

Learning-assisted automated planning: Looking back, taking stock, going forward

Terry Zimmerman & Subbarao Kambhampati
Department of Computer Science & Engineering
Arizona State University, Tempe AZ 85287
Email: {zim, rao}@asu.edu

This paper reports on an extensive survey and analysis of research work related to machine learning as applied to automated planning over the past 30 years. Major research contributions are characterized broadly by learning method and then into descriptive subcategories. Survey results reveal learning techniques that have been extensively 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; *planning* -problem solving in which one uses beliefs about actions and their consequences to construct a sequence of actions that achieve one's goals, and *learning* -using past experience and precepts to improve one's ability to act in the future. Within the A.I. 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, while 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 from 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 employ both tables and graphical maps of existing studies to spotlight to 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-90's 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; "Blocksworld" problems of 10 blocks lay beyond their capabilities as did most logistics problems. (See for example, the texts; *Machine learning methods for planning*. 1993, Minton, S. and *Machine Learning: An artificial intelligence approach*. Vol 3, 1990, Kodratoff, Y. & Michalski, R.S.) The planners of the period employed 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 (Minton, et. al. 1989, Leckie, Zukerman 1998, Veloso, Carbonell 1993, Kambhampati, et al. 1996).

With the advent of several new genres of planning systems in the past 5 - 6 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, as the overhead cost incurred by the learning system can overwhelm the gains in search efficiency. This, in part may explain why the planning community appears to have paid less attention to learning in recent years. From the machine learning community perspective, (Langley 1997) remarked on the swell of research in learning for problem solving and planning that took place in

the 1980's, as well as to note the subsequent tail-off; "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 comes into focus. These range from dealing with incomplete and uncertain environments to developing an effective interface with human users.

Our purpose in this study is to develop, via an extensive survey of published work, a broad perspective of the diverse research that has been conducted to date in learning-in-planning, and then to conjecture as to profitable directions for future work in this area. The remainder of the article is organized into three parts; *where* learning is likely to be of assistance in automated planning, *what* roles has learning actually played in the relevant planning research conducted to date, and *where* might the research community gainfully direct its attentions in the near future. In the next section, 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 possibly oversights. This 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 May 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), so in order 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 it's subsequent search accordingly offers a means of reducing this complexity. Learning seems to have an obvious role in "mixed initiative" planning where it may 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 will 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/Execution phase in which learning is conducted
5. Type of learning method

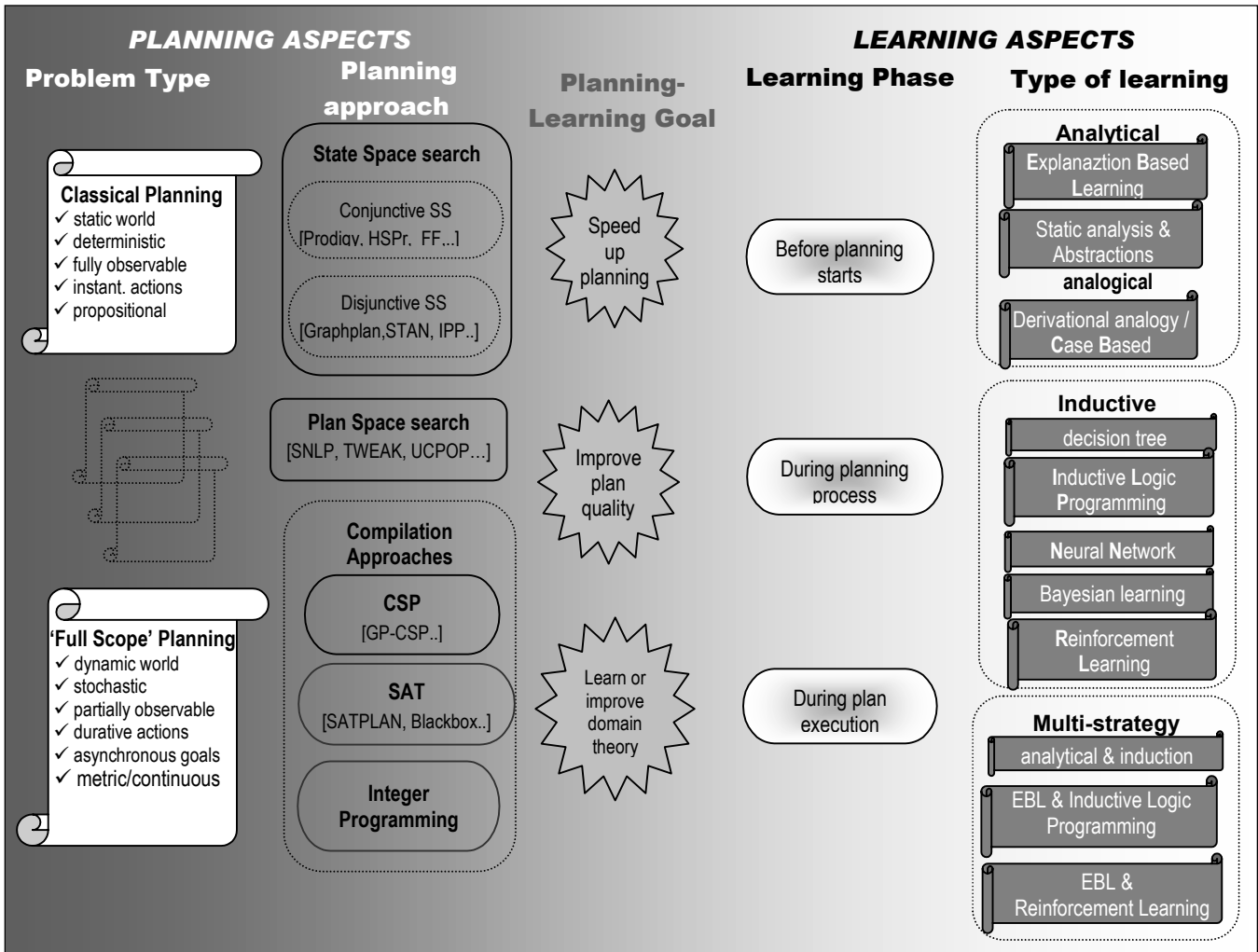


Figure 1. Five dimensions characterizing automated planning systems augmented with a learning component

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, nor do they comprise 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’ (e.g. conditional, conformant, serial or parallel actions), ‘type of knowledge learned’ (domain or search control), ‘learning impetus’ (data-driven or knowledge-driven), and whether plans are hierarchically organized 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 problems 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 (e.g. metric), a fluent might change its value spontaneously or for reasons other than agent actions, 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 (e.g. playing poker), and classical planning where actions realistically require significant periods of time to execute (e.g. logistics domains). The difficulty of even the classical planning problem is such that it largely occupied the full attention of the research community up until the past few years. The current extension into various ‘neo-classical’, 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 sub-field 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. Amongst planners most commonly characterized by search mode, there are two broad categories: search in state space and 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 (though 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: satisfiability (SAT), constraint satisfaction problems (CSP), and integer linear programming (IP). Figure 1 lists these four different methods along with representative planning systems for each. These categories are not entirely disjoint for purposes of classifying planners as some systems employ 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 dynamic CSP, or as conducting “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 at, 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; learning conducted to speed up planning, learning to elicit or improve the planning domain theory, or learning to improve the quality of the plans produced (where “quality” may have a wide range of definitions).

Learning / 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 non-classical planning problem), there are obvious advantages to a planner that can evolve its domain theory via 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 (i.e. too often) go astray. Each such instance, appropriately fed back to the planner, provides a learning opportunity for evolving the domain theory towards 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, she will

likely be faced with tradeoffs 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/or to 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 towards 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:

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 to improve the quality of plans generated. The learning process itself may be either on or offline.

Learning during 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 may be either on 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 *analytical* methods. In Figure 1, we have broadly partitioned the machine learning techniques dimension into these two categories along with a multi-strategy 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.
- Analytical 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/disadvantages of applying a given machine learning technique to a given planning system may help to make sense of any research bias that becomes apparent in the survey tables. The primary types of analytical 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, nor do they nicely partition the entire class (inductive or analytical).

Analytical 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 & 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 may be costly

Table 1 Characterization of the most common analytical learning techniques

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, & vector-valued functions	Robust to noisy & complex data, errors in data	Long training times are common. Learned target function is largely inscrutable
Inductive Logic Programming	1 st 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 may 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 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 non-optimal training sets, facilitates life-long learning	Depends on a real-valued reward signal for each transition. Difficulty handling large state spaces. Convergence can be <i>slow</i> , Space requirements can be huge

Table 2. Characterization of the most common inductive learning techniques

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

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 on this page.

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 formalism. More flexible and powerful forms of analogy can be envisioned (c.f. Hofstadter, Marshall 1993, ‘96), 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 of Tables 1 & 2 give a sense of the degree to which the method may be effective when applied to a given learning problem; in our case, automated planning. Two columns summarize the relative strengths and weaknesses of each 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 tradeoff. A very expressive representation that allows the target function to be represented as close as possible will also require more training data in order to choose among the alternative hypotheses it can represent.

The heart of the learning problem is how to successfully generalize from examples. Analytical learning leans on the learner’s background knowledge to analyze a given training instance so as 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 may prove useful. One possible motivation for adopting a *multi-strategy* approach is that analytical learning methods generate logically justified hypotheses while inductive methods generate statistically justified hypotheses. The logical justifications fall short when the prior knowledge is flawed while the statistical justifications are suspect when data is 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 and described above.

Reinforcement learning, the special case:

In the context of the Figure 1 dimensions for a learning-in-planning system, reinforcement learning (RL) 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 RL as a learning system that contains a planning subsystem or a planning system with a learning component. Reinforcement learning is defined more clearly by characterizing a learning *problem* instead of a learning technique.

A *general* RL problem may be seen as comprised of just three elements; goals an agent must achieve, an observable environment, and actions an agent can take to affect the environment (Sutton, Barto 1998). Through trial-and-error online visitation of states in its environment, such an RL 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 exploration is used (together with the reward associated with each state), in turn, to *refine* the domain theory. Since, in principle, the “exact domain theory” is never acquired, reinforcement learning has been termed a “lifelong learning” process. This stands in sharp contrast to the assumption in classical planning that the planner is provided a complete and perfect domain theory.

Due to the tightly integrated nature of the planning and learning aspects of RL, the 5-dimensional view of Figure 1 is not as useful for characterizing implemented RL-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 RL 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.

¹ For example, “dependency directed backtracking (backjumping)”, a technique closely related to EBL in CSP methods is not tracked in this survey.

What Role has Learning Played in Planning?

We report here the results of an extensive survey of AI research literature² focused on applications of machine learning techniques to planning. Research in the area of machine learning goes back at least as far as 1959, with Arthur Samuel's checkers playing program that improved its performance through learning (Samuel, 1959). 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 et.al., 1972). Space considerations preclude an all-inclusive survey for this 30-year span, of course, but we sought 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 format in a manner such that the five dimensions represented in Figure 1 are usefully reflected. We have employed three different formats, emphasizing different combinations and orderings of the Figure 1 dimensions;

1. A set of three tables organized around just two dimension; type of learning and type of planning.
2. A set of tables reflecting all five dimensions for each relevant study in the survey.
3. A graphical representation providing a visual mapping of the studies' demographics along the five dimensions

We discuss each of these representations next.

Survey tables according to Learning Type / Planning Type

Table 3A deals with studies focused primarily on analytical (deductive) learning in its various forms while Table 3B is concerned with inductive learning. Table 3C addresses studies and multi-strategy systems that aim at some combination of analytical and inductive techniques. All studies/publications appearing in these tables are listed in full in the references section.

The table rows feature the major learning types outlined in Tables 1 & 2, occasionally further subdivided as indicated in the leftmost column. The 2nd column contains a listing of some of the more important *non-planning* 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 last three columns of each table indicate which techniques have been applied in automated planning – subdivided into 'state space', 'plan space', and CSP/SAT/IP planning. Studies dealing with planning problems beyond classical planning (as defined above) appear in shaded blocks in these tables.

Table 3C, covering multi-strategy learning, reflects the fact that the particular combination of techniques employed in some studies could not always be easily subcategorized relative to the analytical 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, Veloso 1997) which exploits the search tree produced by the PRODIGY 4.0 planning system to lazily learn search control heuristics and EGBG, PEGG (Zimmerman, Kambhampati 1999, 2002) 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 "analytical and inductive".

² The major sources employed in the search included AAAI proceedings (1980 - 2000), IJCAI proceedings ('89,'91,'93,'95,'97,'99,'01), Artificial Intelligence Planning and Scheduling proceedings (AIPS -1994, '96, '98, '00, '02), European Conference on Planning proceedings (ECP 1997, '99, '01), Third International Conference on Multistrategy Learning, 1996, International Conference of Machine Learning proceedings (ICML -1991, '94, '96 - '00), Journal of Artificial Intelligence Research (JAIR, 1993 - Jan. 2001), Artificial Intelligence, Elsevier (1987 - 2001), Kluwer journals: AI Review ('96 - '00), Machine Learning ('94 - Mar. 2001), Applied Intelligence ('95 - Mar. 2001), Artificial Intelligence and Law ('95 - Sep. 2000)

ANALYTICAL LEARNING	General Applications	Planning Applications		
		State Space [Conjunctive / Disjunctive]	Plan Space	Compilation [CSP / SAT / IP]
Static / Domain analysis & Abstractions		<i>Learning abstractions:</i> Sacerdoti, '74 ABSTRIPS Knoblock '90 ALPINE <i>Static analysis, domain invars:</i> Etzioni '93 STATIC [-PRODIGY] Perez, Etzioni '92 (<i>w/EBL</i>) DYNAMIC [-PRODIGY] Nebel, Koehler, Dimopoulos '97 RIFO Gerevini, Schubert '98 DISCOPLAN Fox, Long '98, '99 STAN/ TIM [-Graphplan] Rintanen '00	Smith, Peot '93 [-SNLP] Gerevini, Schubert '96 [-UCPOP]	
Explanation Based Learning	<i>General problem solving; 'chunking':</i> Laird et al. '87 SOAR <i>Horn clause rules:</i> Kedar-Cabelli '87 Prolog-EBG <i>Symbolic integration</i> Mitchell, et. al. '86 LEX-2 <see Multi-strategy>	Fikes, Nilsson '72 STRIPS Minton, et. al. '89 PRODIGY Gratch, DeJong '92 COMPOSER [-PRODIGY] Bhatnagar '94. FAILSAFE Borrajo, Veloso '97 <see Multi-strategy> Kambhampati '00 "Graphplan-EBL" <hr/> <i>Permissive real world plans:</i> Bennett, DeJong '96 GRASPER	Chien, '89 Kambhampati, et al. '96 UCPOP-EBL	Wolfman, Weld '99 LPSAT [-RELSAT] <i>Nogood learning:</i> Selman, Kautz '99 BLACKBOX (using RELSAT) Do, Kambhampati '01 GP-CSP [-Graphplan]
Analogical Case-Based Reasoning (derivational & transformational analogy)	Jones, Langley '95 EUREKA <i>Microdomain analogy maker:</i> Hofstadter, Marshall '93 '96 COPYCAT <i>Conceptual design :</i> Sycara, et al '92 CADET <i>Legal reasoning by analogy</i> Ashley, McLaren '95 TRUTH-TELLER Ashley, Alevan '97 CATO Kakuta, et al. '97	<i>transformational:</i> Hammond '89 CHEF Kambhampati, Hendler '92 PRIAR Hanks, Weld '95 SPA Leake, Kinley, Wilson '96 <see Multi-strategy> derivational: Veloso, Carbonell '93 PRODIGY / ANALOGY <i>Learning various abstraction level cases</i> Bergmann, Wilke '96 PARIS <i>User assist planning:</i> Avesani, et. al. '00 CHARADE	<i>derivational:</i> Ihrig, Kambhampati '96 [-UCPOP] <i>with EBL...</i> Ihrig, Kambhampati '97 [-UCPOP]	

Table 3A. Analytical learning applications and studies

Studies in diagonal hashed blocks concern planners applied to problems beyond classical planning.

Implemented system/program names capitalized, Underlying planners & learning subsystems appear in [-]

INDUCTIVE LEARNING	General Applications	Planning Applications		
		State Space [Conjunctive / Disjunctive]	Plan Space	Compilation [CSP / SAT/ IP]
Propositional Decision Trees	<i>Concept learning:</i> Hunt et. al. '66 CLS <i>General DT learning:</i> Quinlan, '86 ID3 Quinlan, '93 C4.5 Khardon, '99 L2ACT Cohen, Singer '99 SLIPPER	<i>Learning operators for real world robotics, clustering:</i> Schmill, Oates, Cohen '00 [-TBA for inducing decision tree]		
Real Valued Neural Network	Hinton, 1989 <i>Symbolic rules from NN:</i> Craven, Shavlik '93 <i>Reflex/Reactive</i> Pomerleau, '93 ALVINN			
1st-Order Logic ILP (Inductive Logic Programming)	<i>Horn-like clauses:</i> Quinlan '90 FOIL Muggleton, Feng '90 GOLEM Lavrac, et al. 1991 LINUS	Leckie, Zukerman '98 GRASSHOPPER [-PRODIGY] Zelle, Mooney '93 <see Multi-strategy> Reddy, Tadepalli '99 ExEL	Estlin, Mooney '96 <see Multi-strategy>	Huang, Selman, Kautz '00 <see Multi-strategy>
Bayesian Learning	<i>Train Bayesian belief networks, unobserved variables:</i> Dempster, et al. '77 EM <i>Text classification:</i> Lang '95 "NewsWeeder" <i>Predict run time of problem solvers for decision-theoretic control</i> Horvitz, et. al. '01			
Other Inductive Learning		<i>Action strategies & Rivest's decision list learning:</i> Khardon '99 Martin, Geffner '00 <i>Plan rewriting:</i> Ambite, Knoblock, Minton '00 PbR		
Reinforcement Learning (RL)	Sutton, '88 TD[lambda] Watkins, '89 "Q learning" Barto, Bradtke, Singh '95 "Real-time dynamic programming" Dearden, Friedman, Russel '98 "Bayesian Q learning"	Dietterich, Flann '95 <see Multi-strategy> <i>Incremental dynamic prog:</i> Sutton, '91 DYNA <i>Planning w/ learned operators:</i> Garcia-Martinez, Borrajo '00 LOPE		

Table 3B. Inductive learning applications and studies

Implemented system/program names capitalized or in double quotes, Underlying planners & subsystems appear in [-]
Studies in diagonal hashed blocks feature planners applied to problems *beyond classical planning*.

MULTI-STRATEGY LEARNING	General Applications	Planning Applications		
		State Space [Conjunctive/Disjunctive]	Plan Space	Compilation [CSP / SAT/ IP]
Analytical & Inductive	<i>Symbolic integration:</i> Mitchell, Keller, Kedar-Cabelli '86 LEX-2 <i>Learn CSP variable ordering:</i> Zweban, et.al. '92 GERRY <i>Incorp. symbolic knowledge in NNs</i> Shavlik, Towell '89 KBANN Fu '89 <i>Learn Horn clause sets focused by domain theory:</i> Pazzani '91 FOCL <i>Refining domain theories using empirical data:</i> Ourston, Mooney '94 EITHER <i>VN and fuzzy logic to implement analogy:</i> Hollatz '99 <i>Genetic, lazy RL, k-Nearest Neighbor:</i> Sheppard, Salzberg '95	<i>Learn / refine operators:</i> Carbonell, Gil '90, Gil '94 EXPO [-PRODIGY] Wang '96a, '96b OBSERVER [-PRODIGY] <hr/> <i>EBL & induction:</i> Calistri-Yeh, Segre, Sturgill '96 ALPS <hr/> <i>CBR and induction:</i> Leake, Kinley, Wilson '96 DIAL Borrajo, Veloso '97 HAMLET [- PRODIGY] Zimmerman, Kambhampati '99, '02 EGBG, PEGG [-Graphplan] <i>Deduct, induct, & genetic:</i> Aler, Borrajo, Isasi '98 HAMLET-EvoCK [-PRODIGY] Aler, Borrajo '02 HAMLET-EvoCK [-PRODIGY]		
EBL & NN	<i>Domain theory cast in neural network form:</i> Mitchell, Thrun '95 EBNN			
EBL & ILP	<i>Search control for logic programs</i> Cohen, '90 AxA-EBL Zelle, Mooney '93 DOLPHIN [-FOIL/PRODIGY]	Zelle, Mooney '93 DOLPHIN [-PRODIGY/FOIL]	Estlin, Mooney '96 SCOPE [-FOIL]	<i>EBL, ILP, & some static analysis:</i> Huang, Selman, Kautz '00 [-BLACKBOX/-FOIL]
EBL & RL		Dietterich, Flann '97 EBRL policies		

Table 3C. Multi-strategy learning applications and studies

EBL: Explanation Based Learning NN: Neural Network ILP: Inductive Logic Programming RL: Reinforcement Learning
 Implemented system/program names capitalized, Underlying planners & learning subsystems appear in [-]
 Studies in diagonal hashed blocks feature planners applied to problems *beyond classical planning*.

In addition to classifying the studies surveyed along the learning-type / planning-type dimensions, these tables illustrate several foci of this corpus of work. For example, the preponderance of research in analytical learning assists to planning as compared to 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 of 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				PLANNING SYSTEMS / STUDIES	
Planning/ Learning Goal	Learning Phase	Type of Learning	Planning Approach		
Speedup	Before planning starts	<i>Analytical</i> Static analysis	Plan Space	Smith, Peot '93 [-SNLP] Gerevini, Schubert '96 [-UCPOP]	
			State Space	Etzioni, '93 STATIC [-PRODIGY] Nebel, Koehler, Dimopoulos '97 RIFO Fox, Long '98, '99 STAN / TIM [-Graphplan] Rintanen '00	
		Static analysis: learn abstractions	.	Sacerdoti, '74 ABSTRIPS Knoblock '90 ALPINE [-PRODIGY]	
	Before & during planning	Static analysis & EBL	.	Perez, Etzioni, '92 DYNAMIC [-PRODIGY]	
	During planning	<i>Analytical</i> EBL	State Space	Fikes, Nilsson '72 STRIPS Minton '89 PRODIGY/EBL Gratch, DeJong '92 COMPOSER [-PRODIGY] Bhatnagar '94. FAILSAFE Kambhampati '00 "Graphplan-EBL"	
			Plan Space	Chien, '89 Kambhampati, et al. '96 UCPOP-EBL	
			(Compilation) SAT	<i>Nogood learning:</i> Kautz, Selman '99 BLACKBOX	
			LP & SAT	Wolfman, Weld '99 LPSAT [-RELSAT]	
		CSP	<i>Nogood learning:</i> Do, Kambhampati '01 GP-CSP [-Graphplan]		
		Analogical Case-based Reasoning	State Space	.	<i>Learning various abstraction level cases</i> Bergmann, Wilke '96 PARIS
				.	<i>User assist planning:</i> Avesani, et. al. '00 CHARADE <i>transformational analogy / adaptation:</i> Hammond '89 CHEF Kambhampati, Hendler '92 PRIAR Hanks, Weld '95 SPA Leake, Kinley, Wilson '96 DIAL <i>derivational analogy / adaptation:</i> Velo, Carbonell '93 PRODIGY/ANALOGY
			Plan Space	Ihrig, Kambhampati '96 [-UCPOP] <i>With EBL...</i> Ihrig, Kambhampati '97 [-UCPOP]	

Table 4A. Survey studies mapped across all five dimensions

Studies in diagonal hashed blocks feature planners applied to problems *beyond classical planning*.

Implemented system/program names capitalized, Underlying planners & learning subsystems appear in [-]

DIMENSIONS				PLANNING SYSTEMS / STUDIES	
Planning/ Learning Goal	Learning Phase	Type of Learning	Planning Approach		
Speedup	During planning	<i>Inductive:</i> Inductive Logic Programming	State Space	Leckie, Zukerman '98 GRASSHOPPER [-PRODIGY] Reddy, Tadepalli '99 ExEL	
		Other induction	.	<i>Action strategies & Rivest's decision list learning of policies:</i> Khardon '99 Martin, Geffner '00	
		Multi-Strategy	.	Calistri-Yeh, Segre, Sturgill '96 ALPS	
		Analytical & Inductive	State Space	<i>CBR and induction:</i> Leake, Kinley, Wilson '96 DIAL Zimmerman, Kambhampati '99 EGBG [-Graphplan]	
	Before & During Planning	EBL, ILP, & static analysis	(<i>Compilation</i>) SAT	Huang, Selman, Kautz '00 [-BLACKBOX/-FOIL]	
			EBL & ILP	State Space	Zelle, Mooney '93 DOLPHIN [-PRODIGY/FOIL]
				Plan Space	Estlin, Mooney '96 SCOPE [-FOIL]
Speedup & Improve Plan Quality	During planning	EBL & Inductive	State Space	Borrajo, Veloso '97 HAMLET [-PRODIGY] <i>Deductive-Inductive & Genetic:</i> Aler, Borrajo '98, '02 HAMLET-EvoCK [-PRODIGY] Zimmerman, Kambhampati '02 PEGG [-Graphplan]	
		<i>Inductive:</i> (analysis of plan differences)	.	<i>Plan rewriting:</i> Ambite, Knoblock, Minton '00 PbR	
Learn or improve domain theory	Before planning starts	(Propositional) Decision Trees	State Space	<i>Learning operators for real world robotics, clustering:</i> Schmill, Oates, Cohen '00 [-TBA for inducing decision tree]	
	During plan execution	Analytical: EBL	.	<i>Permissive real world plans:</i> Bennett, DeJong '96 GRASPER	
		Multi-Strategy: analytical & inductive	.	<i>Learning / refining operators:</i> Wang '96a, '96b OBSERVER [-PRODIGY] Carbonell, Gil '90, Gil '94 EXPO [-PRODIGY]	
Learn or improve domain theory & Improve Plan Quality	During planning	EBL & RL	"State Space"	Dietterich, Flann '97 EBRL	
		<i>Inductive:</i> Reinforcement Learning	.	<i>Incremental dynamic prog:</i> Sutton, '91 DYNA <i>Planning with learned operators:</i> Garcia-Martinez, Borrajo '00 LOPE	

Table 4B. Survey studies mapped across all five dimensions

Studies in diagonal hashed blocks feature planners applied to problems beyond classical planning.

Implemented system/program names capitalized, Underlying planners & learning subsystems appear in [-]

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 to the aspects of learning to improve plan quality or building/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.

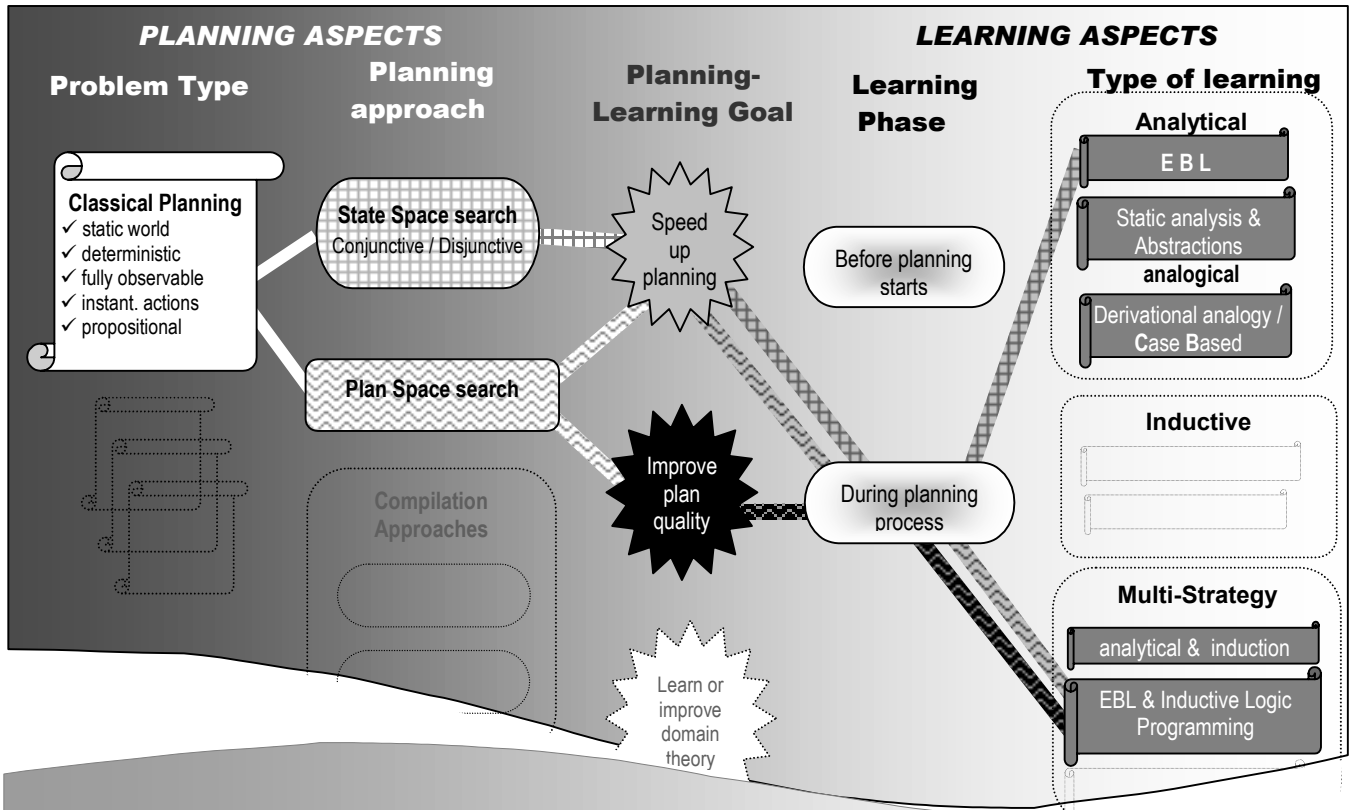


Figure 2. Example graphical mapping of two learning-in-planning systems

The layout of the five dimensions of 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 cross-hatched lines; it’s a classical planning system that conducts state-space search, and it aims to speed up planning via explanation based learning during the planning process. Tracing the subgraph from the left, the edge picks up the cross-hatch at the ‘State Space’ node, and the gray shade of the ‘Speed up planning’ node. The SCOPE system (Estlin, Mooney 1996) is then represented as the branched series of wavy patterned lines; it’s a classical planning system that conducts plan space search, and the goal of it’s learning subsystem is to *both* speedup planning and improve plan quality. Thus, the ‘Plan Space’ node branches to both of these ‘Planning-Learning Goal’ nodes. All of SCOPE’s learning occurs during the planning process, employing both EBL and inductive logic programming. As such, the edges converge at the ‘During planning process’ node, but both edges persist to connect with the EBL & ILP node.

Graphical analysis of survey

There are obvious limitations to what can be readily gleaned from any tabular presentation of a data set across more than 2-3 dimensions. In order to more easily visualize patterns and relationships in learning-in-planning work, we have devised a graphical method of depicting the corpus of work in this survey with respect to the five dimensions of 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 either Figure 3 (classical planning systems) or Figure 4 (systems designed to handle problems beyond the classical paradigm)³. 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 graphical presentation mode permits quick identification of all planner-learning system configurations that embody any of the aspects of the five dimensions (nodes). For example, since 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. This 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 graphical figure alone. However, the tables above can assist with in this regard.

Learning within the classical planning framework:

Figure 3 indicates via dashed lines and fading, those aspects (nodes) of the five dimensions of learning-in-planning that are not relevant to classical planning. Specifically, 'learning / 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 due to efficiency considerations it has remained primarily of 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 (see Fox, Long 1998, '99, Gerevini, 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 as this mode has clear affinity with improving a faulty domain theory—a non-classical problem.

It is apparent, based on the Figure 3 graph in combination with the survey tables, that EBL has been extensively studied and applied to every planning approach and both relevant planning-learning goals. This is perhaps not surprising, given the presence of the sort of domain theory that EBL can readily exploit. Perhaps more notable is the scant attention paid to inductive learning techniques for classical planners. Although inductive logic programming (ILP) has been extensively 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 non-classical 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-C, 4A, and 4B). There are, as yet, far fewer learning-augmented systems, although this a growing area of planning community interest. Those that exist extend the classical planning problem in a variety of different ways, but due to space considerations, we have not reflected this with separate versions of Figure 4 for each combination. This is the natural domain of reinforcement learning systems, and as discussed above, this popular machine learning field does not so readily fit our 5-dimensional learning-in-planning perspective. Figure 4 therefore represents RL in a different manner than the other approaches; a single shade, brick cross-hatch set of edges is used to span the five dimensions. The great majority of RL systems to date adopt a state space perspective so there is an edge skirting this node. With respect to the planning-learning goal dimension, RL can be viewed as both 'improving plan quality' (the process moves toward the optimal policy) and 'learning the

³ For this survey the classical/non-classical partition of a system is drawn based on the applications made by it's designers in the survey reference, regardless of what class of problems it might conceivably handle.

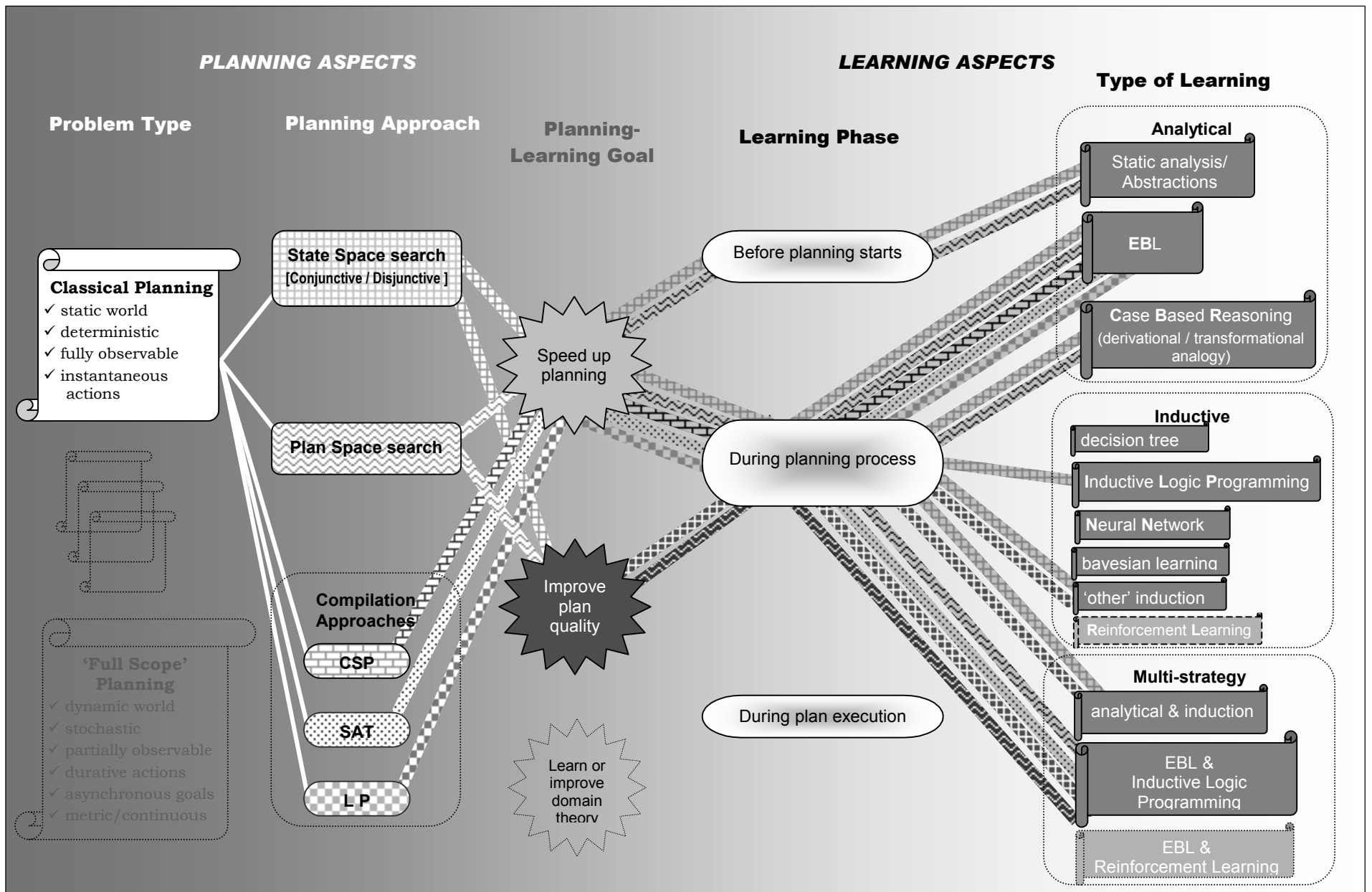


Figure 3: Mapping of the survey planning-learning systems for *classical planning* problems onto the Figure 1 characterization structure.

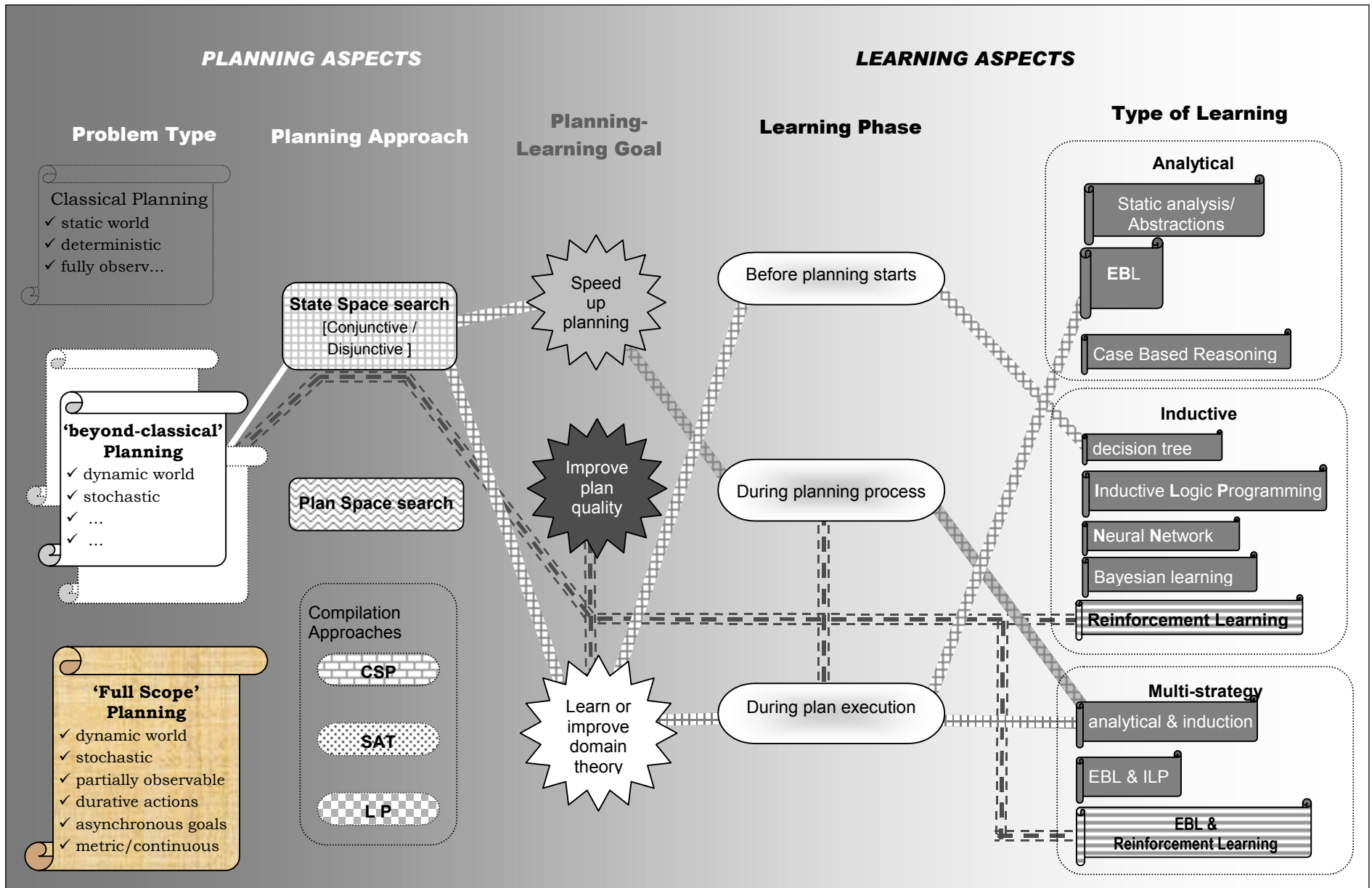


Figure 4: Mapping of the survey planning-learning systems for *beyond-classical planning* problems onto the Figure 1 characterization structure.

domain theory’ (it requires none to determine the best action to take from each state). This view is reflected in Figure 4 as the vertical RL edge spanning these nodes under the planning-learning goal dimension. Finally, since RL 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 non-classical planning problems; all fielded systems plan via 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 non-classical 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. This is to be expected, since some configurations of planning/learning methods are intrinsically infeasible or poorly matched (e.g. the learning of 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.

Non-analytical learning techniques: The survey tables suggest a considerable bias towards analytical learning in planning. This deserves to be questioned. Why is this so favored? In a sense, a planner employing EBL is learning ‘guaranteed knowledge’, control information that is provably correct. But 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 inductive logic programming has been employed 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 be readily employed. It is curious to note, however, decision tree learning has been used in only one study in this entire survey. And yet this inductive technique is at *least* as mature and features its own very effective engines such as ID3 and C4.5 (Quinlan 1986, ‘93) There is no obvious reason why the approach of learning decision trees to, for example, select the best action to apply in a given state has not been investigated within the planning community.

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 (e.g. via EBL). There may be two culprits that led to disenchantment with learning this inter-problem search control:

- The ‘utility problem’ that can surface when too many, or relatively ineffective rules are learned.
- 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 to be insurmountable hurdles, we suggest the following research approaches:

One tradeoff 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 may learn a great number of nogoods during search on a given problem, but in their basic form, they are only relevant to the given problem. Graphplan retains no inter-problem memory. It is worth considering what might constitute effective inter-problem learning for such a system.

The rule utility issue faced by analytical 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 over-specific rules. Learning systems that can reasonably relax the ‘soundness’ criterion for learned rules may move broadly towards a problem goal using ‘generally correct’ search control. Some of the multi-strategy studies reflected in Table 3C, are relevant to this view to the extent that they attempt to leverage the strengths of both analytical 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 speedup 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 –even though a human would find it natural to use what she has learned in solving a AIPS planning competition ‘driverlog’ problem to a ‘depot’ problem. We note that this has been approached from a somewhat different direction in (Fox, Long 1999) via a process of identifying abstract types during domain preprocessing.

Extending Learning to Non-Classical Planning Problems: The preponderance of planning research has been based in classical planning, as is born out by the survey tables and figures. Historically this was largely motivated by 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 non-classical 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 non-classical 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 speed-up learning, the focus may soon shift back in that 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

Off-line 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 may 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 *non-classical* planning; static analysis –the learning of domain invariants

before planning starts. This 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 non-classical 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, Kabanza 2000, ‘Blackbox’; Kautz, 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 hand-code the domain knowledge ahead of time correctly 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 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 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/hardest to learn. This has a direct impact on which 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 this has received little attention with respect to the domain-specific knowledge employed 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 vs. 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 both 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 been largely 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 a 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 employed in LRTA* (Korf 1990)
- a meta-level function that selects or modifies a search heuristics based on the problem / domain

Multi-strategy 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 meta-planning system with capabilities well beyond any individual planner. Interesting steps in this direction have been taken by (Horvitz, et. al. 2001) via the construction and use of Bayesian models to predict the run time of various problem solvers.

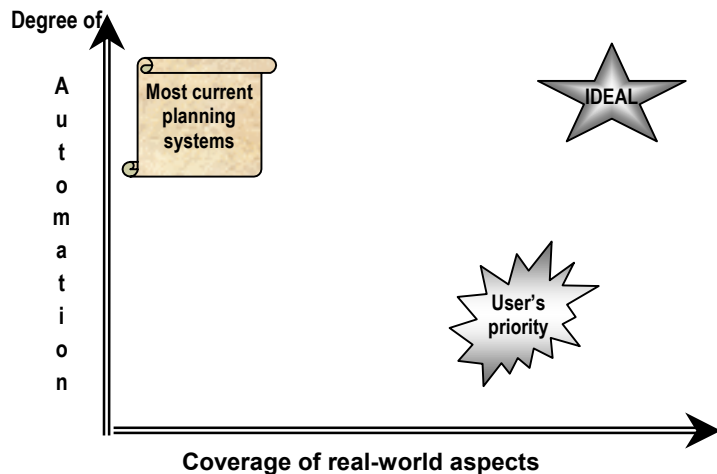


Figure 5. The coverage vs. automation tradeoff in planning systems

Learning to improve plan quality:

The survey tables and figures suggest that the issue of improving ‘plan quality’ via learning, has received much less attention in the planning community than speedup learning. Yet, as planning systems are ported into real-world applications, this is likely to be a primary concern. 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 may be hard-pressed to quantify. These planning systems could be charted in a two-dimensional space with the 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.
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 present 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. And yet, 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 plan generation is neither feasible nor desirable because users wish to observe and control plan generation.

Some planning systems such as HICAP [Aha, et. al 1999] and ALPS [Calistri-Yeh, Segre 1996] have made inroads towards 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 may be that the best way to tailor an interactive planner will be after the manner of the “programming by demonstration” systems that have recently received attention in the machine learning community [Lau, Domingos, 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 a 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 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 that merit more research attention than they have received to date. We have expanded on several of these possibilities and offered our conjectures as to where the more interesting work might lie.

References

- Aler, R. Borrajo, D., Isasi, P. 1998. Genetic Programming and Deductive-inductive Learning: a Multistrategy Approach. Proceedings of the Fifteenth International Conference on Machine Learning, ICML '98, pages 10-18.
- Aler, R. Borrajo, D., 2002. On Control Knowledge Acquisition by Exploiting Human-Computer Interaction. Proceedings of the Sixth International Conference on Artificial Intelligence Planning and Scheduling. Toulouse, France, April, 2002.
- Almuallim, H., Dietterich, T. G. 1991. Learning with many irrelevant features. Proceedings of the Ninth National Conference on Artificial Intelligence (AAAI-91) (pp. 547-552). Anaheim, CA: AAAI Press
- Almuallim, H., Dietterich, T. G. 1994. Learning Boolean concepts in the presence of many irrelevant features. *Artificial Intelligence*, 69(1-2): 279-306.
- Ambite, J.L., Knoblock, C.A., Minton, S. 2000. Learning Plan Rewriting Rules. Proceedings The Fifth International Conference on Artificial Intelligence Planning and Scheduling. Breckinridge, CO., April, 2000.
- Ashley, K. D., McLaren, B. 1995. Reasoning with reasons in case-based comparisons. Proceedings of the First International Conference on Cased-Based Reasoning (ICCB-95) (pp. 133-144). Berlin: Springer.
- Ashley, K., Alevan, V. 1997. Reasoning symbolically about partially matched cases. Proceedings of the Fifteenth International Joint Conference on Artificial Intelligence, IJCAI-97 (pp.335- 341).
- Avesani, P., Perini, A., Ricci, F. 2000. Interactive Case-Based Planning for Forest Fire Management. *Applied Intelligence* 13 (1):41-57, July 2000.
- Barto, A., Bradtke, S., and Singh, S. 1995. Learning to act using real-time dynamic programming. *Artificial Intelligence* 72:81-138.
- Bacchus, F., Kabanza, F. 2000. Using temporal logics to express search control knowledge for planning. *Artificial Intelligence*, 116.
- Bennett, S.W., DeJong, G.F., 1996. Real-World Robotics: Learning to Plan for Robust Execution. *Machine Learning* 23 (2/3):121-161, May 1996.
- Bergmann, R., Wilke, W. 1996. On the Role of Abstractions in Case-Based Reasoning. In EWCB-96 European Conference on Case-Based Reasoning. Springer.
- Bhatnagar, N., Mostow, J. 1994. On-line learning from search failures. *Machine Learning* 15(1):69-117, April, 1994.
- Blum, A., Furst, M.L. 1997. Fast planning through planning graph analysis. *Artificial Intelligence*. 90(1-2).
- Borrajo D., Veloso, M. 1997. Lazy incremental learning of control knowledge for efficiently obtaining quality plans. *Artificial Intelligence Review*, 11(1/5): 371-405, February 1997.
- Bylander, T. 1992. Complexity results for serial decomposability. Proceedings of the 10th National Conference on Artificial Intelligence (AAAI-92).
- Calistri-Yeh, R., Segre, A., Sturgill, D. 1996. The Peaks and Valleys of ALPS: An Adaptive Learning and Planning System for Transportation Scheduling. Proceedings of Third Int'l Conference on Artificial Intelligence Planning Systems (AIPS-96).
- Carbonell, Y.G, Gil, Y. 1990. Learning by experimentation: The operator refinement method. In Y. Kodratoff & R.S. Michalski (Eds.) *Machine Learning: An artificial intelligence approach*. (Vol 3). :Morgan Kaufmann.
- Chien, S.A. 1989. Using and Refining Simplifications: Explanation-Based Learning of Plans in Intractable Domains. In Proceedings of IJCAI 1989: 590-595, Detroit, Mi.
- Cohen, W.W. 1990. Learning approximate control rules of high utility. In Proceedings of the Seventh International Conference on Machine Learning, Austin, Texas, Morgan Kaufmann.
- Cohen, W.W., Singer, Y. 1999. A simple, fast, and effective rule learner. Proceedings of the Sixteenth National Conference on Artificial Intelligence, Orlando, Florida, 1999.
- Craven, M., Shavlik, J. 1993. Learning Symbolic Rules Using Artificial Neural Networks. Proceedings of the Tenth International Conference on Machine Learning, pp. 73-80, Amherst, MA. Morgan Kaufmann.
- Dearden, R., Friedman, N., Russell, S. 1998. Bayesian Q-Learning. Proceedings of the Fifteenth National Conference on Artificial Intelligence (AAAI-98).
- Dempster, A. P., Laird, N.M., & Rubin, D.B. 1977. Maximum likelihood from incomplete data via the EM algorithm. *Journal of the Royal Statistical Society, Series B*, 39(1), 1-38.

- Dietterich, T. G., Flann, N. S. 1997. Explanation-based Learning and Reinforcement Learning: A Unified View. *Machine Learning*, 28, 169-210.
- Do, B., Kambhampati, S. 2001. Planning as Constraint Satisfaction: Solving Planning Graph by Compiling it into a CSP. To appear in *AI Journal*, 2001.
- Estlin, T.A., Mooney, R.J. 1996. Multi-strategy learning of search control for partial-order planning. *Proceedings of the Thirteenth National Conference on Artificial Intelligence* (pp. 843-848). Portland, OR. AAAI Press.
- Etzioni, O. 1993. Acquiring search-control knowledge via static analysis. *Artificial Intelligence*, 62(2):265-301.
- Fikes, R.E., 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., Nilsson, N.J. 1972. Learning and executing generalized robot plans. *Artificial Intelligence* 3, 4 (1972).
- Fox, M., Long, D. 1998. The automatic inference of state invariants in TIM. *JAIR* 9.
- Fox, M., Long, D. 1999. The detection and exploitation of symmetry in planning problems. In *Proceedings of the International Joint Conference on Artificial Intelligence*, 1999.
- Fu, Li-Min. 1989. Integration of neural heuristics into knowledge-based inference. *Connection Science*, 1(3), 1989.
- García-Martínez, R., Borrajo, D. 2000. An Integrated Approach of Learning, Planning, and Execution. *Journal of Intelligent and Robotic Systems*, 29 (1):47-78, September 2000.
- Gerevini, A., Schubert, L. 1996. Accelerating Partial Order Planners: Some techniques for effective search control and pruning. *JAIR* 5:95-137.
- Gerevini, A., Schubert, L. 1998. Inferring state constraints for domain-independent planning. In *Proceedings of the Fifteenth National Conference on Artificial Intelligence*, pp905-912, Madison, WI, July 1998. AAAI Press.
- Gil, Y. 1994. Learning by experimentation: Incremental refinement of incomplete planning domains. *Proceedings of the Eleventh International Conference on Machine Learning*, 1994, Rutgers, NJ.
- Gratch, J., Dejong, G., 1992. COMPOSER: a probabilistic solution to the utility problem in speed-up learning, *Proceedings of the Tenth National Conference on Artificial Intelligence*, 1992.
- Hammond, K. 1989. *Case-Based Planning: Viewing planning as a memory task*. San Diego: Academic Press.
- Hanks, S., Weld, D. 1995. A domain-independent algorithm for plan adaptation. *Journal of Artificial Intelligence Research (JAIR)*. pp 319-360, 1995.
- Hinton, G.E. 1989. Connectionist learning procedures. *Artificial Intelligence*, 40, 185-234.
- Hofstadter, D. R., Marshall, J. B. D. 1993. *A Self-Watching Cognitive Architecture of High-Level Perception and Analogy-Making*. TR100, Indiana University Center for Research on Concepts and Cognition, 1993.
- Hofstadter, D. R., Marshall, J. B. D., 1996. Beyond copycat: Incorporating self-watching into a computer model of high-level perception and analogy-making, In M.Gasser (ed.), *Online Proceedings of the 1996 Midwest Artificial Intelligence and Cognitive Science Conference*.
- Hollatz, J. 1999. Analogy making in legal reasoning with neural networks and fuzzy logic. *Artificial Intelligence and Law*, 7 (2/3):289-301, September 1999.
- Horvitz, E., Ruan, Y., Gomes, C., Kautz, H., Selman, B., Chickering, D.M. 2001. A Bayesian approach to tackling hard computational problems. *Proc. of 17th Conference on Uncertainty in Artificial Intelligence*, August 2001.
- Huang, Y., Kautz, H., Selman, B. 2000. Learning declarative control rules for constraint-based planning. In *Proceedings of Seventeenth ICML*, 2000.
- Hunt, E.B., Marin, J., & Stone, P.J. 1966. *Experiments in Induction*. New York: Academic Press.
- Ihrig, L., Kambhampati, S. 1996. Design and Implementation of a Replay Framework based on a Partial order Planner. *Proceedings of the Thirteenth National Conference on Artificial Intelligence (AAAI-96)*.
- Ihrig, L., Kambhampati, S. 1997. Storing and Indexing plan derivations through explanation-based analysis of retrieval failures. *Journal of Artificial Intelligence*, Vol 7, pp. 161-198. 1997.
- Jones, R., Langley, P. 1995. Retrieval and learning in analogical problem solving. *Proceedings of the Seventeenth Conference of the Cognitive Science Society* (pp. 466-471). Pittsburgh: Lawrence Erlbaum.
- Kakuta, T., Haraguchi, M., Midori-ku, N., Okubo, Y. 1997. A Goal-Dependent Abstraction for Legal Reasoning by Analogy. *Artificial Intelligence and Law*, 5 (1/2):97-118, March 1997.
- Kambhampati, S., & Hendler, J. 1992. A validation structure based theory of plan modification and reuse. *Artificial Intelligence*, 55, 193-258.
- Kambhampati, S., Katukam, Y. Qu. 1996. Failure Driven Dynamic Search Control for Partial Order Planners: An explanation-based approach. *Artificial Intelligence* 88 (1996) 253-315.
- Kambhampati, S. 1999. On the relations between intelligent backtracking and explanation based learning in Planning and CSP. *ASU CSE TR 97-018*. *Artificial Intelligence*, Spring 1999
- Kambhampati, S. 2000. Planning Graph as (dynamic) CSP: Exploiting EBL, DDB and other CSP Techniques in Graphplan. *Journal of Artificial Intelligence research (JAIR)*. 2000.

- Kautz, H., Selman, B. 1998. The Role of Domain-Specific Knowledge in the Planning as Satisfiability Framework. Proceedings of Fifth International Conference on Planning and Scheduling (AIPS-98), Pittsburgh, PA, June 1998.
- Kautz, H., Selman, B. 1999. Blackbox: Unifying sat-based and graph-based planning. Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI-99.
- Kedar-Cabelli, S., McCarty, T. 1987. Explanation-based generalization as resolution theorem proving. Proceedings of the Fourth International Workshop on Machine Learning (pp. 383-389). San Francisco.
- Khardon, R. 1999. Learning Action Strategies for Planning Domains. Artificial Intelligence, 113, 125-148.
- Knoblock, C. 1990. Learning abstraction hierarchies for problem solving. Proceedings of Eighth National Conference on Artificial Intelligence, pages 923-928, Boston, MA, 1990.
- Kodratoff, Y. & Michalski, R.S. editors. 1990. *Machine Learning: An artificial intelligence approach*. (Vol 3) :Morgan Kaufmann.
- Korf, R. 1990. Real-time heuristic search. Artificial Intelligence, 42:189-211, 1990.
- Laird, J., Newell, A., Rosenbloom, P. 1987. SOAR: An Architecture for General Intelligence. Artificial Intelligence, 33, 1987.
- Langley, P. 1997. Challenges for the application of machine learning. Proceedings of the ICML '97 Workshop on Machine Learning Application in the Real World: Methodological Aspects and Implications. 15-18.
- Lang, K. 1995. NewsWeeder: Learning to filter netnews. In Prieditis and Russell (Eds.), Proceedings of the 12th International Conference on Machine Learning (pp. 331-339), San Francisco: Morgan Kaufmann Publishers.
- Lau, T., Domingos, P., Weld, D. 2000 "Version Space Algebra and its Application to Programming by Demonstration" ICML-2000.
- Lavrac, N., Dzeroski, S., Grobelnik, M. 1991. Learning nonrecursive definitions of relations with LINUS. In Proc. Fifth European Working Session on Learning, pp 265-281. Springer, Berlin.
- Leake, D., Kinley, A., Wilson, D. 1996. Acquiring case adaptation knowledge: a hybrid approach. Proceedings of Thirteenth National Conference on Artificial Intelligence. Portland, Ore, 1996.
- Leckie, C., Zuckerman, I. 1998. Inductive learning of search control rules for planning. Artificial Intelligence, Vol. 101 (1-2) (1998) pp. 63-98
- Martin, M., Geffner, H. 2000. Learning generalized policies in planning using concept languages. Proceedings of 7th International Conference on Knowledge Representation and Reasoning (KR 2000). Colorado, Morgan Kaufmann
- Minton, S., editor. 1993. Machine Learning Methods for Planning. San Francisco: 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, Vol. 40, Sept. 1989.
- Mitchell, T., Keller, R., & Kedar-Cabelli, S. 1986. Explanation-based generalization: a unifying view. Machine Learning, 1 (1), 47--80.
- Mitchell, T.M., Thrun, S.B. 1995. Learning Analytically and Inductively. In Mind Matters: A Tribute to Allen Newell, Steier and Mitchell (eds.), Erlbaum.
- Muggleton, S., Feng, C. 1990. Efficient inductin of logic programs. Proceedings of the 1st conference on algorithmic learning theory, pp 368-381. Ohmsma, Tokyo, Japan. 1990.
- Munoz-Avila, H., Aha, D.W., Breslow, L. & Nau, D. 1999. HICAP: An interactive case-based planning architecture and its application to noncombatant evacuation operations. Proceedings of the Ninth Conference on Innovative Applications of Artificial Intelligence (pp. 879 - 885). Orlando, FL: AAAI Press.
- Nau, D., Cao, Y., Lotem, A., Munoz-Avila, H. 1999. SHOP: Simple Hierarchical Order Planner. Proceedings of 16th International Joint Conference on Artificial Intelligence. 1999.
- Nebel, B. Dimopoulos, Y. Koehler, J. 1997. Ignoring Irrelevant Facts and Operators in Plan Generation, European Conference on Planning (ECP-97), pages 338-350
- Nguyen, X., Kambhampati, S. 2001. Reviving Partial Order Planning. In Proceedings IJCAI-2001, Seattle, Wa.
- Ourston, D., Mooney, R. 1994. Theory refinement combining analytical and empirical methods. Artificial Intelligence, 66:311-344, 1994.
- Pazzani, M.J., Brunk, C.A., Silverstein, G. 1991. A knowledge-intensive approach to learning relational concepts. In Proceedings of the Eighth International Workshop on Machine Learning, pp 432-436, Evanston, IL, 1991.
- Perez, M., Etzioni, O. 1992. DYNAMIC: a new role for training problems in EBL. In Proceedings of the 9th International Conference on Machine Learning. Morgan Kaufmann, July 1992.
- Pomerleau, D. A. 1993. Knowledge-based training of artificial neural networks for autonomous robot driving. In J. Connell & S. Mahadevan (Eds.), Robot Learning (pp. 19-43). Boston: Kluwer Academic Publishers.
- Quinlan, J.R. 1986. Induction of decision trees. Machine Learning, 1(1), 81-106.
- Quinlan, J.R. 1990. Learning logical definitions from relations. Machine Learning, 5, 239-266.
- Quinlan, J.R. 1993. C4.5: Programs for Machine Learning. San Mateo, CA: Morgan Kaufmann.
- Reddy, C., Tadepalli, P. 1999. Learning Horn Definitions: Theory and an Application to Planning. New Generation Computing vol 17, 77--98, 1999.

- Rintanen, J. 2000. An iterative algorithm for synthesizing invariants, Proceedings of the 17th National Conference on Artificial Intelligence / 12th Innovative Applications of AI Conference, pages 806-811, AAAI Press, 2000.
- Sacerdoti, E. 1974. Planning in a hierarchy of abstraction spaces. *Artificial Intelligence*, 5(2):115-135, 1974.
- 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., Cohen, P. 2000. Learning Planning Operators in Real-World, Partially Observable Environments. Proceedings of Int'l Conference on Artificial Intelligence Planning Systems (AIPS-2000),
- Shavlik, J.W., Towell, G.G. 1989. An approach to combining explanation-based and neural learning algorithms. *Connection Science*, 1(3):231-253, 1989.
- Sheppard, J., Salzberg, S. 1995. Combining genetic algorithms with memory based reasoning. Proceedings of the Sixth International Conference on Genetic Algorithms, Pittsburgh, Penn. 1995.
- Smith, D., Peot, M. 1993. Postponing Threats in Partial-Order Planning. In Proceedings of the Eleventh National Conference on Artificial Intelligence (AAAI-93), 1993.
- Sutton, R. 1988. Learning to predict by the methods of temporal differences. *Machine Learning*, 3, 9-44.
- Sutton, R. 1991. Planning by incremental dynamic programming. Proceedings of the Eighth International Conference on Machine Learning (pp. 353-357). San Francisco: Morgan Kaufmann.
- Sutton, R., Barto, G. 1998. Reinforcement Learning -An Introduction. Cambridge: The MIT Press.
- Sycara, K., Guttal, R., Koning, J., Narasimhan, S. Navinchandra, D. 1992. "CADET": a case-based synthesis tool for engineering design. *International Journal of Expert Systems*, Vol. 4, No. 2, 1992.
- Tadepalli, P. 1989. Lazy Explanation-Based Learning: A Solution to the Intractable Theory Problem. Proceedings of International Joint Conference on Artificial Intelligence, Detroit, MI, 1989.
- Tadepalli, P. 1993. Learning from queries and examples with tree-structured bias. Proceedings of the Tenth International Conference on Machine Learning, Amherst, Massachusetts. Morgan Kaufmann.
- Veloso, M., Carbonell, J. 1993. Derivational analogy in PRODIGY: automating case acquisition, storage, and utilization. *Machine Learning*, 10:249-278.
- Wang, X. 1996a. A Multistrategy Learning System for Planning Operator Acquisition. Third International Workshop on Multistrategy Learning, Harpers Ferry, West Virginia, May 1996
- Wang, X.: 1996b, Planning while learning operators, in: B. Drabble (ed.), Proc. of the 3rd International Conference on Artificial Intelligence Planning Systems (AIPS'96), Edinburgh, Scotland, pp. 229-236.
- Watkins, C. 1989. Learning from delayed rewards (Ph.D. dissertation). King's College, Cambridge, England.
- Wolfman, S., Weld, D. 1999. The LPSAT Engine & its Application to Resource Planning. Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI-99.
- XuanLong, N., Kambhampati, S. 2001. Reviving Partial Order Planning. In Proceedings of the International Joint Conference on Artificial Intelligence, 2001.
- Zelle, J., Mooney, R. 1993. Combining FOIL and EBG to Speed-up Logic Programs. Proceedings of 13th International Joint Conference on Artificial Intelligence. pp 1106-1111.
- Zimmerman, T., Kambhampati, S. 1999. Exploiting Symmetry in the Planning-graph via Explanation-Guided Search. In *Proceedings AAAI-99*, 1999.
- Zimmerman, T., Kambhampati, S. 2002. Generating parallel plans satisfying multiple criteria in anytime fashion. Sixth International Conference on Artificial Intelligence Planning and Scheduling, workshop on Planning and Scheduling with Multiple Criteria, Toulouse, France. April, 2002.
- Zweben, M., Davis, E., Daun, B., Drascher, E., Deale, M. & Eskey, M., 1992. "Learning to Improve Constraint-Based Scheduling", *Artificial Intelligence*, Vol.58, Nos.1-3, pp271-296, December, 1992.