spanning some 30 years attests that it is an interesting, broad, and fertile field in which learning techniques can be applied to advantage. We focus here on this learning-in-planning research and utilize both tables and graphic maps of existing studies to spotlight the combinations of planning-learning methods that have received the most attention as well as those that have scarcely been explored. We do not attempt to provide, in this limited space, a tutorial of the broad range of planning and learning methodologies, assuming instead that the interested reader has at least passing familiarity with these fields.

A cursory review of the state of the art in learning in planning during the early to mid-1990s reveals that the primary impetus for learning was to make up for often debilitating weaknesses in the planners themselves. The general-purpose planning systems of even a decade ago struggled to solve simple problems in the classical benchmark domains; blocks world problems of 10 blocks lay beyond their capabilities as did most logistics problems (Kodtratoff and Michalski 1990; Minton 1993). The planners of the period used only weak guidance in traversing their search spaces, so it is not surprising that augmenting the systems to learn some such guidance was often a winning strategy. Relative to the largely naïve base planner, the learning-enhanced systems demonstrated improvements in both the size of problems that could be addressed and the speed with which they could be solved (Kambhampati, Katukam, and Qu 1996; Leckie and Zukerman 1998; Minton et. al. 1989; Veloso and Carbonell 1993).

With the advent of several new genres of planning systems in the past five to six years, the entire base-performance level against which any learning-augmented system must compare has shifted dramatically. It is arguably a more difficult proposition to accelerate a planner in this generation by outfitting it with some form of online learning because the overhead cost incurred by the learning system can overwhelm the gains in search efficiency. This, in part, might explain why the planning community appears to have paid less attention to learning in recent years. From the machine learning–community perspective, Langley (1997, p. 18) remarked on the swell of research in learning for problem solving and planning that took place in the 1980s as well as to note the subsequent tailoff: "One source is the absence of robust algorithms for learning in natural language, planning, scheduling, and configuration, but these will come only if basic researchers regain their interest in these problems."

Of course, interest in learning within the planning community should not be limited to anticipated speedup benefits. As automated planning has advanced its reach to the point where it can cross the threshold from toy problems to some interesting real-world applications, a variety of issues come into focus. They range from dealing with incomplete and uncertain environments to developing an effective interface with human users.

Our purpose in this article is to develop, using an extensive survey of published work, a broad perspective of the diverse research that has been conducted to date in learning in planning and to conjecture about profitable directions for future work in this area. The remainder of the article is organized into three parts: (1) what learning is likely to be of assistance in automated planning, (2) what roles has learning actually played in the relevant planning research conducted to date, and (3) where might the research community gainfully direct its attentions in the near future. In the section entitled Where Learning Might Assist Planning, we describe a set of five dimensions for classifying learning-in-planning systems with respect to properties of both the underlying planning engine and the learning component. By mapping the breadth of the surveyed work along these dimensions, we reveal some underlying research trends, patterns, and possible oversights. This mapping motivates our speculation in the final section on some promising directions for such research in the near future, given our current generation of planning systems.

Where Learning Might Assist Planning

In a number of ways, automated planning presents a fertile field for the application of machine learning. The simple (STRIPS) planning problem itself has been shown to be PSPACE complete (Bylander 1992); thus, for planning systems to handle problems large enough to be of interest, they must greatly reduce the size of the search space they traverse. Indeed, the great preponderance of planning research, from alternate formulations of the planning problem to the design of effective search heuristics, can be seen as addressing this problem of pruning the search space. It is therefore not surprising that the earliest and most widespread application of learning to automated planning has focused on the aspect of expediting solution search.

As automated planning advanced beyond solving trivial problems, the issue of plan quality received increased attention. Although there are often many valid plans for a given problem, generating one judged acceptable by the user or optimizing over several quality metrics can increase the complexity of the planning task immensely. A learning-augmented planning system that can perceive a user's preferences and bias its subsequent search accordingly offers a means of reducing this complexity. Learning seems to have an obvious role in mixed-initiative planning, where it might be imperative to perceive and accommodate the expertise, preferences, and idiosyncrasies of humans. Finally, expanding our view to a real-world situation in which a planning system might operate, we are likely to confront uncertainty as a fact of life, and complete and robust domain theories are rare. As we show, the study of machine learning methods in planning approaches that address uncertainty is in its infancy.

Machine learning offers the promise of addressing such issues by endowing the planning system with the ability to profit from observation of its problem space and its decision-making experience, whether or not its currently preferred decision leads to success. However, to actually realize this promise within a given application challenges the planning system designer on many fronts. Success is generally heavily dependent on complex relationships and interconnections between planning and learning. In figure 1, we suggest five dimensions that capture perhaps the most important of these system design issues: (1) type of planning problem, (2) approach to planning, (3) goal for the learning component, (4) planning-

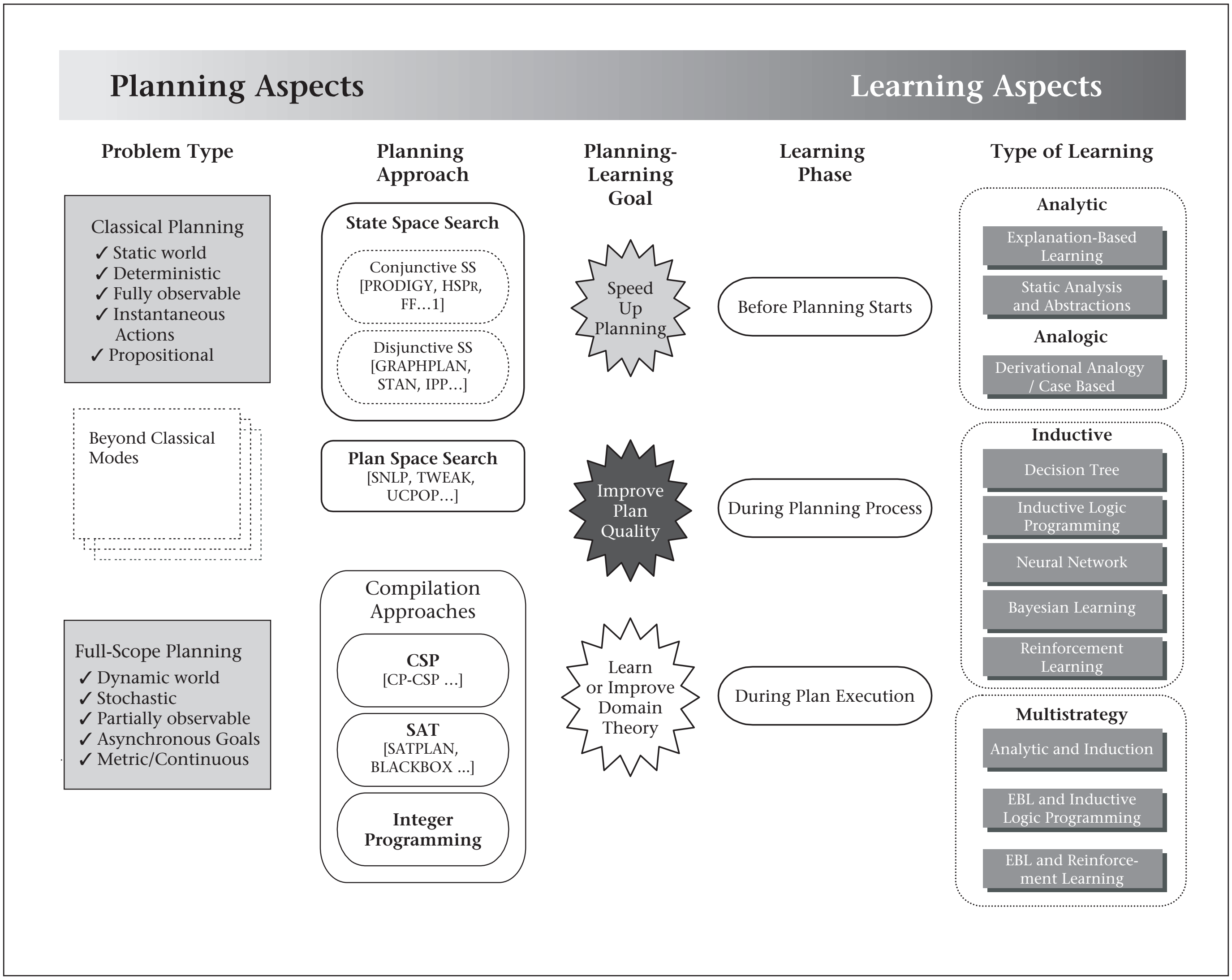

Figure 1. Five Dimensions Characterizing Automated Planning Systems Augmented with a Learning Component.
CSP = constraint-satisfaction programming. EBL = explanation-based learning. SAT = satisfiability.

execution phase in which learning is conducted, and (5) type of learning method.

We hope to show that this set of dimensions is useful in both gaining useful perspective on the work that has been done in learning-augmented planning and speculating about profitable directions for future research. Admittedly, these are not independent or orthogonal dimensions; they also do not make up an exhaustive list of relevant factors in the design of an effective learning component for a given planner. Among other candidate dimensions that could have been included are type of plan (for example, conditional, conformant, serial, or parallel actions), type of knowledge learned (domain or search control), learning impetus (data driven or knowledge driven), and type of organization (hierarchical or flat). Given the corpus of work to date and the difficulty of visualizing and presenting patterns and relationships in high-dimensional data, we settled on the five dimensions of figure 1 as the most revealing. Before reporting on the literature survey, we briefly discuss each of these dimensions.

Planning Problem Type

The nature of the environment in which the planner must conduct its reasoning defines where a given problem lies in the continuum of classes from classical to full-scope planning. Here, *classical planning* refers to a world model in which fluents are propositional, and they don't change unless the planning agent acts to change them, all relevant attributes can be observed at any time, the impact of executing an action on the environment is known and deterministic, and the effects of taking an action occur instantly. If we relax all these constraints such that fluents can take on a continuous range of values (for example, metric), a fluent might change its value spontaneously or for reasons other than agent actions—for example, the world has hidden variables, the exact impact of acting cannot be predicted, and actions have durations—then we are in the class of *full-scope planning* problems. In between these extremes lies a wide variety of interesting and practical planning problem types, such as classical planning with a partially observable world (for example, playing poker) and classical planning where actions realistically require significant periods of time to execute (for example, logistics domains). The difficulty with even the classical planning problem is that it largely occupied the full attention of the research community until the past few years. The current extension into various neoclassical, temporal, and metric planning modes has been spurred in part by impressive advances in automated planning technology over the past six years or so.

Planning Approach

Planning as a subfield of AI has roots in Newell and Simon's 1960-era problem-solving system, GPS, and theorem proving. At a high level, planning can be viewed as either a problem solver or theorem prover. Planning methods can further be seen as either search processes or model checking. Among planners most commonly characterized by search mode, there are two broad categories: (1) search in state space and (2) search in a space of plans. It is possible to further partition current state-space planners into those that maintain a conjunctive state representation and those that search in a disjunctive representation of possible states.

Planners most generally characterized as model checkers (although they also conduct search) involve recompiling the planning problem into a representation that can be tackled by a particular problem solution engine. These systems can be partitioned into three categories: (1) satisfiability (SAT), constraint-satisfaction problems (CSPs), and integer linear programming (IP). Figure 1 lists these three different methods along with representative planning systems for each. These categories are not entirely disjoint for purposes of classifying planners because some systems use a hybrid approach or can be viewed as examples of more than one method. GRAPHPLAN (Blum and Furst 1997), for example, can be seen as either a dynamic CSP or as a conductor for disjunctive state-space search (Kambhampati 2000). BLACKBOX (Kautz and Selman 1999) uses GRAPHPLAN's disjunctive representation of states and iteratively converts the search into a SAT problem.

Goal of Planner's Learning Component

There is a wide variety of targets that the learning component of a planning system might aim toward, such as learning search control rules, learning to avoid dead-end or unpromising states, or improving an incomplete domain theory. As indicated in figure 1, they can be categorized broadly into one of three groups: (1) learning to speed up planning, (2) learning to elicit or improve the planning domain theory, or (3) learning to improve the quality of the plans produced (where *quality* can have a wide range of definitions).

Learning and Improving Domain Theory Automated planning implies the presence of a *domain theory*—the descriptions of the actions available to the planner. When an exact model of how an agent's actions affect its world is unavailable (a nonclassical planning problem), there are obvious advantages to a planner that can evolve its domain theory by learning. Few interesting environments are simple and certain enough to admit a complete model of their physics, so it's likely that even "the best laid plans" based on a static domain theory will occasionally (that is, too often) go astray. Each such instance, appropriately fed back to the planner, provides a learning opportunity for evolving the domain theory toward a version more consistent with the actual environment in which its plans must succeed.

Even in classical planning, the designer of a problem domain generally has many valid alternative ways of specifying the actions, and it is well known that the exact form of the action descriptions can have a large impact on the efficiency of a given planner on a given problem. Even if the human designer can identify some of the complex manner in which the actions in a domain description will interact, he/she will likely be faced with trade-offs between efficiency and factors such as compactness, comprehensibility, and expressiveness.

Planning Speedup In all but the most trivial of problems, a planner will have to conduct considerable search to construct a solution, in the course of which it will be forced to backtrack numerous times. The primary goals of speedup learning are to avoid unpromising portions of the search space and bias the search in directions most likely to lead to high-quality plans.

Improving Plan Quality This category ranges from learning to bias the planner toward plans with a specified attribute or metric value to learning a user's preferences in plans and variations of mixed-initiative planning.

Planning Phase in Which Learning Is Conducted

At least three opportunities for learning present themselves over the course of a planning and execution cycle: (1) before planning starts, (2) during the process of finding a valid plan, and (3) during the execution of a plan.

Learning before Planning Starts Before the solution search even begins, the specification of the planning problem itself presents learning opportunities. This phase is closely connected to the aspect of learning and improving the domain theory but encompasses only preprocessing of a given domain theory. It is done offline and produces a modified domain that is useful for all future domain problems.

Learning during the Process of Finding a Valid Plan Planners capable of learning in this mode have been augmented with some means of observing their own decision-making process. They then take advantage of their experience during planning to expedite the further planning or improve the quality of plans generated. The learning process itself can either be online or offline.

Learning during the Execution of a Plan A planner has yet another opportunity to improve its performance when it is an embedded component of a system that can execute a plan and provide sensory feedback. A system that seeks to improve an incomplete domain theory would conduct learning in this phase, as might a planner seeking to improve plan quality based on actual execution experience. The learning process itself can either be online or offline.

Type of Learning

The machine learning techniques themselves can be classified in a variety of ways, irrespective of the learning goal or the planning phase they might be used in. Two of the broadest traditional class distinctions that can be drawn are between so-called *inductive* (or empirical) methods and *deductive* (or analytic) methods. In figure 1, we broadly partition the machine learning–techniques dimension into these two categories along with a multistrategy approach. We then consider additional properties that can be used to characterize a given method. The inductive-deductive classification is drawn based on the following formulations of the learning problem:

Inductive learning: The learner is confronted with a hypothesis space H and a set of training examples D. The desired output is a hypothesis h from H that is consistent with these training examples.

Analytic learning: The learner is confronted with the same hypothesis space and training examples as for inductive learning. However, the learner has an additional input: a domain theory B composed of background knowledge that can be used to help explain observed training examples. The desired output is a hypothesis h from H that is consistent with both the training examples D and the domain theory B.

Understanding the advantages and disadvantages of applying a given machine learning technique to a given planning system can help to make sense of any research bias that becomes apparent in the survey tables. The primary types of analytic learning systems developed to date, along with their relative strengths and weaknesses and an indication of their inductive biases, are listed in table 1. The major types of pure inductive learning systems are similarly described in table 2. Admittedly, the various subcategories within these tables are not disjoint, and they don't nicely partition the entire class (inductive or analytic).

The research literature itself conflicts at times about what constitutes learning in a given implementation, so tables 1 and 2 reflect the decisions made in this regard for this study.

The classification scheme we propose for learning-augmented planning systems is perhaps most inadequate when it comes to reinforcement learning. We discuss this special case, in which planning and learning are inextricably intertwined, in the sidebar "Reinforcement Learning: The Special Case."

Analogical learning is only represented in table 1 by a specialized and constrained form known as *derivational analogy* and the closely related case-based reasoning formulism. More flexible and powerful forms of analogy can be envisioned (compare Hofstadter and Marshall [1996, 1993]), but the lack of active research in this area within the machine learning community effectively eliminates more general analogy as a useful category in our learning-in-planning survey.

The three columns for each technique given in tables 1 and 2 give a sense of the degree to which the method can be effective when applied to a given learning problem, in our case, automated planning. Two columns summarize the relative strengths and weaknesses of each

Analytic Technique	Models	Strengths	Weaknesses
Nogood Learning (Memoization, Caching)	Inconsistent states and sets of fluents	Simple, fast learning Generally low computational overhead Practical, widely used	Low strength learning—each nogood typically prunes small sections of search space Difficult to generalize across problems Memory requirements can be high
Explanation-Based Learning (EBL)	Search control rules Domain refinement	Uses a domain theory—the available background knowledge Can learn from a single training example If-then rules are generally intuitive (readable) Widely used	Requires a domain theory—incorrect domain theory can lead to incorrect deductions Rule utility problem
Static Analysis and Abstractions Learning	Existing problem / domain invariants or structure	Performed "offline", benefits generally available for all subsequent problems in domain.	Benefits vary greatly depending on domain and problem
Derivational Analogy / Case-Based Reasoning (CBR)	Similarity between current state and previously cataloged states	Holds potential for shortcutting much planning effort where similar problem states arise frequently. Extendable to full analogy?	Large space required as case library builds Case-matching overhead Revising old plan can be costly

Table 1. Characterization of the Most Common Analytic Learning Techniques.

technique. The column headed Models refers to the type of function or structure that the method was designed to represent or process. A method chosen to learn a particular function is not well suited if it is either incapable of expressing the function or is inherently much more expressive than required. This choice of representation involves a crucial trade-off. A very expressive representation that allows the target function to be represented as close as possible will also require more training data to choose among the alternative hypotheses it can represent.

The heart of the learning problem is how to successfully generalize from examples. Analytic learning leans on the learner's background knowledge to analyze a given training instance to discern the relevant features. In many domains, such as the stock market, complete and correct background knowledge is not available. In these cases, inductive techniques that can discern regularities over many examples in the absence of a domain model can prove useful. One possible motivation for adopting a multistrategy approach is that analytic learning methods generate logically justified hypotheses, but inductive methods generate statistically justified hypotheses. The logical justifications fall short when the prior knowledge is flawed, and the statistical justifications are suspect when data are scarce, or assumptions about distributions are questionable.

We next consider the learning-in-planning work that has been done in light of the characterization structure given in figure 1.

What Role Has Learning Played in Planning?

We report here the results of an extensive survey of AI research literature focused on applications of machine learning techniques to planning. Research in the area of machine learning goes back at least as far back as 1959, with Arthur Samuel's (1959) checkers-playing program that improved its performance through learning. It is noteworthy that perhaps the first work in what was to become the AI field of planning (STRIPS [Fikes and Nilsson 1971]) was quickly followed by a learning-augmented version that could improve its performance by analyzing its search experience (Fikes, Hart, and Nilsson 1972). Space considerations preclude an all-inclusive survey for this 30-year span,

Inductive Technique	Models	Strengths	Weaknesses
Decision Tree Learning	Discrete-valued functions, classification problems	Robust to noisy data, missing values Learns disjunctive clauses If-then rules are easily understandable Practical, widely used	Approximating real-valued or vector-valued, functions (essentially propositional) Incapable of learning relational predicates
Artificial Neural Networks	Discrete-, real-, and vector-valued functions	Robust to noisy and complex data, errors in data	Long training times are common; learned target function is largely inscrutable
Inductive Logic Programming	First-order logic, theories as logic programs	Robust to noisy data, missing values. More expressive than propositional-based learners Able to generate new predicates. If-then rules (Horn clauses) are easily understandable	Large training sample size might be needed to acquire effective set of predicates Rule utility problem
Bayesian Learning	Probabilistic inference Hypotheses that make probabilistic predictions	Readily combine prior knowledge with observed data Modifies hypothesis probability incrementally based on each training example.	Require large initial probability sets High computational cost to obtain Bayes's optimal hypothesis
Reinforcement Learning	Control policy to maximize rewards. Fits the MDP setting	Domain theory not required Handling actions with non-deterministic outcomes Optimal policy from nonoptimal training sets, facilitates life-long learning	Depends on a real-valued reward signal for each transition Difficulty handling large state spaces. Convergence can be *slow,* space requirements can be huge

Table 2. Characterization of the Most Common Inductive Learning Techniques.

but we wanted to list either seminal studies in each category or a typical representative study if the category has many.

It is difficult to present the survey results in 2-dimensional (2D) format such that the five dimensions represented in figure 1 are usefully reflected. We used three different formats, emphasizing different combinations and orderings of the figure 1 dimensions:

First is a set of three tables organized around just two dimensions: (1) type of learning and (2) type of planning.

Second is a set of tables reflecting all five dimensions for each relevant study in the survey.

Third is a graphic representation providing a visual mapping of the studies' demographics along the five dimensions.

We discuss each of these representations in the following subsections.

Survey Tables according to Learning Type and Planning Type

Table 3A deals with studies focused primarily on analytic (deductive) learning in its various forms, and table 3B is concerned with inductive learning. Table 3C addresses studies and multistrategy systems that aim at some combination of analytic and inductive techniques. All studies and publications appearing in these tables are listed in full in the reference section.

Analytic Learning	General Applications	Planning Applications		
		State Space (Conjunctive / Disjunctive)	Plan Space	Compilation (CSP / SAT / IP)
Static / Domain Analysis and Abstractions		*Learning Abstractions* Sacerdoti (1974) ABSTRIPS Knoblock (1990) ALPINE *Static Analysis, Domain Invars* Dawson and Siklossy (1977) REFLECT Etzioni (1993) STATIC [PRODIGY] Perez and Etzioni (1992) DYNAMIC (with EBL)[PRODIGY] Nebel, Koehler, and Dimopoulos (1997) RIFO Gerevini and Schubert (1998) DISCOPLAN Fox and Long (1998, 1999) STAN/ TIM [GRAPHPlan] (Rintanen 2000)	Smith and Peot (1993) [SNLP] Gerevini and Schubert (1996) [UCPOP]	
Explanation-Based Learning (EBL)	*General Problem Solving (Chunking)* Laird et al. (1987) SOAR *Horn Clause Rules* Kedar-Cabelli (1987) Prolog-EBG *Symbolic Integration* Mitchell et al. (1986) LEX-2 *(See also multi-strategy)*	Fikes and Nilsson (1972) STRIPS Minton et al. (1989) PRODIGY Gratch and DeJong (1992) COMPOSER [PRODIGY] Bhatnagar and Mostow (1994) FAILSAFE Borrajo and Veloso (1997) HAMLET *(See also multistrategy)* Kambhampati (2000) GRAPHPlan-EBL *Permissive Real-World Plans* Bennett and DeJong (1996) GRASPER	Chien (1989) Kambhampati, Katukam, and Qu (1996) UCPOP-EBL	Wolfman and Weld (1999) LPSAT [RELSAT] *Nogood Learning* Kautz and Selman (1999) BLACKBOX (using RELSAT) Do and Kambhampati (2001) GP-CSP [GRAPHPlan]
Analogical Case-Based Reasoning	Jones and Langley (1995) EUREKA *Microdomain Analogy Maker* Hofstadter and Marshall (1993, 1996) COPYCAT *Conceptual Design* Sycara et al. (1992) CADET *Legal Reasoning by Analogy* Ashley and McLaren (1995) TRUTH-TELLER Ashley and Aleven (1997) CATO Kakuta et al. (1997)	*Learning Various Abstraction-Level Cases* Bergmann and Wilke (1996) PARIS *User-Assisted Planning* Avesani, Perini, and Ricci (2000) CHARADE *CBR Derivational* Veloso and Carbonell (1993) PRODIGY / ANALOGY *CBR Transformational* Hammond (1989) CHEF PRIAR (Kambhampati and Hendler 1992) SPA (Hanks and Weld 1995) Leake, Kinley, and Wilson (1996) *(see also multistrategy)*	*CBR Derivational* Ihrig and Kambhampati (1996) [UCPOP] *With EBL* Ihrig and Kambhampati (1997) [UCPOP]	

Table 3A. Analytic Learning Applications and Studies.

Studies in heavily shaded blocks concern planners applied to problems beyond classical planning. Implemented system and program names appear in all caps, and underlying planners and learning subsystems appear in small caps but enclosed in brackets.

Inductive Learning	General Applications	Planning Applications: State Space (Conjunctive / Disjunctive)	Planning Applications: Plan Space	Planning Applications: Compilation (CSP / SAT / IP)
Propositional Decision Trees	*Concept Learning* Hunt, Marin, and Stone (1966) CLS *General DT Learning* Quinlan (1986) ID3 Quinlan (1993) C4.5 Khardon (1999) L2ACT Cohen and Singer (1999) SLIPPER	*Learning Operators for Real-World Robotics, Clustering* Schmill, Oates, and Cohen (2000) [TBA for inducing decision tree]		
Real-Valued Neural Network	Hinton (1989) *Symbolic Rules from NN* Craven and Shavlik (1993) *Reflex/Reactive* Pomerleau (1993) ALVINN			
First-Order Logic Inductive Logic Programming (ILP)	*Hornlike Clauses* Quinlan (1990) FOIL Muggleton and Feng (1990) GOLEM Lavrac, Dzeroski, and Grobelnik (1991) LINUS	Leckie and Zukerman (1998) GRASSHOPPER [PRODIGY] Zelle and Mooney (1993) *(See also multistrategy)* Reddy and Tadepalli (1999) ExEL	Estlin and Mooney (1996) *(See also multistrategy)*	Huang, Selman, and Kautz (2000) *(See also multistrategy)*
Bayesian Learning	*Train Bayesian Belief Networks, Unobserved Variables* Dempster, Laird, and Rubin (1977) EM *Text Classification* Lang (1995) NewsWeeder *Predict Run Time of Problem Solvers for Decision-Theoretic Control* Horvitz et al. (2001)			
Other Inductive Learning		*Action Strategies and Rivest's Decision List Learning* Khardon (1999) Martin and Geffner (2000) *Plan Rewriting* Ambite, Knoblock, and Minton (2000) PbR		
Reinforcement Learning (RL)	Sutton (1988) TD / TDLambda Watkins (1989) Q Learning *Real-Time Dynamic Programming* Barto, Bradtke, and Singh (1995) Dearden, Friedman, and Russel (1998) Bayesian Q Learning	(Dietterich and Flann 1995) *(See also multistrategy)* *Incremental Dynamic Programming* Sutton (1991) DYNA *Planning with learned operators:* Garcia-Martinez and Borrajo (2000) LOPE		

Table 3B. Inductive Learning Applications and Studies.

CSP = constraint-satisfaction programming. DT = decision tree. IP = integer linear programming. NN = neural network. SAT = satisfiability. Studies in heavily shaded blocks feature planners applied to problems beyond classical planning. Implemented system and program names appear in all caps, and underlying planners and learning subsystems appear in small caps but enclosed in brackets.

Multistrategy Learning	General Applications	Planning Applications: State Space Conjunctive/Disjunctive	Planning Applications: Plan Space	Planning Applications: Compilation [CSP / SAT/ IP]
Analytic and Inductive	*Symbolic Integration* Mitchell, Keller, and Kedar-Cabelli (1986) LEX-2 *Learn CSP Variable Ordering* Zweban et al. (1992) GERRY *Incorporate Symbolic Knowledge in Neural Networks* Shavlik and Towell (1989) KBANN Fu (1989) *Learn Horn Clause Sets Focused by Domain Theory* Pazzani, Brunk, and Silverstein (1991) FOCL *Refining Domain Theories Using Empirical Data* Ourston and Mooney (1994) EITHER *Neural Networks and Fuzzy Logic to Implement Analogy* Hollatz (1999) *Genetic, Lazy RL, k-Nearest Neighbor* Sheppard and Salzberg (1995)	*Learn / Refine Operators* Carbonell and Gil (1990), Gil (1994) EXPO [PRODIGY] Wang (1996a, (1996b) OBSERVER [PRODIGY] McCluskey, Richardson, and Simpson (2002) OPMAKER *EBL and Induction:* Calistri-Yeh, Segre, Sturgill (1996) ALPS *CBR and Induction* Leake, Kinley, and Wilson (1996) DIAL Borrajo and Veloso (1997) HAMLET [PRODIGY] Zimmerman and Kambhampati (1999, 2002) EGBG, PEGG [GRAPHPLAN] *Deduction, Induction, and Genetic* Aler, Borrajo, and Isasi (1998) HAMLET-EvoCK [PRODIGY] Aler and Borrajo (2002) HAMLET-EvoCK [PRODIGY]		
Explanation-Based Learning and Neural Networks	*Domain Theory Cast in Neural Network Form* Mitchell and Thrun (1995) EBNN			
Explanation-Based Learning and Inductive Logic Programming	*Search Control for Logic Programs* Cohen (1990) AxA-EBL Zelle and Mooney (1993) DOLPHIN [FOIL/PRODIGY]	Zelle and Mooney (1993) DOLPHIN [PRODIGY/FOIL]	Estlin and Mooney (1996) SCOPE [FOIL]	*EBL, ILP, and Some Static Analysis* Huang, Selman, and Kautz (2000) [BLACKBOX-FOIL]
Explanation-Based Learning and Reinforcement Learning		Dietterich and Flann (1997) EBRL Policies		

Table 3C. Multistrategy Learning Applications and Studies.

CSP = constraint-satisfaction programming. DT = decision tree. EBL = explanation-based learning. IP = integer linear programming. NN = neural network. RL = reinforcement learning. SAT: satisfiability. Studies in the heavily shaded blocks feature planners applied to problems beyond classical planning. Implemented system and program names appear in all caps, and underlying planners and learning subsystems appear in small caps but enclosed in brackets.

The table rows feature the major learning types outlined in tables 1 and 2, occasionally further subdivided as indicated in the leftmost column. The second column contains a listing of some of the more important nonplanning studies and implementations of the learning technique in the first column. These General Applications were deemed particularly relevant to planning, and of course, the list is highly abridged. Comparing the General Applications column with the Planning columns for each table provides a sense of which machine learning methods have been applied within the planning community. The three columns making up the Planning Applications partition subdivide the applications into state space; plan space; and CSP, SAT, and IP planning. Studies dealing with planning problems beyond classical planning (as defined in Planning Problem Type earlier) appear in shaded blocks in these tables.

Table 3C, covering multistrategy learning, reflects the fact that the particular combination of techniques used in some studies could not always be easily subcategorized relative to the analytic and inductive approaches of tables 3A and 3B. This is often the case, for example, with an inductive learning implementation that exploits the design of a particular planning system. Examples include HAMLET (Borrajo and Veloso 1997), which exploits the search tree produced by the PRODIGY 4.0 planning system to lazily learn search control heuristics, and EGBG and PEGG (Zimmerman and Kambhampati 2002, 1999), which exploit GRAPHPLAN's use of the planning graph structure to learn to shortcut the iterative search episodes. Studies such as these appear in table 3c under the broader category, analytic and inductive.

In addition to classifying the studies surveyed along the learning-type and planning-type dimensions, these tables illustrate several foci of this corpus of work. For example, the preponderance of research in analytic learning as it applies to planning rather than inductive learning styles is apparent, as is the heavy weighting in the area of state-space planning. We return to such issues when discussing implications for future research in the final section.

Survey Tables Based on All Five Dimensions

The same studies appearing in tables 3A, 3B, and 3C are tabulated in tables 4A and 4B according to all five dimensions in figure 1. We have used a block structure within the tables to emphasize shared attribute values wherever possible, given the left-to-right ordering of the dimensions. Here, the two dimensions not represented in the previous set of tables, "Planning-Learning Goal" and "Learning Phase," are ordered first, so this block structure reveals the most about the distribution of work across attributes in these dimensions. It's apparent that the major focus of learning-in-planning work has been on speedup, with much less attention given to the aspects of learning to improve plan quality or building and improving the domain theory. Also obvious is the extent to which research has focused on learning prior to or during planning, with scant attention paid to learning during plan execution.

Graphic Analysis of Survey

There are obvious limitations to what can readily be gleaned from any tabular presentation of a data set across more than two or three dimensions. To more easily visualize patterns and relationships in learning-in-planning work, we have devised a graphic method of depicting the corpus of work in this survey with respect to the five dimensions given in figure 1. Figure 2 illustrates this method of depiction by mapping two studies from the survey onto a version of figure 1.

In this manner, every study or project covered in the survey has been mapped onto at least one 5-node, directed subgraph of figure 3 (classical planning systems) or figure 4 (systems designed to handle problems beyond the classical paradigm). The edges express which combinations of the figure 1 dimensional attributes were actually realized in a system covered by the survey.

Besides providing a visual characterization of the corpus of research in learning in planning, this graphic presentation mode permits quick identification of all planner-learning system configurations that embody any of the aspects of the five dimensions (nodes). For example, because the survey tables don't show all possible values in each dimension's range, aspects of learning in planning that have received scant attention are not obvious until one glances at the graphs, which entails simply observing the edges incident on any given node. Admittedly, a disadvantage of this presentation mode is that the specific planning system associated with a given subgraph cannot be extracted from the figure alone. However, the tables can assist in this regard.

Learning within the Classical Planning Framework Figure 3 indicates with dashed lines and fading those aspects (nodes) of the five dimensions of learning in planning that are not relevant to classical planning. Specifically, Learning or Improving the Domain Theory is inconsistent with the classical planning assumption of a complete and correct domain theory. Similarly, the strength of reinforcement learning lies in its ability to handle stochastic environments in which the domain theory is either unknown or incomplete. (*Dynamic programming,* a close cousin to reinforcement learning methods, requires a complete and perfect domain theory, but because of efficiency considerations, it has remained primarily of

Dimensions: Planning/ Learning Goal	Learning Phase	Type of Learning	Planning Approach	Planning Systems / Studies
Speedup	Before planning starts	Analytic Static analysis	Plan space	Smith and Peot (1993) [SNLP]
				Gerevini and Schubert (1996) [UCPOP]
			State space	Etzioni (1993) STATIC [PRODIGY]
				Dawson and Siklossy (1977) REFLECT
				Nebel, Koehler, and Dimopoulos (1997) RIFO
				Fox and Long (1998, 1999), Rintanen (2000) STAN / TIM [GrRAPHPLAN]
		Static analysis: Learn abstractions		Sacerdoti (1974) ABSTRIPS
				Knoblock (1990) ALPINE [PRODIGY]
	Before and during planning	Static analysis and EBL		Perez and Etzioni (1992) DYNAMIC [PRODIGY]
	During planning	Analytic EBL	State space	Fikes and Nilsson (1972) STRIPS
				Minton (1989) PRODIGY/EBL
				Gratch and DeJong (1992) COMPOSER [PRODIGY]
				Bhatnagar (1994) FAILSAFE
				Kambhampati (2000) GRAPHPLAN-EBL
			Plan space	Chein (1989)
				Kambhampati, Katukam, and Qu (1996) UCPOP-EBL
			(Compilation) SAT	*Nogood Learning* Kautz and Selman (1999) BLACKBOX
			LP & SAT	Wolfman and Weld (1999) LPSAT [RELSAT]
			CSP	*Nogood Learning* Do and Kambhampati (2001) GP-CSP [GRAPHPLAN]
		Analytic Analogical Case-Based Reasoning	State space	*Learning Various Abstraction-Level Cases* Bergmann and Wilke (1996) PARIS
				User Assist Planning Avesani, Perini, and Ricci (2000) CHARADE
				Transformational Analogy / Adaptation Hammond (1989) CHEF
				Kambhampati and Hendler (1992) PRIAR
				Hanks and Weld (1995) SPA
				Leake, Kinley, and Wilson (1996) DIAL
				Derivational Analogy / Adaptation Veloso and Carbonell (1993) PRODIGY / ANALOGY
			Plan space	Ihrig and Kambhampati (1996) [UCPOP]
				With EBL Ihrig and Kambhampati (1997) [UCPOP]

Table 4A. Survey Studies Mapped across All Five Dimensions, Part 1.

CSP = constraint-satisfaction programming. EBL = explanation-based learning. LP = linear programming. SAT = satisfiability. Studies in heavily shaded blocks feature planners applied to problems beyond classical planning. Implemented system and program names appear in all caps, and underlying planners and learning subsystems appear in small caps but enclosed in brackets.

Dimensions				Planning Systems / Studies
Planning / Learning Goal	Learning Phase	Type of Learning	Planning Approach	
Speedup	During planning	Inductive: Inductive logic programming (ILP)	State space	Leckie and Zuckerman (1998) GRASSHOPPER [PRODIGY] Reddy and Tadepalli (1999) ExEL
Speedup	During planning	Other induction		*Action Strategies and Rivest's Decision List Learning of Policies* Khardon (1999) Martin and Geffner (2000)
Speedup	During planning	Multistrategy	State space	Calistri-Yeh, Segre, and Sturgill (1996) ALPS
Speedup	During planning	Analytic and inductive	State space	*CBR and Induction* Leake, Kinley, and Wilson (1996) DIAL
Speedup	During planning	Analytic and inductive	State space	Zimmerman and Kambhampati (1999) EGBG [Graphplan]
Speedup	Before and during planning	EBL, ILP, and static analysis	(Compilation) SAT	Huang, Selman, and Kautz (2000) [BLACKBOX/FOIL]
Speedup		EBL and ILP	State space	Zelle and Mooney (1993) DOLPHIN [PRODIGY/FOIL]
Speedup		EBL and ILP	Plan space	Estlin and Mooney (1996) SCOPE [FOIL]
Speedup and improve plan quality	During planning	EBL and inductive	State space	Borrajo and Veloso (1997) HAMLET [PRODIGY] *Deductive-Inductive and Genetic* HAMLET-EvoCK (PRODIGY) (Aler and Borrajo 1998, 2002) Zimmerman and Kambhampati (2002) PEGG [GRAPHPLAN]
Speedup and improve plan quality	During planning	Inductive (Analysis of plan differences)		*Plan Rewriting* Ambite, Knoblock, and Minton (2000) PbR
Learn or improve domain theory	Before planning starts	(Propositional) decision trees	State space	*Learning Operators for Real World Robotics, Clustering* Schmill, Oates, and Cohen (2000) TBA for inducing decision tree
Learn or improve domain theory	During plan execution	Analytic: EBL		*Permissive Real-World Plans* Bennett and DeJong (1996) GRASPER
Learn or improve domain theory	During plan execution	Multistrategy: Analytic and inductive		*Learning / Refining Operators* Wang (1996a, 1996b) OBSERVER [PRODIGY] McClusky, Richardson, and Simpson (2002) OPMAKER Carbonell and Gil (1990); Gil (1994) EXPO [PRODIGY]
Learn or improve domain theory and improve plan quality	During planning	EBL and RL	State space	EBRL Dietterich and Flann (1997)
Learn or improve domain theory and improve plan quality	During planning	Inductive: Reinforcement learning		*Incremental Dynamic Programming* Sutton (1991) DYNA *Planning with Learned Operators* Garcia-Martinez and Borrajo (2000) LOPE

Table 4B. Survey Studies Mapped across All Five Dimensions, Part 2.

EBL = explanation-based learning. ILP = inductive logic programming. RL = reinforcement learning. SAT = satisfiability. Studies in heavily shaded blocks feature planners applied to problems beyond classical planning. Implemented system and program names appear in small caps, and underlying planners and learning subsystems appear in small caps but enclosed in brackets.

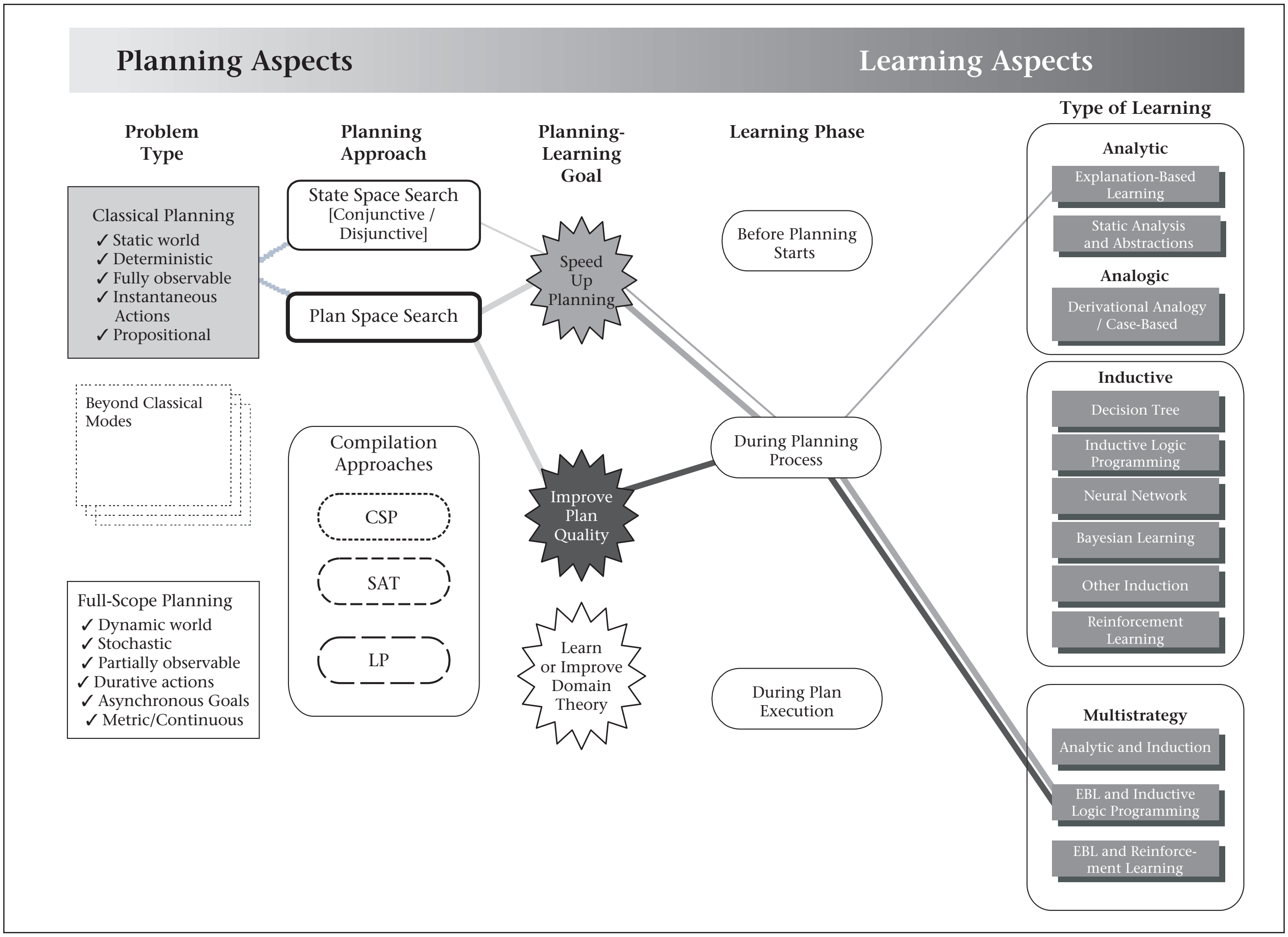

Figure 2. Example Graphic Mapping of Two Learning-in-Planning Systems.

EBL = explanation-based learning. ILP = inductive logic programming. The layout of the five dimensions in figure 1 and their range of values can be used to map the research work covered in the survey tables. By way of example, the PRODIGY-EBL system (Minton et al. 1989) is represented by the top connected series of gray lines; it's a classical planning system that conducts state-space search, and it aims to speed up planning using explanation-based learning (EBL) during the planning process. Tracing the subgraph from the left, the edge picks up the line thickness at the State-Space node and the gray shade of the Speed-Up Planning node. The SCOPE system (Estlin and Mooney 1996) is then represented as the branched series of thicker (2 pt) lines. SCOPE is a classical planning system that conducts plan-space search, and the goal of its learning subsystem is to both speed up planning and improve plan quality. Thus, the Plan-Space node branches to both Planning-Learning Goal nodes. All SCOPE's learning occurs during the planning process, using both EBL and inductive logic programming (ILP). As such, the edges converge at the during planning process node, but both edges persist to connect with the EBL and ILP node.

theoretical interest with respect to classical planning.)

Broadly, the figure indicates that some form of learning has been implemented with all planning approaches. If we consider the Learning Phase dimension of figure 3, it is obvious that the vast majority of the work to date has focused on learning conducted during the planning process. Work in automatic extraction of domain-specific knowledge through analysis of the domain theory (Fox and Long 1999, 1998; Gerevini and Schubert 1998) constitutes the learning conducted before planning. Not surprisingly, learning in the third phase, during plan execution, is not a focus for classical planning scenarios because this mode has clear affinity with improving a faulty domain theory—a nonclassical problem.

It is apparent, based on the figure 3 graph in combination with the survey tables, that explanation-based learning (EBL) has been extensively studied and applied to every planning approach and both relevant planning-learning goals. This is perhaps not surprising given that planning presumes the sort of domain theory that EBL can readily exploit. Perhaps more no-

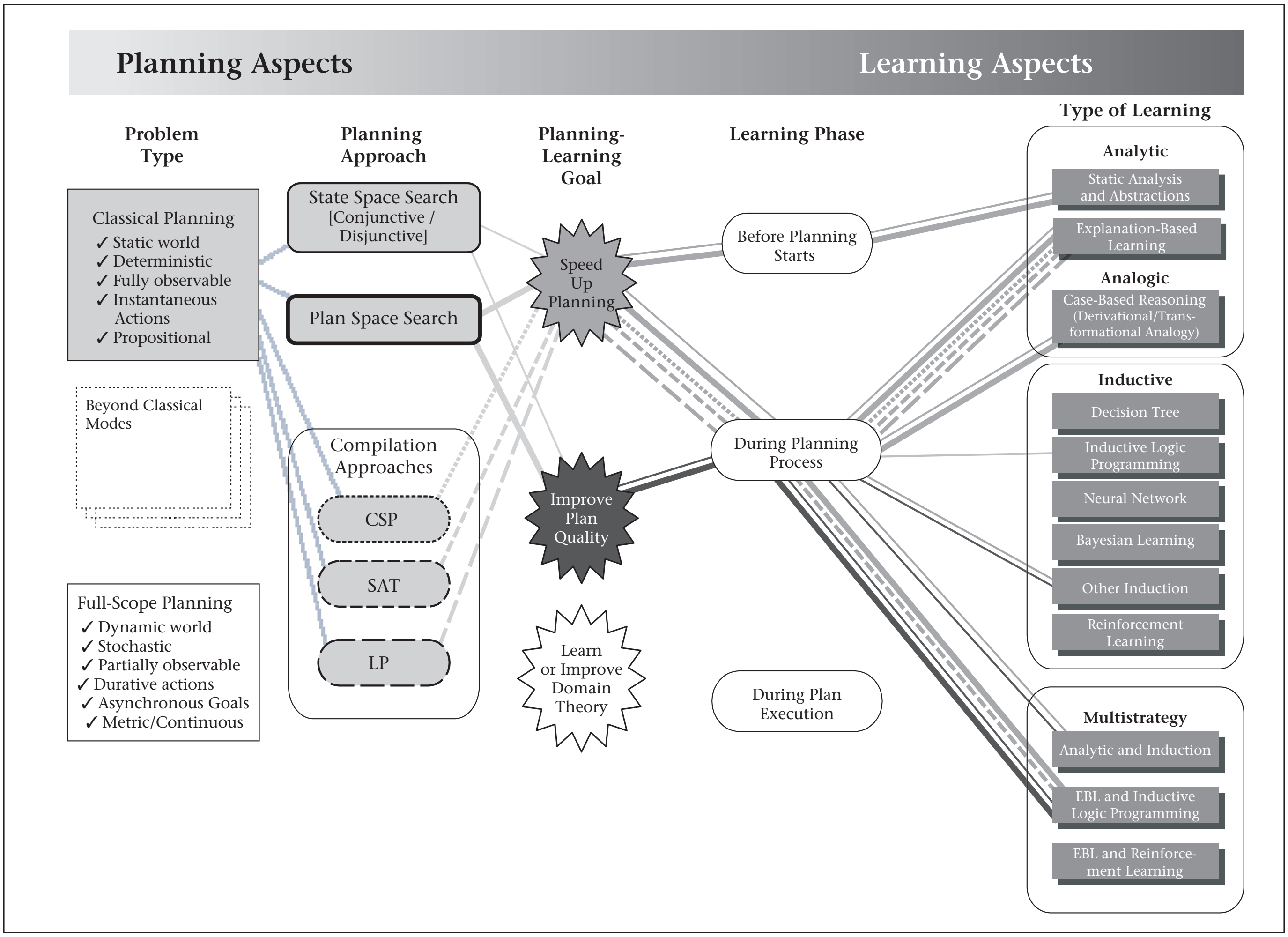

Figure 3. Mapping of the Survey Planning-Learning Systems for Classical Planning Problems on the Figure 1 Characterization Structure.

table is the scant attention paid to inductive learning techniques for classical planners. Although ILP has extensively been applied as a learning tool for planners, other inductive techniques such as decision tree learning, neural networks, and Bayesian learning, have seen few planning applications.

Learning within a Nonclassical Planning Framework Figure 4 covers planning systems designed to learn in the wide range of problem classes beyond the classical formulation (shown in shaded blocks in tables 3A, 3B, and 3C and 4A and 4B). There are, as yet, far fewer such learning-augmented systems, although this area of planning community interest is growing. Those "beyond classical planning" systems that exist extend the classical planning problem in a variety of different ways, but because of space considerations, we have not reflected these variations with separate versions of figure 4 for each combination. Learning in a dynamic, stochastic world is the natural domain of reinforcement learning systems, and as discussed earlier, this popular machine learning field does not so readily fit our five-dimensional learning-in-planning perspective. Figure 4 therefore represents reinforcement learning in a different manner than the other approaches; a single shade, brick crosshatch set of edges is used to span the five dimensions. The great majority of reinforcement learning systems to date adopt a state-space perspective, so there is an edge skirting this node. With respect to the planning-learning goal dimension, reinforcement learning can be viewed as both "improving plan quality" (the process moves toward the optimal policy) and "learning the domain theory" (it begins without a model of transition probability between states). This view is reflected in figure 4 as the vertical rein-

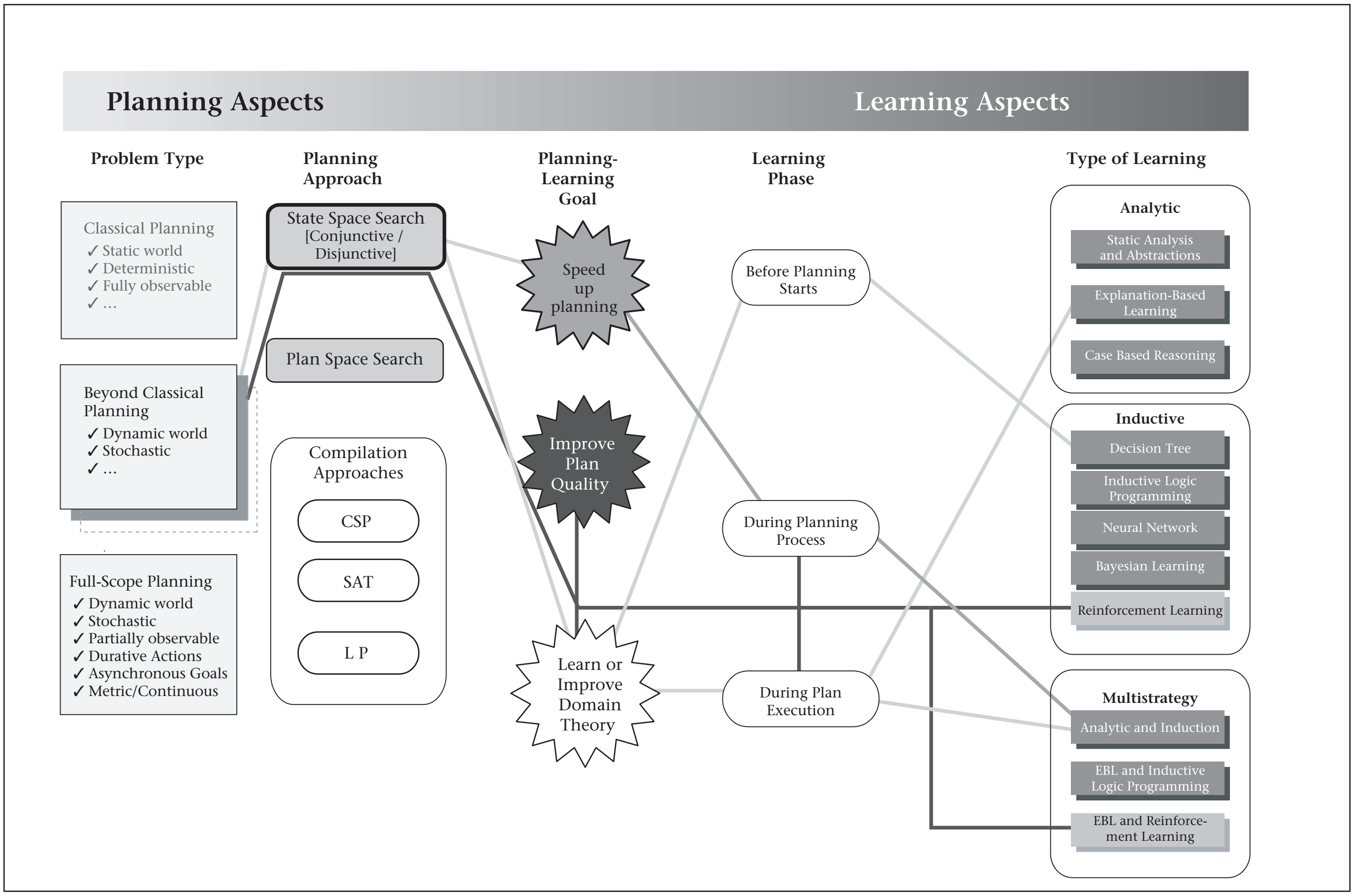

Figure 4. Mapping of the Survey Planning-Learning Systems for Beyond-Classical Planning Problems on the Figure 1 Characterization Structure.

forcement learning edge spanning these nodes under the planning-learning goal dimension. Finally, because reinforcement learning is both rooted in interacting with its environment and takes place during the process of building the plan, there is a vertical edge spanning these nodes under the learning-phase dimension.

Beyond reinforcement learning systems, figure 4 suggests at least three aspects to the learning-in-planning work done to date for nonclassical planning problems, all fielded systems plan using state-space search, most systems conduct learning during the plan execution phase, and EBL is again the learning method of choice. It is also notable that the only decision tree learning conducted in any planner is based in a nonclassical planning system.

With this overview of where we have been with learning in planning, we next turn our attention to open issues and research directions that beckon.

Where to for Learning in Automated Planning?

We organize this discussion of promising directions for future work in this field along two broad partitions: (1) apparent gaps in the corpus of learning-in-planning research as suggested by the survey tables and figures of this report and (2) recent advances in planning that suggest a role for learning notably beyond the modes investigated by existing studies.

Research Gaps Suggested by the Survey

There are significant biases apparent in the focus and distribution of the survey studies relative to the five dimensions we have defined. To an extent, these biases are to be expected because some configurations of planning-learning methods are intrinsically infeasible or poorly matched (for example, learning domain theory in a classical planning context or combining reinforcement learning with SAT, which does not capture the concept of a state). In assessing the survey tables here, however, we seek learning-in-planning configurations that are feasible, have been largely ignored, and appear to hold promise.

Nonanalytic Learning Techniques The survey tables suggest a considerable bias toward analytic learning in planning, which deserves to be questioned. Why is analytic learning so favored? In a sense, a planner using EBL is learning *guaranteed knowledge,* control information that is provably correct. However, it is well known within the machine learning community that approximately correct knowledge can be at least as useful, particularly if we're careful not to sacrifice completeness. Given the presence of a high-level domain theory, it is reasonable to exploit it to learn. However, large constraints are placed on just what can be learned if the planner doesn't also take advantage of the full planning search experience. The tables and figures of this study indicate the extent to which ILP has been used in this spirit together with EBL. This is a logical marriage of two mature methodologies; ILP in particular has powerful engines for inducing logical expressions, such as FOIL (Quinlan 1990), that can readily be employed. It is curious to note, however, that decision tree learning has been used in only one study in this entire survey, yet this inductive technique is at least as mature and features its own very effective engines such as ID3 and C4.5 (Quinlan 1993, 1986). In the 1980s, decision tree algorithms were generally not considered expressive enough to capture complex target concepts (such as under what conditions to apply an operator). However, given subsequent evolutions in both decision tree methods and the current opportunities for learning to assist the latest generation of planners, the potential of decision tree learning in planning merits reconsideration.

Learning across Problems A learning aspect that has largely fallen out of favor in recent years is the compilation and retention of search guidance that can be used across different problems and perhaps even different domains. One of the earliest implementations of this took the form of learning search control rules (for example, using EBL). There might be two culprits that led to disenchantment with learning this interproblem search control:

First is the *utility problem* that can surface when too many, or relatively ineffective rules are learned.

Second is the *propositionalization* of the planning problem, wherein lifted representations of the domain theory were forsaken for the faster processing of grounded versions involving only propositions. The cost of rule checking and matching in more recent systems that use grounded operators is much lower than for planning systems that handle uninstantiated variables.

Not conceding these hurdles to be insurmountable, we suggest the following research approaches:

One trade-off associated with a move to planning with grounded operators is the loss of generality in the basic precepts that are most readily learned. For example, GRAPHPLAN can learn a great number of "no goods" during search on a given problem, but in their basic form, they are only relevant to the given problem. GRAPHPLAN retains no interproblem memory. It is worth considering what might constitute effective interproblem learning for such a system.

The rule utility issue faced by analytic learning systems (and possibly all systems that learn search control rules) can be viewed as the problem of incurring the cost of a large set of sound, exact, and probably overspecific rules. Learning systems that can reasonably relax the soundness criterion for learned rules can move broadly toward a problem goal using generally correct search control. Some of the multistrategy studies reflected in table 3C are relevant to this view to the extent that they attempt to leverage the strengths of both analytic and inductive learning techniques to acquire more useful rules. Initial work with an approach that does not directly depend on a large set of training examples was reported in Kambhampati (1999). Here, a system is described that seeks to learn approximately correct rules by relaxing the constraint of the UCPOP-EBL system that requires regressed failure explanations from all branches of a search subtree before a search control rule is constructed.

Perhaps the most ambitious approach to learning across problems would be to extend some of the work being done in analogical reasoning elsewhere in AI to the planning field. The goal is to exploit any similarity between problems to speed up solution finding. Current case-based reasoning implementations in planning are capable of recognizing a narrow range of similarities between an archived partial plan and the current state the planner is working from. Such systems cannot apply knowledge learned in one logistics domain, for example, to another system—even though a human would find it natural to use what he/she has learned in solving an AIPS planning competition driver log problem to a depot problem. We note that transproblem learning has been ap-

Reinforcement Learning
The Special Case

In the context of the figure 1 dimensions for a learning-in-planning system, reinforcement learning must be seen as a special case. Unlike the other learning types, this widely studied machine learning field is not readily characterized as a learning technique for augmenting a planning system. Essentially, it's a toss up whether to view reinforcement learning as a learning system that contains a planning subsystem with a learning component. Reinforcement learning is defined more clearly by characterizing a learning problem instead of a learning technique.

A general reinforcement learning problem can be seen as composed of just three elements: (1) goals an agent must achieve, (2) an observable environment, and (3) actions an agent can take to affect the environment (Sutton and Barto 1998). Through trial-and-error online visitation of states in its environment, such a reinforcement learning system seeks to find an optimal policy for achieving the problem goals. When reinforcement learning is applied to a planning problem, a fourth element, the presence of a domain theory, comes into play. The explicit model of the valid operators is used to direct the exploration of the state space, and this space is used (together with the reward associated with each state), in turn, to refine the domain theory. Because, in principle, the "exact domain theory" is never acquired, reinforcement learning has been termed a "lifelong learning process." This aspect stands in sharp contrast to the assumption in classical planning that the planner is provided a complete and perfect domain theory.

Because of the tightly integrated nature of the planning and learning aspects of reinforcement learning, the five-dimensional view of figure 1 is not as useful for characterizing implemented reinforcement learning-planning systems as it is for other learning-augmented planners. Nonetheless, when we analyze the survey results in the next section, we will map planning-oriented reinforcement learning work onto this dimensional structure for purposes of comparison with the other nine learning techniques that have been (or could be) used to augment planning systems.

proached from a somewhat different direction in Fox and Long (1999) using a process of identifying abstract types during domain preprocessing.

Extending Learning to Nonclassical Planning Problems The preponderance of planning research has been based in classical planning, as is borne out by the survey tables and figures. Historically, this weighting arose because of the need to study a less daunting problem than full-scope planning, and much of the progress realized in classical planning has indeed provided the foundation for advances now being made in nonclassical formulations. It is a reasonable expectation that the body of work in learning methods adapted to classical planning will similarly be modified and extended to nonclassical planning systems. With the notable exception of reinforcement learning, the surface has scarcely been scratched in this regard.

If, as we suggest in the introduction, the recent striking advances in speed for state-of-the-art planning systems lies behind the relative paucity of current research in speedup learning, the focus might soon shift back in this direction. These systems, impressive though they are, demonstrated their speedup abilities in classical planning domains. As the research attention shifts to problems beyond the classical paradigm, the greatly increased difficulty of the problems themselves seems likely to renew planning community interest in speed-up learning approaches.

New Avenues for Learning in Planning Motivated by Recent Developments in Planning

Recent advances in planning research suggest several aspects of the new generation of planners for which machine learning methods might provide important enhancements. We discuss here three such avenues for learning in planning: (1) offline learning of domain knowledge, (2) learning to improve heuristics, and (3) learning to improve plan quality.

Offline Learning of Domain Knowledge We have previously noted the high overhead cost of conducting learning online during the course of solving a single problem, relative to often-short solution times for the current generation of fast and efficient planners. This handicap might help explain more recent interest in offline learning, such as domain analysis, which can be reused to advantage over a series of problems within a given domain. The survey results and figure 3 also suggest an area of investigation that has so far been neglected in studies focused on nonclassical planning—the learning of domain invariants before planning starts. This *static analysis* has been shown to be an effective speedup approach for many classical planning domains, and there is no reason to believe it cannot similarly boost nonclassical planning.

On another front, there has been much enthusiasm in parts of the planning community for applying domain-specific knowledge to speed up a given planner (for example, TL PLAN [Bacchus and Kabanza 2000] and BLACKBOX [Kautz and Selman 1998]). This advantage has also been realized in hierarchical task network (HTN) planning systems by supplying domain-specific task-reduction schemas to the planner (SHOP [Nau et al. 1999]). Such leveraging of user-supplied domain knowledge has been shown to greatly decrease planning time for a variety of domains and problems. One drawback of this approach is the burden it places on the user to correctly hand code the domain knowledge ahead of time and in a form usable by the particular planner. Offline learning techniques might be exploited here. If the user provides very high-level domain knowledge in a format readily understandable by humans, the system could learn in supervised fashion to operationalize this background knowledge to the particular formal representation usable by a given target planning system. If the user is not to be burdened with learning the planner's low-level language for knowledge representation, this approach might entail solving sample problems iteratively with combinations of these domain rules to determine both correctness and efficacy.

An interesting related issue is the question of which types of knowledge are easiest and hardest to learn, which has a direct impact on the types of knowledge that might actually be worth learning. The closely related machine learning aspect of *sample complexity* addresses the number and type of examples that are needed to induce a given concept or target function. To date, the relative difficulty of learning tasks has received little attention with respect to the domain-specific knowledge used by some planners. What are the differences in terms of the sample complexity of learning different types of domain-specific control knowledge? For example, it would be worth categorizing the TL PLAN control rules versus the SHOP/HTN–style schemas in terms of their sample complexity.

Learning to Improve Heuristics The credit for both the revival of plan-space planning and the impressive performance of most state-space planners in recent years goes largely to the development of heuristics that guide the planner at key decision points in its search. As such, considerable research effort is focusing on finding more effective domain-independent heuristics and tuning heuristics to particular problems and domains. The role that learning might play in acquiring or refining such heuristics has largely been unexplored. In particular, learning such heuristics inductively during the planning process would seem to hold promise. Generally, the heuristic values are calculated by a linear combination of weighted terms where the designer chooses both the terms and their weights in hopes of obtaining an equation that will be robust across a variety of problems and domains. The *search trace* (states visited) resulting from a problem-solving episode could provide the negative and positive examples needed to train a neural network or learn a decision tree. Possible target functions for inductively learning or improving heuristics include term weights that are most likely to lead to higher-quality solutions for a given domain, term weights that will be most robust across many domains, attributes that are most useful for classifying states, exceptions to an existing heuristic such as used in LRTA* (Korf 1990), and a metalevel function that selects or modifies a search heuristic based on the problem or domain.

Multistrategy learning might also play a role in that the user might provide background knowledge in the form of the base heuristic.

The ever-growing cadre of planning approaches and learning tools, each with their own strengths and weaknesses, suggests another inviting direction for speedup learning. Learning a rule set or heuristic that will direct the application of the most effective approach (or multiple approaches) for a given problem could lead to a metaplanning system with capabilities well beyond any individual planner. Interesting steps in this direction have been taken by Horvitz et al. (2001) using the construction and use of Bayesian models to predict the run time of various problem solvers.

Learning to Improve Plan Quality The survey tables and figures suggest that the issue of improving plan quality using learning has received much less attention in the planning community than speedup learning. However, because planning systems are ported into real-world applications, this concern is likely to be a primary one. Many planning systems that successfully advance into the marketplace will need to interact frequently with human users in ways that have received scant attention in the lab. Such users are likely to have individual biases with respect to plan quality that they can be hard pressed to quantify. These plan-

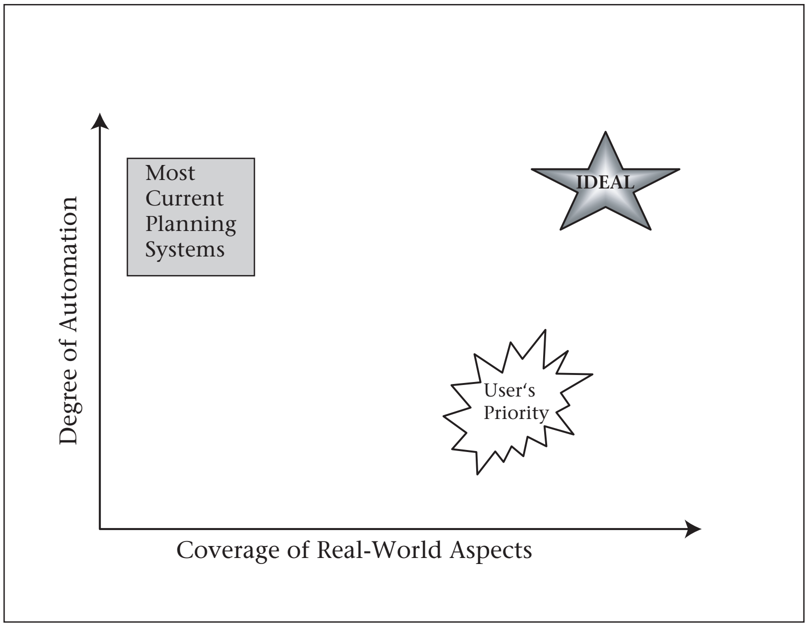

Figure 5. The Coverage versus Automation Trade-Off in Planning Systems.

ning systems could be charted in a 2D space with the following two axes: (1) degree of coverage of the issues confronted in a real-world problem, that is, the capability of the system to deal with all aspects of a problem without abstracting them away and (2) degree of automation, that is, the extent to which the system automatically reasons about the various problem aspects and makes decisions without guidance by the human user.

Figure 5 shows such a chart for current-day planning systems. The ideal planner plotted on this chart would obviously lie in the top-right corner. It is interesting to note that most users—aware that they can't have it all—prefer a system that can, in some sense, handle most aspects of the real-world problem at the expense of full automation. However, most current-day planning systems abstract away large portions of the real-world problem in favor of fully automating what the planner can actually accomplish. In large practical planning environments, fully automated planning is neither feasible nor desirable because users want to observe and control plan generation.

Some planning systems such as HICAP (Munoz-Avila et al. 1999) and ALPS (Calistri-Yeh, Segre, and Sturgill 1996) have made inroads toward building an effective interface with their human users. No significant role for learning has been established yet for such systems, but possibilities include learning user preferences with respect to plan actions, intermediate states, and pathways. Given the human inclination to "have it their way," it might be that the best way to tailor an interactive planner will be in the manner of the programming-by-demonstration systems that have recently received attention in the machine learning community (Lau, Domingos, and Weld 2000). Such a system implemented on top of a planner might entail having the user create plans for several problems that the learning system would then parse to learn plan aspects peculiar to the particular user.

Summary and Conclusion

We have presented the results of an extensive survey of research conducted and published since the first application of learning to automated planning was implemented some 30 years ago. In addition to compiling categorized tables of the corpus of work, we have presented a five-dimensional characterization of learning in planning and mapped the studies onto it. This process has clarified the foci of the work in this area and suggested a number of avenues along which the community might reasonably proceed in the future. It is apparent that automated planning and machine learning are well-matched methodologies in a variety of configurations, and we suggest there are a number of these approaches that merit more research attention than they have received to date. We have expanded on several of these possibilities and offered our conjectures about where the more interesting work might lie.

References

Aler, R., and Borrajo, D. 2002. On Control Knowledge Acquisition by Exploiting Human-Computer Interaction. Paper presented at the Sixth International Conference on Artificial Intelligence Planning and Scheduling. 23–27 April, Toulouse, France.

Aler, R.; Borrajo, D.; and Isasi, P. 1998. Genetic Programming and Deductive-Inductive Learning: A Multistrategy Approach. Paper presented at the Fifteenth International Conference on Machine Learning, ICML '98, 24–27 July, Madison, Wisconsin.

Ambite, J. L.; Knoblock, C. A.; and Minton, S. 2000. Learning Plan Rewriting Rules. Paper presented at the Fifth International Conference on Artificial Intelligence Planning and Scheduling, 14–17 April, Breckenridge, Colorado.

Ashley, K., and Aleven, V. 1997. Reasoning Symbolically about Partially Matched Cases. In Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI-97, 335–341. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Avesani, P.; Perini, A.; and Ricci, F. 2000. Interactive Case-Based Planning for Forest Fire Management. *Applied Intelligence* 13(1): 41–57.

Barto, A.; Bradtke, S.; and Singh, S. 1995. Learning to Act Using Real-Time Dynamic Programming. *Artifi-*

cial Intelligence 72(1): 81–138.

Ashley, K. D., and McLaren, B. 1995. Reasoning with Reasons in Case-Based Comparisons. In *Proceedings of the First International Conference on Cased-Based Reasoning (ICCBR-95),* 133–144. Berlin: Springer.

Bacchus, F., and Kabanza, F. 2000. Using Temporal Logics to Express Search Control Knowledge for Planning. *Artificial Intelligence* 116(1–2): 123–191.

Bennett, S. W., and DeJong, G. F. 1996. Real-World Robotics: Learning to Plan for Robust Execution. *Machine Learning* 23(2–3): 121–161.

Bergmann, R., and Wilke, W. 1996. On the Role of Abstractions in Case-Based Reasoning. In *Proceedings of EWCBR-96, the European Conference on Case-Based Reasoning,* 28–43. New York: Springer.

Bhatnagar, N., and Mostow, J. 1994. Online Learning from Search Failures. *Machine Learning* 15(1): 69–117.

Blum, A., and Furst, M. L. 1997. Fast Planning through Planning Graph Analysis. *Artificial Intelligence* 90(1–2): 281–300.

Borrajo D., and Veloso, M. 1997. Lazy Incremental Learning of Control Knowledge for Efficiently Obtaining Quality Plans. *Artificial Intelligence Review* 11(1–5): 371–405.

Bylander, T. 1992. Complexity Results for Serial Decomposability. In Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92), 729–734. Menlo Park, Calif.: American Association for Artificial Intelligence.

Calistri-Yeh, R.; Segre, A.; and Sturgill, D. 1996. The Peaks and Valleys of ALPS: An Adaptive Learning and Planning System for Transportation Scheduling. Paper presented at the Third International Conference on Artificial Intelligence Planning Systems (AIPS-96), 29–31 May, Edinburgh, United Kingdom.

Carbonell, Y. G., and Gil, Y. 1990. Learning by Experimentation: The Operator Refinement Method. In *Machine Learning: An Artificial Intelligence Approach, Volume 3,* eds. Y. Kodtratoff and R. S. Michalski, 191–213. San Francisco, Calif.: Morgan Kaufmann.

Chien, S. A. 1989. Using and Refining Simplifications: Explanation-Based Learning of Plans in Intractable Domains. In Proceedings of the Eleventh International Joint Conference on Artificial Intelligence, 590–595. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Cohen, W. W. 1990. Learning Approximate Control Rules of High Utility. Paper presented at the Seventh International Conference on Machine Learning, 21–23 June, Austin, Texas.

Cohen, W. W., and Singer, Y. 1999. A Simple, Fast, and Effective Rule Learner. In Proceedings of the Sixteenth National Conference on Artificial Intelligence (AAAI-99), 335–342. Menlo Park, Calif.: American Association for Artificial Intelligence.

Craven, M., and Shavlik, J. 1993. Learning Symbolic Rules Using Artificial Neural Networks. Paper presented at the Tenth International Conference on Machine Learning, 24–27 July, London, United Kingdom.

Dawson, C., and Siklossy, L. 1977. The Role of Preprocessing in Problem-Solving Systems. In Proceedings of the Fifth International Joint Conference on Artificial Intelligence, 465–471. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Dearden, R.; Friedman, N.; and Russell, S. 1998. Bayesian Q-Learning. In Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98), 761–768. Menlo Park, Calif.: American Association for Artificial Intelligence.

Dempster, A. P.; Laird, N. M.; and Rubin, D. B. 1977. Maximum Likelihood from Incomplete Data via the EM Algorithm. *Journal of the Royal Statistical Society* B39(1): 1–38.

Dietterich, T. G., and Flann, N. S. 1997. Explanation-Based Learning and Reinforcement Learning: A Unified View. *Machine Learning* 28:169–210.

Do, B., and Kambhampati, S. 2003. Planning as Constraint Satisfaction: Solving the Planning Graph by Compiling It into a CSP. *Artificial Intelligence* 132: 151–182.

Estlin, T. A., and Mooney, R. J. 1996. Multi-Strategy Learning of Search Control for Partial-Order Planning. In Proceedings of the Thirteenth National Conference on Artificial Intelligence, 843–848. Menlo Park, Calif.: American Association for Artificial Intelligence.

Etzioni, O. 1993. Acquiring Search-Control Knowledge via Static Analysis. *Artificial Intelligence* 62(2): 265–301.

Fikes, R. E., and Nilsson, N .J. 1971. STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving. *Artificial Intelligence* 2(3–4): 189–208.

Fikes, R. E.; Hart, P.; and Nilsson, N. J. 1972. Learning and Executing Generalized Robot Plans. *Artificial Intelligence* 3:251–288.

Fox, M., and Long, D. 1999. The Detection and Exploitation of Symmetry in Planning Problems. Paper presented at the Sixteenth International Joint Conference on Artificial Intelligence, 31 July–6 August, Stockholm, Sweden.

Fox, M., and Long, D. 1998. The Automatic Inference of State Invariants in TIM. *Journal of Artificial Intelligence Research* 9: 317–371.

Fu, L.-M. 1989. Integration of Neural Heuristics into Knowledge-Based Inference. *Connection Science* 1(3): 325–340.

García-Martínez, R., and Borrajo, D. 2000. An Integrated Approach of Learning, Planning, and Execution. *Journal of Intelligent and Robotic Systems* 29(1): 47-78.

Gerevini, A., and Schubert, L. 1998. Inferring State Constraints for Domain-Independent Planning. In Proceedings of the Fifteenth National Conference on Artificial Intelligence, 905–912. Menlo Park, Calif.: American Association for Artificial Intelligence.

Gerevini, A., and Schubert, L. 1996. Accelerating Partial-Order Planners: Some Techniques for Effective Search Control and Pruning. *Journal of Artificial Intelligence Research* 5:95–137.

Gil, Y. 1994. Learning by Experimentation: Incremental Refinement of Incomplete Planning Domains. Paper presented at the Eleventh International Conference on Machine Learning, 10–13 July, New Brunswick, New Jersey.

Gratch, J., and Dejong, G. 1992. COMPOSER: A Probabilistic Solution to the Utility Problem in Speed-Up Learning, In Proceedings of the Tenth National Conference on Artificial Intelligence (AAAI-92), 235–240. Menlo Park, Calif.: American Association for Artificial Intelligence.

Hammond, K. 1989. *Case-Based Planning: Viewing Planning as a Memory Task*. San Diego, Calif.: Academic Press.

Hanks, S., and Weld, D. 1995. A Domain-Independent Algorithm for Plan Adaptation. *Journal of Artificial Intelligence Research* 2:319–360.

Hinton, G. E. 1989. Connectionist Learning Procedures. *Artificial Intelligence* 40(1–3): 185–234.

Hofstadter, D. R., and Marshall, J. B. D. 1993. A Self-Watching Cognitive Architecture of High-Level Perception and Analogy-Making. Technical Report, TR100, Center for Research on Concepts and Cognition, Indiana University.

Hofstadter, D. R., and Marshall, J. B. D. 1996. Beyond Copycat: Incorporating Self-Watching into a Computer Model of High-Level Perception and Analogy Making. Paper presented at the 1996 Midwest Artificial Intelligence and Cognitive Science Conference, 26–28 April, Bloomington, Indiana.

Hollatz, J. 1999. Analogy Making in Legal Reasoning with Neural Networks and Fuzzy Logic. *Artificial Intelligence and Law* 7(2–3): 289–301.

Horvitz, E.; Ruan, Y.; Gomes, C.; Kautz, H.; Selman, B.; and Chickering, D. M. 2001. A Bayesian Approach to Tackling Hard Computational Problems. Paper presented at the Seventeenth Conference on Uncertainty in Artificial Intelligence, 2–5 August, Seattle, Washington.

Huang, Y.; Kautz, H.; and Selman, B. 2000. Learning Declarative Control Rules for Constraint-Based Planning. Paper presented at the Seventeenth International Conference on Machine Learning, 29 June–2 July, Stanford, California.

Hunt, E. B.; Marin, J.; and Stone, P. J. 1966. *Experiments in Induction*. San Diego, Calif.: Academic Press.

Ihrig, L., and Kambhampati, S. 1997. Storing and Indexing Plan Derivations through Explanation-Based Analysis of Retrieval Failures. *Journal of Artificial Intelligence* 7:161–198.

Ihrig, L., and Kambhampati, S. 1996. Design and Implementation of a Replay Framework Based on a Partial-Order Planner. In Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96). Menlo Park, Calif.: American Association for Artificial Intelligence.

Jones, R., and Langley, P. 1995. Retrieval and Learning in Analogical Problem Solving. In Proceedings of the Seventeenth Conference of the Cognitive Science Society, 466–471. Pittsburgh, Pa.: Lawrence Erlbaum.

Kakuta, T.; Haraguchi, M.; Midori-ku, N.; and Okubo, Y. 1997. A Goal-Dependent Abstraction for Legal Reasoning by Analogy. *Artificial Intelligence and Law* 5(1–2): 97–118.

Kambhampati, S. 2000. Planning Graph as (Dynamic) CSP: Exploiting EBL, DDB, and Other CSP Techniques in GRAPHPLAN. *Journal of Artificial Intelligence Research* 12:1–34.

Kambhampati, S. 1998. On the Relations between Intelligent Backtracking and Failure-Driven Explanation-Based Learning in Planning. *Constraint Satisfaction and Artificial Intelligence* 105(1–2): 161–208.

Kambhampati, S., and Hendler, J. 1992. A Validation Structure–Based Theory of Plan Modification and Reuse. *Artificial Intelligence* 55(23): 193–258.

Kambhampati, S., and Katukam, Y. Q. 1996. Failure-Driven Dynamic Search Control for Partial Order Planners: An Explanation-Based Approach. *Artificial Intelligence* 88(1–2): 253–315.

Kautz, H., and Selman, B. 1999. BLACKBOX: Unifying SAT-Based and Graph-Based Planning. In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99), 318–325. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Kautz, H., and Selman, B. 1998. The Role of Domain-Specific Knowledge in the Planning as Satisfiability Framework. Paper presented at the Fifth International Conference on Planning and Scheduling (AIPS-98), 7–10 June, Pittsburgh, Pennsylvania.

Kedar-Cabelli, S., and McCarty, T. 1987. Explanation-Based Generalization as Resolution Theorem Proving. In *Proceedings of the Fourth International Workshop on Machine Learning,* 383–389. San Francisco, Calif.: Morgan Kaufmann.

Khardon, R. 1999. Learning Action Strategies for Planning Domains. *Artificial Intelligence* 113(1–2): 125–148.

Knoblock, C. 1990. Learning Abstraction Hierarchies for Problem Solving. In Proceedings of the Eighth National Conference on Artificial Intelligence, 923–928. Menlo Park, Calif.: American Association for Artificial Intelligence.

Kodtratoff, Y., and Michalski, R. S., eds. 1990. *Machine Learning: An Artificial Intelligence Approach, Volume 3*. San Francisco, Calif.: Morgan Kaufmann.

Korf, R. 1990. Real-Time Heuristic Search. *Artificial Intelligence* 42(2–3): 189–211.

Laird, J.; Newell, A.; and Rosenbloom, P. 1987. SOAR: An Architecture for General Intelligence. *Artificial Intelligence* 33(1): 1–64.

Lang, K. 1995. NEWSWEEDER: Learning to Filter Netnews. In *Proceedings of the Twelfth International Conference on Machine Learning,* 331–339. San Francisco, Calif.: Morgan Kaufmann.

Langley, P. 1997. Challenges for the Application of Machine Learning. In *Proceedings of the ICML '97 Workshop on Machine Learning Application in the Real World: Methodological Aspects and Implications,* 15–18. San Francisco, Calif.: Morgan Kaufmann.

Lau, T.; Domingos, P.; and Weld, D. 2000. Version Space Algebra and Its Application to Programming by Demonstration. Paper presented at the Seventeenth International Conference on Machine Learning, 29 June–2 July, Stanford, California.

Lavrac, N.; Dzeroski, S.; and Grobelnik, M. 1991.

Learning Nonrecursive Definitions of Relations with LINUS. In *Proceedings of the Fifth European Working Session on Learning,* 265–281. Berlin: Springer.

Leake, D.; Kinley, A.; and Wilson, D. 1996. Acquiring Case Adaptation Knowledge: A Hybrid Approach. In Proceedings of the Thirteenth National Conference on Artificial Intelligence, 684–689. Menlo Park, Calif.: American Association for Artificial Intelligence.

Leckie, C., and Zuckerman, I. 1998. Inductive Learning of Search Control Rules for Planning. *Artificial Intelligence* 101(1–2): 63–98.

Martin, M., and Geffner, H. 2000. Learning Generalized Policies in Planning Using Concept Languages. In *Proceedings of the Seventh International Conference on Knowledge Representation and Reasoning (KR 2000)*, 667–677. San Francisco, Calif.: Morgan Kaufmann.

Minton, S., ed. 1993. *Machine Learning Methods for Planning*. San Francisco, Calif.: Morgan Kaufmann.

Minton, S.; Carbonell, J.; Knoblock, C.; Kuokka, D. R.; Etzioni, O.; and Gil, Y. 1989. Explanation-Based Learning: A Problem-Solving Perspective. *Artificial Intelligence* 40:63–118.

Mitchell, T. M., and Thrun, S. B. 1995. Learning Analytically and Inductively. In *Mind Matters: A Tribute to Allen Newell (Carnegie Symposia on Cognition),* eds. J. D. Steier, T. Mitchell, and A. Newell. New York: Lawrence Erlbaum.

Mitchell, T.; Keller, R.; and Kedar-Cabelli, S. 1986. Explanation-Based Generalization: A Unifying View. *Machine Learning* 1(1): 47–80.

Muggleton, S., and Feng, C. 1990. Efficient Induction of Logic Programs. Paper presented at the First Conference on Algorithmic Learning Theory, 8–10 October, Ohmsma, Tokyo, Japan.

Munoz-Avila, H.; Aha, D. W.; Breslow, L.; and Nau, D. 1999. HICAP: An Interactive Case-Based Planning Architecture and Its Application to Noncombatant Evacuation Operations. In Proceedings of the Ninth Conference on Innovative Applications of Artificial Intelligence, 879–885. Menlo Park, Calif.: American Association for Artificial Intelligence.

Nau, D.; Cao, Y.; Lotem, A.; and Munoz-Avila, H. 1999. SHOP: Simple Hierarchical Order Planner. In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence (IJCAI-99), 968–975. Menlo Park, Calif.: International Joint Conference on Artificial Intelligence.

Nebel, B.; Dimopoulos, Y.; and Koehler, J. 1997. Ignoring Irrelevant Facts and Operators in Plan Generation. Paper presented at the Fourth European Conference on Planning (ECP-97), 24–26 September, Toulouse, France.

Ourston, D., and Mooney, R. 1994. Theory Refinement Combining Analytical and Empirical Methods. *Artificial Intelligence* 66(2): 273–309.

Pazzani, M. J.; Brunk, C. A.; and Silverstein, G. 1991. A Knowledge-Intensive Approach to Learning Relational Concepts. Paper presented at the Eighth International Workshop on Machine Learning, 27–29 June, Evanston, Illinois.

Perez, M., and Etzioni, O. 1992. DYNAMIC: A New Role for Training Problems in EBL. Paper presented at the Ninth International Conference on Machine Learning, 27–30 June, Bled, Slovenia.

Pomerleau, D. A. 1993. Knowledge-Based Training of Artificial Neural Networks for Autonomous Robot Driving. In *Robot Learning,* eds. J. Connell and S. Mahadevan, 19–43. Boston: Kluwer Academic.

Quinlan, J. R. 1993. *C4.5: Programs for Machine Learning*. San Francisco, Calif.:. Morgan Kaufmann.

Quinlan, J. R. 1990. Learning Logical Definitions from Relations. *Machine Learning* 5:239–266.

Quinlan, J. R. 1986. Induction of Decision Trees. *Machine Learning* 1(1): 81–106.

Reddy, C., and Tadepalli, P. 1999. Learning Horn Definitions: Theory and an Application to Planning. *New Generation Computing* 17(1): 77–98.

Rintanen, J. 2000. An Iterative Algorithm for Synthesizing Invariants. In Proceedings of the Seventeenth National Conference on Artificial Intelligence and the Twelfth Innovative Applications of AI Conference, 806-811. Menlo Park, Calif.: American Association for Artificial Intelligence.

Sacerdoti, E. 1974. Planning in a Hierarchy of Abstraction Spaces. *Artificial Intelligence* 5(2): 115–135.

Samuel, A. L. 1959. Some Studies in Machine Learning Using the Game of Checkers. *IBM Journal of Research and Development* 3(3): 210–229.

Schmill, M.; Oates, T.; and Cohen, P. 2000. Learning Planning Operators in Real-World, Partially Observable Environments. Paper presented at the Fifth Conference on Artificial Intelligence Planning Systems (AIPS-2000), 14–17 April, Breckenridge, Colorado.

Shavlik, J. W., and Towell, G. G. 1989. An Approach to Combining Explanation-Based and Neural Learning Algorithms. *Connection Science* 1(3): 231–253.

Sheppard, J., and Salzberg, S. 1995. Combining Genetic Algorithms with Memory-Based Reasoning. Paper presented at the Sixth International Conference on Genetic Algorithms, 15–19 July, Pittsburgh, Pennsylvania.

Smith, D., and Peot, M. 1993. Postponing Threats in Partial-Order Planning. In Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93), 500–506. Menlo Park, Calif.: American Association for Artificial Intelligence.

Sutton, R. 1991. Planning by Incremental Dynamic Programming. In *Proceedings of the Eighth International Conference on Machine Learning,* 353–357. San Francisco, Calif.: Morgan Kaufmann.

Sutton, R. 1988. Learning to Predict by the Methods of Temporal Differences. *Machine Learning* 3(1): 9–44.

Sutton, R., and Barto, G. 1998. *Reinforcement Learning—-An Introduction.* Cambridge, Mass.: MIT Press.

Sycara, K.; Guttal, R.; Koning, J.; Narasimhan, S.; and Navinchandra, D. 1992. CADET: A Case-Based Synthesis Tool for Engineering Design. *International Journal of Expert Systems* 4(2).

Veloso, M., and Carbonell, J. 1993. Derivational Analogy in PRODIGY: Automating Case Acquisition, Storage, and Utilization. *Machine Learning* 10(3): 249–278.

Wang, X. 1996a. A Multistrategy Learning System for Planning Operator Acquisition. Paper presented at

the Third International Workshop on Multistrategy Learning, 23–25 May, Harpers Ferry, West Virginia.

Wang, X. 1996b. Planning While Learning Operators. Paper presented at the Third International Conference on Artificial Intelligence Planning Systems (AIPS'96), 29–31 May, Edinburgh, Scotland.

Watkins, C. 1989. Learning from Delayed Rewards. Ph.D. dissertation, Psychology Department, University of Cambridge, Cambridge, United Kingdom.

Wolfman, S., and Weld, D. 1999. The LPSAT Engine and Its Application to Resource Planning. In Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, 310–317. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Zelle, J., and Mooney, R. 1993. Combining FOIL and EBG to Speed Up Logic Programs. In Proceedings of the Thirteenth International Joint Conference on Artificial Intelligence, 1106–1111. Menlo Park, Calif.: International Joint Conferences on Artificial Intelligence.

Zimmerman, T., and Kambhampati, S. 2002. Generating Parallel Plans Satisfying Multiple Criteria in Anytime Fashion. Paper presented at the Sixth International Conference on Artificial Intelligence Planning and Scheduling Workshop on Planning and Scheduling with Multiple Criteria, 23–27 April, Toulouse, France.

Zimmerman, T., and Kambhampati, S. 1999. Exploiting Symmetry in the Planning Graph via Explanation-Guided Search. Paper presented at the Sixteenth National Conference on Artificial Intelligence, 18–22 July, Orlando, Florida.

Zweben, M.; Davis, E.; Daun, B.; Drascher, E.; Deale, M.; and Eskey, M. 1992. Learning to Improve Constraint-Based Scheduling. *Artificial Intelligence* 58(1–3): 271–296.

Terry Zimmerman has an M.S. in nuclear science and engineering and is in the throes of completing his PhD. in computer science and engineering at Arizona State University. His dissertation study in the area of automated planning combines aspects of heuristic search control, learning, and optimization over multiple-quality criteria. He previously conducted probabilistic risk assessment and reliability analysis for energy facilities and developed software for statistical analysis of experimental nuclear fuel assemblies. His e-mail address is zim@asu.edu.

Subbarao Kambhampati is a professor of computer science and engineering at Arizona State University, where he directs the Yochan research group. His current research interests are in automated planning, scheduling, learning, and information integration. He received his formative education in Peddapuram, B.Tech from the Indian University of Technology, Madras (Chennai), and his M.S. and Ph.D. from the University of Maryland at College Park. His e-mail address is rao@asu.edu.