In *Acquisition, Learning and Demonstration: Automating Tasks for Users. Papers from the 1996 AAAI Symposium.* March 1996, Stanford, CA. Technical Report SS-96-02. AAAI Press. Menlo Park, CA

Learning from a Domain Expert to Generate Good Plans

M. Alicia Pérez*
Computer Science Department
Boston College
Chestnut Hill, MA 02167
E-mail: *perezab@cs.bc.edu*

Abstract

This paper describes QUALITY, a domain-independent architecture that learns operational quality-improving search-control knowledge given a domain theory, a domain-specific metric of plan quality, and problems which provide planning experience. QUALITY can (optionally) interact with a human expert in the planning application domain who suggests improvements to the plans at the operator (plan step) level. The framework includes two distinct domain-independent learning mechanisms which differ in the language used to represent the learned knowledge, namely control rules and control knowledge trees, and in the kinds of quality metrics for which they are best suited. QUALITY is fully implemented on top of the PRODIGY4.0 nonlinear planner and its empirical evaluation has shown that the learned knowledge is able to substantially improve plan quality.

1 Introduction

In spite of advances in knowledge acquisition techniques and tools, acquiring knowledge, and in particular acquiring control knowledge, is still a major bottleneck in building complex planning domains. Previous research has shown the effectiveness of a variety of machine learning methods to capture problem solving heuristics expressed in a number of representation formalisms. In particular, most of the research to date in the application of machine learning to planning systems has focused on *planning efficiency*, that is, on acquiring problem solving strategies that control search in order to make problem solving more efficient. This area of research has been termed "speed-up learning" [Mitchell *et al.*, 1986; Minton *et al.*, 1989; Tadepalli, 1989; Gratch *et al.*, 1993; Pérez and Etzioni, 1992]. The research described in this paper looks instead at the application of machine learning to acquire strategies that lead a planner towards improving *plan quality*, an essential step in transforming planners from research tools into real-world applications. The paper describes QUALITY, an architecture to learn quality-enhancing search control knowledge. Given a domain-specific quality metric, QUALITY compares search traces[1] corresponding to plans of different quality, explains why their qualities differ, and uses those explanations to produce search control knowledge. Planning traces provide the learner with planning experience. These traces can be obtained autonomously by the planner by exploring more or less exhaustively the search space in order to provide good training examples. However, to limit that computational cost, QUALITY can also interact with a human domain expert who criticizes and improves the planner's proposed solution. Thus QUALITY advantageously exploits human expertise on building good plans, which is available in many domains. The paper starts by introducing the problem of finding good quality plans and QUALITY's architecture (Section 2). Section 3 describes the motivation for and characteristics of the domain expert interaction. Section 4 briefly presents two representation formalisms for quality-enhancing control knowledge that are learned automatically by QUALITY. QUALITY has been been fully implemented on top of the PRODIGY4.0 nonlinear planner [Veloso *et al.*, 1995] and some of the experimental results obtained are described in Section 5. The remain-

*This research was conducted while the author was at Carnegie Mellon University and was sponsored by the Wright Laboratory, Aeronautical Systems Center, Air Force Materiel Command, USAF, and the Advanced Research Projects Agency (ARPA) under grant number F33615-93-1-1330. The views and conclusions contained in this document are those of the author and should not be interpreted as necessarily representing the official policies or endorsements, either expressed or implied, of Wright Laboratory or the U. S. Government. The author held previously a scholarship from the Ministerio de Educación y Ciencia of Spain. Thanks to Jaime Carbonell, Manuela Veloso, Reid Simmons, Tom Mitchell, Martha Pollack, and the members of the PRODIGY project for helpful discussions and comments on this research.

[1] By *search trace* we mean the sequence of decisions made during plan generation.

ing sections discuss some related work and conclusions.

2 The Problem of Learning to Generate Good Plans

Knowledge about plan quality in a domain D comes in two forms: (a) a post-facto quality metric $Q_D(P)$ that computes the quality (e.g. the execution cost) of plan P, and (b) planning-time decision-control knowledge used to guide the planner towards producing higher-quality plans. The first kind of knowledge, $Q_D(P)$, is non-operational; it cannot be brought into play until *after* a plan is produced. Yet, cost functions is exactly the kind of quality knowledge typically available, in contrast to the far more complex operational decision-time knowledge. Hence, automatically acquiring the second kind of knowledge from the first is a very useful, if quite difficult endeavor. In essence, learning operational (planning-time) quality control knowledge can be seen as a *translation* problem of domain knowledge D and quality metric Q_D into runtime decision control guidance aided by planning experience E in the form of actual planning episodes and planning decisions and human expert suggestions that bias the learning

$$\textit{Quality: } D \times Q_D \times E \rightarrow \textit{Decision-Control}_{D,Q_D}$$

And the *full automation of the quality mapping problem* is the ultimate objective of our research.

Figure 1 summarizes the problem of finding good plans with an example in a process planning domain (Figure 1(a)). Three distinct metrics of plan quality in this domain are displayed in the table of Figure 1(b). In this example plan quality corresponds to plan execution cost and the metrics assign a fixed cost to each operator (lower values are higher quality). However those costs in general may depend on the particular operator instantiation (bindings). The total quality (cost) of a plan P is computed by adding the cost of the plan operators. (Throughout the paper we focus on quality metrics of that kind.) Figure 1(c) presents two plans for the problem in (a). Figure 1(d) shows the quality values for each of the plans and the quality metrics. Values in bold face indicate the cost of the better plan in each case: Plan 1 is better under metrics Q_1 and Q_3. Plan 2 is better under metric Q_2.

Figure 1 (e) shows the decisions that the planner confronts at problem solving time when given the example problem. Those decisions are different depending on the chosen quality metric. The example shown is a simple one. There may be other goals to achieve, and thus decisions to make about which goal to work on next, and also interactions among goals. There may be other alternative operations to achieve the problem goals, and many resources (tools, machines, different types of raw materials) that must be selected by the planner.

Thus, the problem we address is how to automatically acquire search-control knowledge that will guide the planner to make the decisions leading to the better

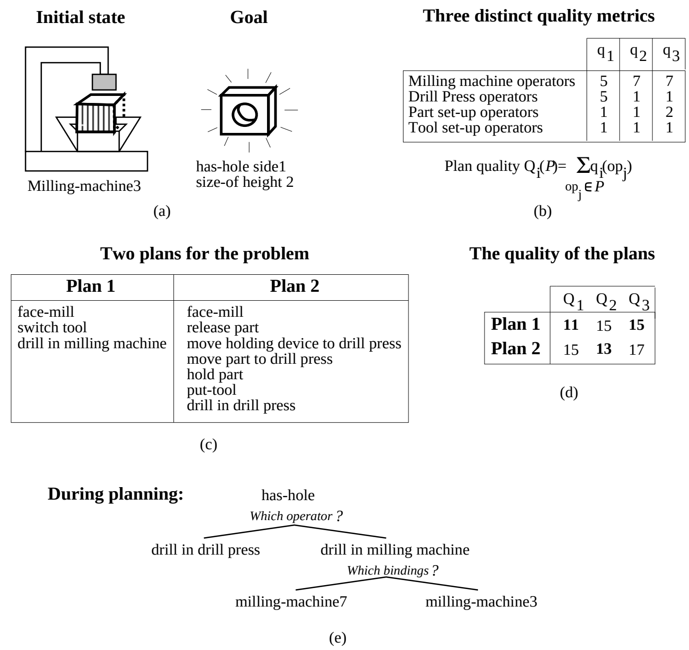

Figure 1: The problem of finding good plans.

plans, for each particular domain-specific quality metric. Note that such control knowledge is orthogonal to planning efficiency control knowledge for early pruning of choices that are guaranteed to lead to failure paths. (Such control knowledge has been the target of extensive research [Mitchell *et al.*, 1986; Minton *et al.*, 1989; Tadepalli, 1989; Gratch *et al.*, 1993; Pérez and Etzioni, 1992]).

Figure 2 shows the architecture of QUALITY which addresses the quality mapping problem. QUALITY is *given* a domain theory D (operators and inference rules) and a domain-dependent metric that evaluates the quality of the plans produced $Q_D(P)$. It is also given problems to solve in that domain. QUALITY analyzes the planning episodes by comparing the search trace for the plan obtained given the current control knowledge, and another search trace corresponding to a better plan (*better* according to the quality metric). Section 3 describes how the latter search trace is obtained. QUALITY then analyzes the differences between the sequence of decisions that the planner made initially and the ones that should have been made to generate the plan of better quality. The learner interprets these differences as learning opportunities and identifies the conditions under which the individual planning choices will lead to the desired final plan. In this way QUALITY compiles knowledge to control the decision making process in new similar planning situations to generate plans of better quality.

3 Interaction with a Domain Expert

In order to obtain the search trace corresponding to the better plan QUALITY can either function autonomously or interact with a human expert. In its *autonomous*

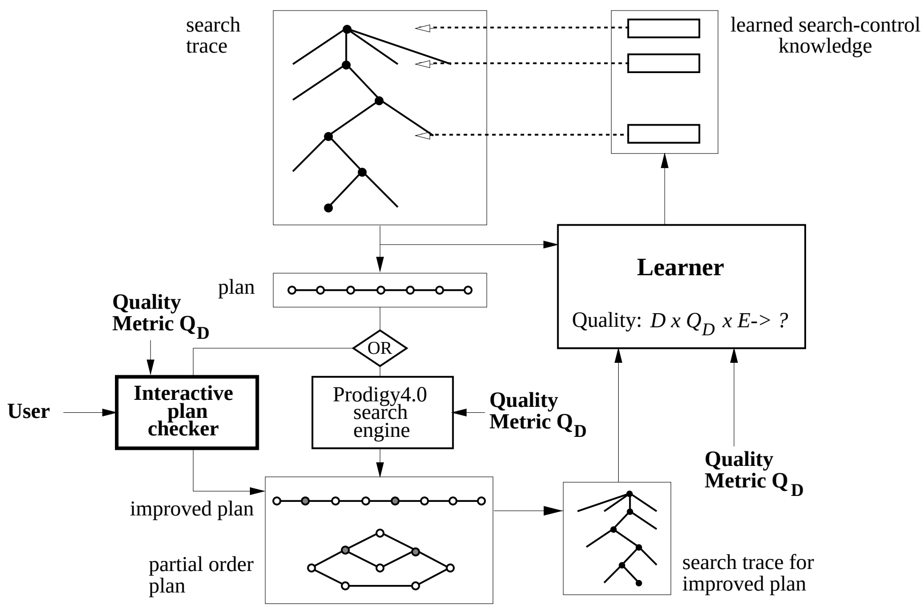

Figure 2: Architecture of QUALITY.

mode, once the planner has come up with the initial plan, QUALITY asks it to further explore the search space finding plans of increasing quality as determined by the available quality metric Q_D, until the space is exhausted or some typically large resource bound is met. The best plan found within that bound and its corresponding search trace are passed to the learner. If the best plan is different from the initial plan learning is triggered.

Because of the large search spaces in complex domains, finding good enough plans from which to learn can be computationally very expensive. Optimization by exhaustive search is exponential in the length of the optimal plan (which may not be the shortest plan). On the other hand, human expertise is available in many domains, and can be advantageously used to help the system find useful strategies to obtain good plans. In its *interactive mode* QUALITY asks a human for a better plan and then calls the planner to produce a search trace that leads to that plan. Details on how the search trace is generated from the plan can be found in [Pérez, 1995]. QUALITY assumes that the expert's model of plan quality and the quality metric Q_D are consistent. In particular if the expert's plan is worse than the initial one, the expert's plan is rejected. The interaction with the expert is the task of the Interactive Plan Checker in the figure. This interaction occurs at the level of plan steps, i.e. concrete actions in the plan, which correspond to instantiated operators, and not at the level of the full range of problem-solving time decisions. The Interactive Plan Checker offers to the expert the initial plan obtained by the planner with the current knowledge as a guide to build a better plan. The expert critiques the plan by adding, removing, or modifying the steps of the initial plan.[2] The Interactive Plan Checker checks the correctness of the plan suggested by verifying the applicability of its steps and whether it achieves the problem goal. This type of interaction requires the expert to be familiar with the available operators, their parameters, preconditions, and effects. QUALITY's assumption (and design goal) is that the human is an expert in the application domain but can remain oblivious to the planner's algorithm and control knowledge representation language. This relieves the expert of understanding PRODIGY4.0's search procedure and control knowledge encoding, a very important feature to interact productively with domain experts who are not knowledge engineers.

With such design assumption in mind, when we allow the system to learn through interaction with a human, we must consider the gap between the expert's view of the world and efficient search control knowledge that captures such expertise. Ideally the expert should be able to provide advice using terms about the specific application domain and to ignore the details of the internal representation and the planning algorithm. On the other hand, the planner's control knowledge typically refers to problem solving states (e.g. operators that have been expanded, goals pending exploration) in which the control decision applies. This kind of control knowledge is hard to give by such an expert. However the expert can easily provide a plan for the problem and, even better, critique and improve the plan obtained by the planner by suggesting additions, deletions, or modifications of plan steps. This advice, if consistent with the

[2]Developing the user interface was not a goal of this research. A more user-friendly interface, possibly integrated with PRODIGY4.0's graphical user interface, is under development.

quality metric, is then *operationalized* by the learning system, that is, translated into knowledge usable efficiently during problem solving. Thus, the goal of learning problem solving expertise can be seen as translating a non-operational domain theory into an operational one [Tadepalli, 1990]. In the previous section we referred to it as the *quality mapping problem.* In our case the domain theory consists of the description of the domain (planning operators, inference rules, and type hierarchy) and the domain-specific quality metric. The goal of learning is to translate it into operational planning-time search-control knowledge based on the planner's experience and, if available, the advice of a human expert.

4 Learning Search Control Knowledge

We have developed two different domain-independent learning mechanisms within QUALITY to efficiently acquire quality control knowledge.[3] They differ in the language used to represent the learned knowledge, in the algorithms themselves, and in the kinds of quality metrics for which they are best suited:

- *Learning search-control rules.*
  The first mechanism learns control knowledge in the form of control rules, in particular of PRODIGY's prefer control rules [Minton *et al.*, 1989; Veloso *et al.*, 1995]. These are productions (*if-then* rules) that indicate a preferred choice of an operator to achieve the current goal, choice of bindings (instantiation) for the chosen operator, or choice of the next goal to work on among the set of goals still pending. These decisions correspond to most of PRODIGY's decision points. The example in Section 2 shows how those choices are relevant to obtaining good plans. Previous algorithms developed in the context of the PRODIGY architecture learned control rules for PRODIGY2.0, the initial, linear planner of PRODIGY, with the goal of *planning efficiency* [Minton, 1988; Etzioni, 1990; Pérez and Etzioni, 1992]. QUALITY focuses on *plan quality.*

- *Learning control knowledge trees.*
  In the second learning mechanism within QUALITY the learned control knowledge is represented using a formalism that we call *control knowledge trees* (cktrees). The rest of this section focuses on control knowledge trees.

The motivation for this new representation is that, in general, complex quality metrics require reasoning about tradeoffs and taking a global view of the plan to make a set of globally optimal choices. Acquiring control rules that apply at individual decision points may prove insufficient. Instead, a more globally-scoped method is required. Control knowledge trees provide a more global view of the planning decisions and are used, together with the quality metric, to estimate the quality of each available alternative at a given planning decision point.

A control knowledge tree (*cktree*) has goal, operator, and parameterized operator binding nodes.[4] For our purposes consider that an operator node and its child binding node form a single node. Figure 3 shows two cktrees in the process planning domain. The root of a cktree is a parameterized goal node that corresponds to a goal of a problem in which learning was invoked. A goal node is an OR node and its children are operator nodes corresponding to the operators that have been used in past planning experience to achieve that goal. In the example, the `has-spot` goal node has two children: two alternative ways to achieve the goal (cf. Figure 1), and it was learned from a problem in which the planner's initial plan and the improved plan achieved the goal in two different ways. When the cktree is built, constants are generalized to variables. The children of an operator node are the operator preconditions. The cktree contains additional links that capture achievement and operator side effect information. For example, the link marked *achieves* in Figure 3 indicates that achieving the `holding` goal with the appropriate instantiation fulfills a precondition of both the face-mill and drill in milling machine operators. The link was added to the cktree because such achievement occurred in a past planning episode.

Cktrees are automatically learned from a planning episode by translating the search trace into a cktree and parameterizing its constants. The learner checks whether a cktree for the problem goal exists already. If not, a new one is built. Otherwise the learner updates the existing cktree detecting which new information is relevant to a decision that should have been made differently in order to obtain the higher quality plan, and adding it in the form of subtrees to the existing cktrees. The cktrees in the figure were learned from two planning episodes. The area in lighter font shows the subtree learned from the second episode. In the first episode, the `holding` subgoal was true in the initial state and needed not to be achieved. In the second episode that was not the case and further work (represented by the subtree below `holding`) was needed.

The cktrees are used to suggest operators and their instantiations for the goals and subgoals that appear in the cktree by *estimating* the quality of each alternative for the current situation and quality metric. When the planner is solving a new problem and confronts a control decision, if a cktree rooted at the problem top-level goal is available, it is used to provide an estimate of the quality of the final plan for each of the alternatives. The cktree matcher traverses the cktree starting at its root

[3]Describing these algorithms is out of the scope of this paper. Details on the algorithms, implementation, and experimental evaluation can be found in [Pérez, 1995].

[4]These three types of nodes correspond to the goal, operator, and bindings choice points in PRODIGY4.0's planning algorithm.

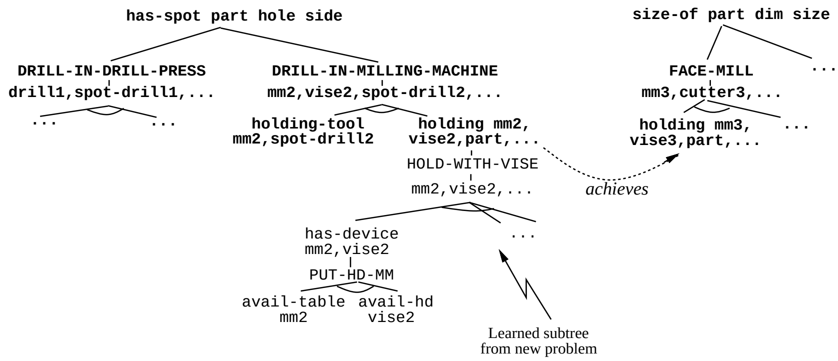

Figure 3: An example cktree automatically learned from a planning episode in the process planning domain similar to the one described in Figure 1.

which is instantiated with the current goal and recursively computing the estimate of achieving each subgoal as the sum of the instantiated operator's cost (according to the quality metric Q_D) plus the estimates of achieving its preconditions. At any time if the estimate exceeds a bound (the cost of the best alternative found so far) the exploration of the current alternative is abandoned. When exploring a goal node, if an estimate is available and can be reused, the subtree below the goal node needs not be explored. If the achievement links indicate that the goal is achieved as a side effect of other node visited, its cost is 0. Otherwise if there is no subtree below the node (because in past planning experience it as never achieved by subgoaling on it) a default approximate value is returned as an estimate. Thus the cktree provides *partial match* of previous experience to the current situation. When a node is traversed its achievement links are followed to mark as achieved other nodes it adds or deletes as a side effect (Figure 3). The cktree matcher may also provide guidance for other subgoals that appear in the traversed cktree that were part of the best alternative and that the planner will confront when subgoaling. For example, if the current goal is to have a hole in `part5` and the milling machine is suggested as the better choice, the cktree could also suggest operators and instantiations for drilling the hole (which instances of machine and tool to use), for holding the part, and even for face-milling it (cf Figure 3). In the learned control knowledge trees the quality metric is parameterized and therefore they provide adequate guidance when the metric differs from that for which they were learned.

Discussion. The learned quality-enhancing *control rules* provide effective guidance when the quality metric does not require reasoning about complex global trade-offs. They are highly operational and are efficiently used at planning time. In our empirical evaluation we have found that they do not reduce planning efficiency but increase it in many cases (when the higher-quality plans are also shorter in length). On the negative side, as the learned rules are domain and quality metric specific, if the metric changes they are invalidated and must be re-learned. The performance in improving plan quality of the learned *control knowledge trees* is equivalent to that of control rules for simpler non-interacting situations, and superior for more complex interactions and trade-off situations. However using control knowledge trees is computationally more expensive and may reduce planning efficiency. In addition control knowledge trees do not provide goal ordering control knowledge. In some of our experiments we have used them together with the goal preference rules learned by the first method achieving a synergistic effect. An important advantage of control knowledge trees is their robustness to changes in the quality metric. Integrating the two learning mechanisms to take advantage of their complementary characteristics remains an open issue.

5 Experimental Results

We have fully implemented all the algorithms described and evaluated their performance in a complex process-planning domain [Gil, 1991], in which they lead to significant plan quality improvements. We have also used a small transportation domain to further test the performance of control knowledge trees. This section focuses on one of the process planning experiments to evaluate the quality gained by using the learned control knowledge. Table 1 shows a quality that focuses on execution cost, in particular on the cost of setting up the work before machining a part.

QUALITY was given 60 randomly-generated problems in order to learn search control knowledge and in 13 of those an improved plan was suggested by the human expert. QUALITY's learning is cheap compared to planning: in those 13 problems the average planning time was 25.9 seconds (in a Sun Sparcstation ELC running Allegro Common Lisp version 4.1 under Mach/Unix) while the average learning time was 3.5 seconds. (Similar values were obtained when learning rules and cktrees.)

To test the performance of the knowledge learned PRODIGY4.0 was given 180 randomly-generated problems, different from the training problems. Table 2 com-

Type	Q_1	Operators	
Drill press operators	1	drill-in-drill-press, ...	(4 ops.)
Milling machine operators	1	face-mill, drill-in-milling-machine, ...	(6 ops.)
Machine and holding device set-up operators	8	put-holding-device-in-drill, ...	(6 ops.)
Tool operators	1	put-tool-on-milling-machine, ...	(4 ops.)
Cleaning operators	6	clean, remove-burrs	
Oil operators	3	add-soluble-oil, ...	(3 ops.)

Table 1: A quality metric in the process planning domain.

pares the quality (execution cost) of the plans obtained with and without the learned control rules. The improvement obtained for this quality metric using cktrees was similar. Each set of 30 problems corresponds to different parameters such as the number and type of goals. The problem sets are roughly ordered in increasing complexity. Not in all problems quality could actually be improved because the planner was able to find the better solution without any control knowledge. The table shows improvements over the base planner ranging between 20 and 60% in average and upto 96% in individual problems. By sampling the problem sets we found that most plans were *virtually optimal*, meaning that they could not be improved by the human expert (without systematically exploring the complete space of plans). The experiments also showed that the learned knowledge does not degrade considerably planning efficiency; in fact it improves it in many cases due to shorter solutions. [Pérez, 1995] shows detailed results on these experiments.

6 Related Work

Other learning research has focused on finding good plans. Iwamoto's system [Iwamoto, 1994] has developed an extension to PRODIGY to solve optimization problems and an EBL method to learn control rules to find near-optimal solutions in LSI design. HAMLET [Veloso *et al.*, 1995] learns control rules that improve both planning efficiency and also the quality (length) of the plans generated, by a combination of bounded explanation and induction. However neither method can take advantage of user guidance because they use exhaustive search to find the best solution.

Some learning apprentice systems learn by observing expert actions [Mitchell *et al.*, 1990; Martin and Redmond, 1989; Wilkins, 1988; Dent *et al.*, 1992; Wang, 1995]. Others attempt to solve problems and then allow the expert to critique their decisions [Laird *et al.*, 1990; Tecucci, 1992; Huffman and Laird, 1994; Porter and Kibler, 1986]. QUALITY falls in the latter category, although the expert does not critique problem solving decisions directly but only the final plan in order to make oblivious the planning algorithm and representation language to the expert. Learning apprentices rely on the interaction with an expert in different degrees ranging from a non-intrusive observation (for example [Mitchell *et al.*, 1990; Dent *et al.*, 1992; Wang, 1995]) to direct advice at decision points [Golding *et al.*, 1987; Laird *et al.*, 1990] or illustrative examples supplied by the expert within the system's current knowledge and abilities [Golding *et al.*, 1987]. QUALITY can work autonomously (without human expert interaction) as the quality metric is known to the planner and it can always solve the problem from first principles.

Search control knowledge can also be acquired by knowledge acquisition methods [Gruber, 1989; Joseph, 1992], by extracting from the human experts justifications for their choices. QUALITY instead allows a fully automated acquisition task by using a purely machine-learning approach. The expert, if present, does not need to make explicit the reasons for the choices of plan steps. In addition, its focus is on quality-enhancing control knowledge.

CABINS [Sycara and Miyashita, 1994] and the system described in [Hamazaki, 1992] acquire user preferences to generate good schedules. Both pieces of work are motivated by the fact that which is a better schedule depends on user preferences, which balance conflicting objectives and tradeoffs, and are difficult to express as a cost function. QUALITY assumes instead that a quality metric is available and therefore can function autonomously without the human expert. In future work we would like to loosen that assumption, given what Sycara, Miyashita, and Hamazaki suggest about how quality knowledge is represented in certain domains, and explore how the quality metric can be built from the expert's interaction.

7 Conclusion

This paper has presented a general framework to solve the problem of generating good plans in AI planning systems which represents knowledge about plan quality as operational, planning-time search-control knowledge and automatically acquires such control knowledge from planning experience. This general framework has been implemented in the QUALITY architecture. QUALITY can benefit from the interaction with a human expert in the application domain. This interaction is at the level of plan steps. Our objective was that the expert could remain oblivious to the planning algorithm and representation language, thus reducing the knowledge engineering

Problem set (30 probs per set)	1	2	3	4	5	6
# problems with improvement	9	24	14	25	30	20
Cost decrease (improved probs)	49%	55%	40%	28%	57%	20%
Max cost decrease (in a single prob)	63%	71%	87%	88%	96%	34%

Table 2: Quality (with metric Q_1) improvements obtained by the learned control rules.

effort of acquiring quality-enhancing control knowledge.

The interaction of the learner with the domain expert through the Interactive Plan Checker in QUALITY can be improved in several different directions, which include its integration with PRODIGY's graphical user interface. As planners address more realistic problems and move towards more interactive, mixed-initiative approaches, the role of user interfaces increases. "The machine learning problem is really one of interaction. The key is to establish an interaction language for which the human teacher finds it easy to convey notions of interest, and at the same time, for which the computer as a student is capable of learning the appropriate inferences."[Lieberman, 1994] QUALITY closes the gap between the expert advice and the level of inferences required by the learner. Machine learning for planning systems can get increased leverage by a careful interaction with the experts to whom they support, especially when solving problems closer to the real world. We plan to further explore that interaction in the larger context of PRODIGY as an integrated planning and learning system.

References

[Dent *et al.*, 1992] Lisa Dent, Jesús G. Boticario, John McDermott, Tom Mitchell, and David Zabowski. A personal learning apprentice. In *Proceedings of the National Conference on Artificial Intelligence*, pages 96–103, San Jose, CA, 1992.

[Etzioni, 1990] Oren Etzioni. *A Structural Theory of Explanation-Based Learning*. PhD thesis, Carnegie Mellon University, School of Computer Science, Pittsburgh, PA, 1990. Also appeared as Technical Report CMU-CS-90-185.

[Gil, 1991] Yolanda Gil. A specification of process planning for PRODIGY. Technical Report CMU-CS-91-179, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, August 1991.

[Golding *et al.*, 1987] Andrew Golding, Paul S. Rosenbloom, and John E. Laird. Learning general search control from outside guidance. In *Proceedings of the Tenth International Conference on Artificial Intelligence*, pages 334–337, Milan, Italy, 1987.

[Gratch *et al.*, 1993] Jonathan Gratch, Steve Chien, and Gerald DeJong. Learning search control knowledge for deep space network scheduling. In *Machine Learning. Proceedings of the Tenth International Conference*, pages 135–142, Amherst, MA, June 1993. Morgan Kaufmann.

[Gruber, 1989] Thomas R. Gruber. Automated knowledge acquisition for strategic knowledge. *Machine Learning*, 4:293–336, 1989.

[Hamazaki, 1992] Takashi Hamazaki. High quality production scheduling system. In *Proceedings of SPICIS 92*, pages 195–200, 1992.

[Huffman and Laird, 1994] Scott B. Huffman and John E. Laird. Learning from higly flexible tutorial instruction. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 506–512, Seattle, WA, July 1994. AAAI Press/The MIT Press.

[Iwamoto, 1994] Masahiko Iwamoto. A planner with quality goal and its speedup learning for optimization problem. In *Proceedings of the Second International Conference on AI Planning Systems*, pages 281–286, Chicago, IL, 1994.

[Joseph, 1992] Robert L. Joseph. *Graphical Knowledge Acquisition for Visually-Oriented Planning Domains.* PhD thesis, Carnegie Mellon University, School of Computer Science, August 1992. Also appeared as Technical Report CMU-CS-92-188.

[Laird *et al.*, 1990] John E. Laird, Michael Hucka, Eric S. Yager, and Christopher M. Tuck. Correcting and extending domain knowledge using outside guidance. In Bruce W. Porter and Ray J. Mooney, editors, *Machine Learning: Proceedings of the Seventh International Conference, ML90*, pages 235–243, Austin, TX, June 1990. Morgan Kaufmann.

[Lieberman, 1994] Henry Lieberman. A user interface for knowledge acquisition from video. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 527–534, Seattle, WA, July 1994. AAAI Press/The MIT Press.

[Martin and Redmond, 1989] Joel D. Martin and Michael Redmond. Acquiring knowledge for explaining observed problem solving. *SIGART Newsletter, Knowledge Acquisition Special Issue*, 108:77–83, April 1989.

[Minton *et al.*, 1989] Steven Minton, Jaime G. Carbonell, Craig A. Knoblock, Daniel R. Kuokka, Oren Etzioni, and Yolanda Gil. Explanation-based learning: A problem-solving perspective. *Artificial Intelligence*, 40:63–118, 1989.

[Minton, 1988] Steven Minton. *Learning Effective Search Control Knowledge: An Explanation-based Approach.* Kluwer Academic Publishers, Boston, MA, 1988. PhD thesis available as Technical Report CMU-CS-88-133, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA.

[Mitchell *et al.*, 1986] Tom M. Mitchell, Richard Keller, and Smadar Kedar-Cabelli. Explanation-based generalization: A unifying view. *Machine Learning*, 1(1), 1986.

[Mitchell *et al.*, 1990] Tom M. Mitchell, Sridhar Mahadevan, and Louis I. Steinberg. LEAP: A learning apprentice system for VLSI design. In Yves Kodratoff and Ryszard Michalski, editors, *Machine Learning: An Artificial Intelligence Approach*, volume III, pages 271–289. Morgan Kaufmann, San Mateo, CA, 1990.

[Pérez and Etzioni, 1992] M. Alicia Pérez and Oren Etzioni. DYNAMIC: A new role for training problems in EBL. In D. Sleeman and P. Edwards, editors, *Machine Learning: Proceedings of the Ninth International Conference, ML92*, pages 367–372. Morgan Kaufmann, San Mateo, CA., 1992.

[Pérez, 1995] M. Alicia Pérez. Learning search control knowledge to improve plan quality. Technical Report CMU-CS-95-175, School of Computer Science, Carnegie Mellon University, Pittsburgh, PA, July 1995. PhD thesis.

[Porter and Kibler, 1986] Bruce Porter and Dennus Kibler. Experimental goal regression: A method for learning problem-solving heuristics. *Machine Learning*, 1:249–286, 1986.

[Sycara and Miyashita, 1994] Katia Sycara and Kazuo Miyashita. Case-based acquisition of user preferences for solution improvement in ill-structured domains. In *Proceedings of the Twelfth National Conference on Artificial Intelligence*, pages 44–49, Seattle, WA, July 1994. AAAI Press/The MIT Press.

[Tadepalli, 1989] Prasad Tadepalli. Lazy explanation-based learning: A solution to the intractable theory problem. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, pages 694–700, Detroit, MI, 1989.

[Tadepalli, 1990] Prasad Tadepalli. *Tractable Learning and Planning in Games.* PhD thesis, Rutgers, The State University of New Jersey, Department of Computer Science, May 1990. Technical Report ML-TR-31.

[Tecucci, 1992] Gheorghe D. Tecucci. Automating knowledge acquisition as extending, updating, and improving a knowledge base. *IEEE Transactions on Systems, Man and Cybernetics*, 22(6), November/ December 1992.

[Veloso *et al.*, 1995] Manuela Veloso, Jaime Carbonell, M. Alicia Pérez, Daniel Borrajo, Eugene Fink, and Jim Blythe. Integrating planning and learning: The PRODIGY architecture. *Journal of Experimental and Theoretical Artificial Intelligence*, 7(1), January 1995.

[Wang, 1995] Xuemei Wang. Learning by observation and practice: An incremental approach for planning operator acquisition. In *Proceedings of the Twelfth International Conference on Machine Learning*, Tahoe City, CA, 1995.

[Wilkins, 1988] David C. Wilkins. Knowledge base refinement using apprenticeship learning techniques. In *Proceedings of the National Conference on Artificial Intelligence*, pages 646–651, St. Paul, MN, 1988.