In Proceedings of the Second International Conference on AI Planning Systems, June 1994, Chicago, IL

Control Knowledge to Improve Plan Quality*

M. Alicia Pérez and Jaime G. Carbonell
School of Computer Science
Carnegie Mellon University
5000 Forbes Ave
Pittsburgh PA 15213-3891
{aperez,carbonell}@cs.cmu.edu

Abstract

Generating production-quality plans is an essential element in transforming planners from research tools into real-world applications. However most of the work to date on learning planning control knowledge has been aimed at improving the *efficiency of planning*; this work has been termed "speed-up learning". This paper focuses on learning control knowledge to guide a planner towards better solutions, i.e. to improve the *quality of the plans* produced by the planner, as its problem solving experience increases. We motivate the use of quality-enhancing search control knowledge and its automated acquisition from problem solving experience. We introduce an implemented mechanism for learning such control knowledge and some of our preliminary results in a process planning domain.

Introduction

Most research on planning so far has concentrated on methods for constructing sound and complete planners that find a satisficing solution, and on how to find such solution in an efficient way (Chapman 1987; McAllester & Rosenblitt 1991; Peot & Smith 1993). Accordingly most work on machine learning in the context of planning systems has focused on two types of learning goals (Pérez 1994): learning driven by domain goals, when the domain knowledge is incomplete or inaccurate (Gil 1992; Wang 1994; Shen 1989), and learning driven by planning efficiency goals, or "speed-up learning" (Mitchell, Keller, & Kedar-Cabelli 1986; Minton *et al.* 1989; Etzioni 1990; Pérez & Etzioni 1992; Knoblock 1990; Veloso 1992; Gratch, Chien, & DeJong 1993). This paper focuses on a third kind of learning goals: to improve the *quality of plans* produced by the planner. The learner builds new control knowledge by analyzing previous experience, and that knowledge guides subsequent search towards better plans. Learning is triggered by the possibility of improving the quality of the solution obtained with the planner's current control knowledge. The planner's domain theory is extended with a function that evaluates the quality of a plan. In particular the metric we use is related to plan execution cost, or the efficiency of the plans when they are executed, instead of the efficiency of the process of finding those plans. Learning consists in a *reformulation* task of the plan evaluation function into control knowledge usable by the planner at problem solving time. The next sections present the motivation for using and learning control knowledge to obtain quality plans, discuss some metrics for plan quality, and introduce a mechanism for learning such control knowledge and preliminary results of the full implementation of such mechanism. Finally, we briefly discuss some related work and present some conclusions.

*This research is partially 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. Views and conclusions contained in this document are those of the authors and should not be interpreted as necessarily representing official policies or endorsements, either expressed or implied, of Wright Laboratory or the United States Government. The first author holds a scholarship from the Ministerio de Educación y Ciencia of Spain.

Plan Quality and Control Knowledge

Generating production-quality plans is an essential element in transforming planners from research tools into real-world applications. For example, in the process planning domain plan quality is crucial in order to minimize both resource consumption and execution time.

Planning goals rarely occur in isolation and the interactions between conjunctive goals have an effect in the quality of the plans that solve them. In (Pérez & Veloso 1993) we argued for a distinction between *explicit goal interactions* and *quality goal interactions*. Explicit goal interactions are explicitly represented as part of the domain knowledge in terms of preconditions and effects of the operators. They enforce particular goal orderings so that the planner may be able to produce a solution to the problem. As a particular problem may have many different solutions the interactions may be the result of the particular problem solving path explored. For example, in a machine-shop scheduling domain, when two identical machines are available to achieve two goals, these goals may interact, if the problem solver chooses to use just one machine to achieve both goals, as it will have to wait for the machine to be idle. If the problem solver uses the two machines instead of just one, then the goals do not interact in this particular solution. These interactions are related to plan quality as the use of resources dictates the interaction between the goals. Whether one alternative is better than the other depends on the particular quality measure used for the domain. The control knowledge that guides the planner to solve these interactions is harder to learn automatically, as the domain theory does not encode these quality criteria.

It can be argued that the heuristics to guide the planner towards better solutions could be incorporated into the planner itself, as in the case of removing unnecessary steps or using breadth-first search when plan length is used as evaluation function. However in some domains plan length is not an accurate metric of plan quality, as different operators have different costs. Take an example in the process planning domain. The goal of process planning is to produce plans for machining parts given their specifications [1]. In order to perform an operation on a part, certain set-up operations are required as the part has to be secured to the machine table with a holding device in certain orientation, and in many cases the part has to be clean and without burrs from preceding operations. The appropriate tool has to be selected and installed in the machine as well. As an example, Figure 1 (a) shows a machine set-up to drill a hole in a part. Figure 1 (b) shows the specification of the sides and dimensions of a rectangular part. Suppose the planner is

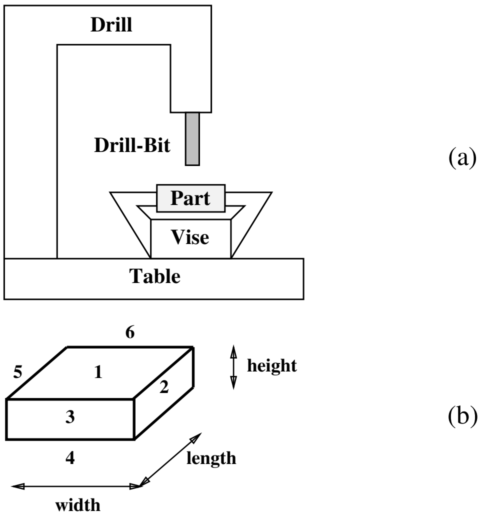

Figure 1: *(a) An example of set-up in the machining domain (from (Joseph 1992)). In this example the holding device is a vise, the machine a drill press, and the tool a drill-bit. (b) Dimensions and sides of a part.*

given as a goal to drill two holes on different sides (1 and 4) of a part, where each hole requires a different diameter drill bit. In the initial state the part is being held ready to drill the hole on side 4, but the drill-spindle holds the drill bit suitable for the hole on side 1. One possible plan starts by releasing the part and holding it again in the orientation suitable for the hole in side 1, while keeping the tool in the drill spindle, and then drilling that hole. Other plan starts by switching the tool while keeping the part in place, and drilling first the hole in side 4. Figure 2 shows these two alternative plans. Both plans have the same length but may have different quality. For example if the tool can be switched automatically but holding the part requires human assistance (Hayes 1990), the cost of plan (a) is higher than that of (b).

This knowledge about plan quality, cost in the example, is not encoded in the domain definition (set of operators and inference rules) and might vary over time. If it is given as a domain-dependent plan evaluation function, it must be *translated into knowledge operational at planning time*, capturing the fact that the plan and search tree are only partially available when a decision has to be made. The translated knowledge can be expressed as search-control knowledge in terms of the problem solving state and meta-state, such as which operators and bindings have been chosen to achieve the goals or which are the candidate goals to expand (we present later a more detailed example). In the previous example the goal of drilling the hole on side 4 has to be achieved before the goal of drilling the hole on side 1. Note that for a different plan evaluation function, namely one in which switching the tool is more expensive than holding the part again, the order to achieve the two goals should be the opposite.

Measuring Plan Quality Plan quality metrics can be classified in three large groups ((Pérez & Carbonell 1993) contains a detailed taxonomy):

- Execution cost. Some of the factors that affect a plan's execution cost can be computed by summing over all the steps or operators in the plan, that is $C_{total} = \Sigma c_i$ where C_{total} is the total cost of executing the plan and c_i is the cost for each operator. c_i can be the operator execution time, the cost of the resources used by the step, or 1 if the measure is simply the length of the plan or total number of actions. Several factors that influence a plan's execution cost are the execution time, the material resources, or the agent skill requirements (which refers to the extent to which an agent can perform an action; plans with less agent skill requirements are typically less expensive).
- Plan robustness or ability to respond well under changing or uncertain conditions.
- Other factors that capture the satisfaction of the client with the solution itself (for example the accuracy of the result, or the comfort it provides to the user). These in some cases are hard to quantify.

The work on learning quality-enhancing control knowledge described in this paper concentrates on quality metrics related to plan execution cost, and expressed as an evaluation function additive on the cost of the individual operators. These functions are linear and do not capture the existence of tradeoffs between different quality factors.

Learning to Improve Plan Quality

To date no work has focused on automatically acquiring knowledge to improve plan quality, i.e. on building learning systems that address the third type of learning goals presented in above. This section presents work in progress

[1](Gil 1991) describes the implementation of this domain in the PRODIGY planner, concentrating on the machining, joining, and finishing steps of production manufacturing.

```
release drill1 vise1 part51
hold drill1 vise1 part51 side1 side3-side6
drill-with-twist-drill drill1 twist-drill5
  vise1 part51 hole1 side1 side3-side6 1 1/8
remove-tool drill1 twist-drill5
put-tool-dr drill1 twist-drill13
release drill1 vise1 part51
remove-burrs part51 brush1
clean part51
hold drill1 vise1 part51 side4 side2-side5
drill-with-twist-drill drill1 twist-drill13
  vise1 part51 hole3 side4 side2-side5 3 1/4
```

(a)

```
remove-tool drill1 twist-drill5
put-tool-dr drill1 twist-drill13
drill-with-twist-drill drill1 twist-drill13
  vise1 part51 hole3 side4 side2-side5 3 1/4
remove-tool drill1 twist-drill13
put-tool-dr drill1 twist-drill5
release drill1 vise1 part51
remove-burrs part51 brush1
clean part51
hold drill1 vise1 part51 side1 side3-side6
drill-with-twist-drill drill1 twist-drill5
  vise1 part51 hole1 side1 side3-side6 1 1/8
```

(b)

Figure 2*: Two solutions for a process planning problem with the same length but different cost, as switching the tool is cheaper than setting-up the part.*

to address this problem. Our goal is to have a system that improves over experience the quality of the plans it generates by acquiring in a fully automated fashion control knowledge to guide the search. Figure 3 shows the architecture of the current system, fully implemented and built on top of the PRODIGY nonlinear planner (Carbonell & The PRODIGY Research Group 1992).

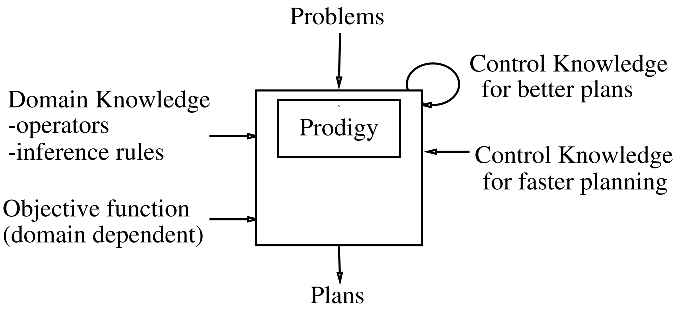

Figure 3*: Architecture of a system to learn control knowledge to improve the quality of plans.*

The learning algorithm is *given* a domain theory (operators and inference rules) and a domain-dependent objective function that describes the quality of the plans. It is also given problems to solve in that domain. The algorithm analyzes the problem-solving episodes by comparing the search trace for the planner solution given the current control knowledge, and another search trace corresponding to a better solution (*better* according to the evaluation function). The latter search trace is obtained by letting the problem solver search further until a better solution is found, or by asking a human expert for a better solution and then producing a search trace that leads to that solution. The algorithm explains why one solution is better than the other and its *output* is search control knowledge that leads future problem solving towards better quality plans. Two points are worth mentioning:

- Learning is driven by the existence of a better solution and a failure of the current control knowledge to produce it.
- There is a change of representation from the knowledge about quality encoded on the objective function into knowledge that the planner may use at problem solving time (as mentioned before the plan and search tree are only partially available when a decision has to be made).

We do not claim that this control knowledge will necessarily guide the planner to find optimal solutions, but that the quality of the plans will incrementally improve with experience, as the planner sees new interesting problems in the domain.

Example

Take the example in the process planning domain of Figure 4. The goal is to produce a part with a given height and

```
(state (and (diameter-of-drill-bit twist-drill6 9/64)
            (material-of part5 ALUMINUM) (size-of part5 LENGTH 5)
            (size-of part5 HEIGHT 3) (size-of part5 WIDTH 3)...))

(goal ((<part> PART))
      (and (size-of <part> HEIGHT 2)
           (has-spot <part> hole1 side1 1.375 0.25)))
```

Figure 4*: Example problem, showing the goal and a subset of the initial state.*

with a spot hole on its side 1. The system also knows about a domain dependent function that evaluates the quality of the plans. The one we use in this example is additive on the plan operators and intends to capture the cost of the set-up operations required to machine the part. As we mentioned earlier these operations have different costs.

Figure 5 (a) shows the solution obtained by the planner with the current control knowledge. The orientation in which a part is held and then machined is given by the side that faces up, and the pair of sides that face the holding device. For example, the operation `(drill-with-spot-drill drill1 spot-drill1 vise1 part5 hole1 side1 side2-side5)` indicates that `hole1` will be drilled in `part5` on the drill press `drill1` using as a tool `spot-drill1`, while the part is being held with its side 1 up, and sides 2 and 5 facing the holding device, `vise1`.

Milling machines can be used both to reduce the size of a part, and to drill holes and spot-holes on it. Therefore this solution can be improved, according to the evaluation function, by using the milling machine to drill the spot-hole, and by picking the same orientation, machine and holding device for the drill and mill operations so that the part does not have to be released and held again. These changes correspond at problem-solving

```
put-tool-dr drill1 spot-drill1
put-holding-device-dr drill1 vise1
clean part5
put-on-machine-table drill1 part5
hold drill1 vise1 part5 side1 side2-side5
drill-with-spot-drill drill1 spot-drill1 vise1
   part5 hole1 side1 side2-side5

put-tool-mm milling-mach1 plain-mill1
release drill1 vise1 part5
remove-holding-device drill1 vise1
put-holding-device-mm milling-mach1 vise1
remove-burrs part5 brush1
clean part5
put-on-machine-table milling-mach1 part5
hold milling-mach1 vise1 part5 side1 side3-side6
face-mill milling-mach1 part5 plain-mill1 vise1
   side1 side3-side6 height 2
```

(a)

```
put-tool-mm milling-mach1 spot-drill1
put-holding-device-mm milling-mach1 vise1
clean part5
put-on-machine-table milling-mach1 part5
hold milling-mach1 vise1 part5 side1 side3-side6
drill-with-spot-drill-mm milling-mach1
   spot-drill1 vise1 part5 hole1 side1 side3-side6
remove-tool milling-mach1 spot-drill1
put-tool-mm milling-mach1 plain-mill1

face-mill milling-mach1 part5 plain-mill1
   vise1 side1 side3-side6 height 2
```

(b)

Figure 5*: (a) Solution obtained by the planner guided by the current control knowledge. (b) A better solution according to the evaluation function, input by a human expert.*

time to an operator decision (`drill-with-spot-drill` vs `drill-with-spot-drill-in-milling-machine`) and a bindings decision (bindings for `drill-with-spot-drill-in-milling-machine`)[2]. In this case a domain expert inputs the improved solution, shown in Figure 5 (b) and the system produces a search trace that leads to that solution. As mentioned above what drives the learning process is the planner's lack of knowledge to achieve the better solution, and the existence of decision points where that knowledge can be used. Next the learner compares both problem solving traces to find where the savings come from. In this example, the cost of achieving the subgoals of drilling the spot-hole is reduced in solution (b) because part of the set-up is common with the milling operation. The learner explains why these savings occur and makes that description operational so it uses knowledge available at the decision points. The learning mechanism comes up with the control rules in Figure 6 that guide the planner in the decisions mentioned above.

The explanation is built from a single example, explaining why one alternative at the decision point is preferred to the other. Therefore the explanation is *incomplete* as it does not consider all possible hypothetical scenarios. For example, if in the above problem a third goal is to have `hole1` counterbored, and the counterboring can only be performed by the drill-press, using the milling machine for the spot-hole may not be the best alternative as the part has to be set up also in the drill-press. However the rules in Figure 6 would still apply. The consequence of using incomplete explanations is learning over-general knowledge (Tadepalli 1989; Chien 1989). The system refines the learned knowledge incrementally, upon unexpected failures. By a failure we mean that the preferred alternative leads to a plan that can be improved. The refinement does not modify the applicability conditions of the learned rules but adds new rules (if needed) and sets priorities among rules.

[2]Control rules for goal preferences are learned from other problems.

```
(control-rule pref-drill-with-spot-drill-in-milling-machine30
 (if (and
      (current-goal (has-spot <part> <hole> <side> <loc-x> <loc-y>))
      (pending-goal
       (holding <mach> <holding-dev> <part> <side> <side-pair>))
      (type-of-object <mach> milling-machine)))
 (then prefer operator drill-with-spot-drill-in-milling-machine
                       drill-with-spot-drill))

(control-rule pref-bnds-drill-with-spot-drill-in-milling-machine31
 (if (and
      (current-goal (has-spot <part> <hole> <side> <loc-x> <loc-y>))
      (current-operator drill-with-spot-drill-in-milling-machine)
      (pending-goal
       (holding <mach4> <holding-dev5> <part> <side> <side-pair-6>))
      (or (diff <mach4> <mach1>) (diff <holding-dev5> <holding-dev2>)
          (diff <side-pair-6> <side-pair-3>))))
 (then prefer bindings
  ((<mach> . <mach4>)(<hd> . <holding-dev5>)(<sp> . <side-pair-6>))
  ((<mach> . <mach1>)(<hd> . <holding-dev2>)(<sp> . <side-pair-3>))))
```

Figure 6*: Search control rules learned from the problem in Figure 4.*

Empirical Results

We have implemented the mechanism described in the previous section and run preliminary experiments to evaluate the solution quality gained by the use of the learned control knowledge. Table 1 shows the effect of that knowledge on the solution cost, according to the above mentioned evaluation function [3], over 70 randomly-generated problems in the process planning domain. Each column corresponds to a set of 10 problems with common parameters (number and type of goals, parts, etc) which determine the problem difficulty and the usefulness of quality-enhancing control knowledge. In many of the training and test problems the planner did not require control knowledge to obtain a good solution and therefore for each problem set we have only recorded the problems where the solution was actually improved. The third row corresponds to the cost of the solutions obtained by the planner when it lacks any quality-enhancing control

[3]In the experiment plan quality was related to the set-up cost, and in the evaluation function all machining operations have the same cost, and the operations to set-up the part on the machine were more expensive than those to switch the tool.

knowledge. The fourth row shows the cost of the solutions obtained using the learned control rules. The rules were learned from a different set of 60 randomly generated problems with the same parameters than for sets 1 to 6 of Table 1. In only 41 of these problems the solution could be improved and therefore learning was invoked. The smaller quality improvement obtained for set 7 is due to the fact that the training phase did not include problems with similar parameters and therefore some of the control knowledge relevant to those problems had not yet been learned. The number of nodes was also reduced by using the learned control knowledge, due to shorter solution lengths. The total CPU time was also reduced, but still we plan to further analyze the cost of using learned knowledge and the possible tradeoff between the matching cost and the savings obtained by using the learned knowledge instead of doing a more exhaustive search until a reasonably good solution is found according to the evaluation function. We also plan to explore the effect of this learning mechanism on other domains and on other types of evaluation functions.

Table 1: *Improvement on the quality of the plans obtained for 70 randomly-generated problems in the process planning domain.*

Problem set (10 probs per set)	1	2	3	4	5	6	7
# problems with improvement	3	9	3	10	10	4	9
Without learned control knowledge	107	202	190	431	362	442	732
With learned control knowledge	91	132	166	350	220	409	665
Cost decrease	44%	48%	33%	24%	47%	17%	8%

Related Work

As we mentioned in the introduction, although there have been a number of systems that learn control knowledge for planning systems, most of them are oriented towards improving search efficiency and not plan quality. Some EBL for problem solving systems (Golding, Rosenbloom, & Laird 1987) learn from externally provided guidance. When used, the guidance takes the form of a plan reducing the need for the user to understand the problem solving process. However our method does not rely on such guidance, as since the quality evaluation function is available, the system continues searching for better solutions and then learns about which paths are better than others. R1-Soar (Rosenbloom *et al.* 1985) uses a similar approach for optimizing a computer configuration task. Chunking is used indistinctly to reduce the search and to prefer better solutions. Our method explains the difference in cost between the two paths rather than simply the union of what mattered along the two paths and therefore may be able to build more general rules. Most of the work on plan quality has been on the relationship between plan quality and goal interactions. (Wilensky 1983) explores such relationship in detail. Several planners for the process planning domain deal with the quality of plans (Hayes 1990; Nau & Chang 1985) using domain-dependent heuristics. Other work focuses on post-facto plan modification (Karinthi, Nau, & Yang 1992; Foulser, Li, & Yang 1992) by merging plans for individual subgoals taking advantage of helpful goal interactions. Our method uses instead control knowledge at problem solving time and is able to learn from experience in a domain-independent fashion. Our work is different from SteppingStone (Ruby & Kibler 1990) in that it heuristically decomposes a problem into simpler subproblems, and then learns to deal with the interactions that arise between them. (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. The quality goals are represented explicitly and based on the quality of the final state instead of that of the plan. This is equivalent to our use of a quality evaluation function. The learning method is similar to ours in that it compares two solutions of different quality, builds an explanation, and learns operator preferences. Our method however makes use of the quality evaluation function to build the explanation, and learns in addition bindings and goal preference rules. As in our case, the learned knowledge may be overgeneral and is refined upon further experience by learning more specific rules and preferences among them. The method does not allow user guidance as it uses exhaustive search until the quality goal is satisfied to find the best solution. This is possible because of the relatively small size of the search space of the examples used.

The problem of finding optimal plans has been attacked by decision theorists. However this problem is computationally very expensive. Simon introduced the idea of "satisficing" (Simon 1981) arguing that a rational agent does not always have the resources to determine what the optimal action is, and instead should attempt only to make good enough, to satisfy. We acknowledge the computational cost of finding the optimal behavior and do not claim that the acquired control knowledge will necessarily guide the planner to optimal plans, but that plan quality will improve incrementally over experience as the planner sees new interesting problems.

Conclusion

This paper described a mechanism for learning quality-enhancing search control knowledge from problem solving experience. The mechanism has been implemented in a system that acquires control rules in a fully automated fashion. The learning process translates knowledge about plan quality encoded in a domain-dependent plan evaluation function into control knowledge that the planner can use at problem solving time. The motivation for our work is the belief that generating production-quality plans is an essential element in transforming planners from research tools into real-world applications. To date not much research on planning and learning has approached this problem. The work presented here demonstrates that learning from problem solving experience is not only useful for improving planning efficiency but also can be an effective tool to generate better plans.

Acknowledgements

Thanks to Manuela Veloso and the anonymous reviewers for insightful comments on a draft of this paper. This manuscript is submitted for publication with the understanding that the U.S. Government is authorized to reproduce and distribute reprints for Governmental purposes, notwithstanding any copyright notation thereon.

References

Carbonell, J. G., and The PRODIGY Research Group. 1992. PRODIGY4.0: The manual and tutorial. Technical Report CMU-CS-92-150, School of Computer Science, Carnegie Mellon University. Editor: M. A. Pérez.

Chapman, D. 1987. Planning for conjunctive goals. *Artificial Intelligence* 32:333–378.

Chien, S. A. 1989. Using and refining simplifications: Explanation-based learning of plans in intractable domains. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 590–595.

Etzioni, O. 1990. *A Structural Theory of Explanation-Based Learning*. Ph.D. Dissertation, Carnegie Mellon University, School of Computer Science. Also appeared as Technical Report CMU-CS-90-185.

Foulser, D. E.; Li, M.; and Yang, Q. 1992. Theory and algorithms for plan merging. *Artificial Intelligence* 57:143–181.

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

Gil, Y. 1992. *Acquiring Domain Knowledge for Planning by Experimentation*. Ph.D. Dissertation, Carnegie Mellon University, School of Computer Science. Available as technical report CMU-CS-92-175.

Golding, A.; Rosenbloom, P. S.; and Laird, J. E. 1987. Learning general search control from outside guidance. In *Proceedings of the Tenth International Conference on Artificial Intelligence*, 334–337.

Gratch, J.; Chien, S.; and DeJong, G. 1993. Learning search control knowledge for deep space network scheduling. In *Machine Learning. Proceedings of the Tenth International Conference*, 135–142. Amherst, MA: Morgan Kaufmann.

Hayes, C. 1990. *Machining Planning: A Model of an Expert Level Planning Process*. Ph.D. Dissertation, The Robotics Institute, Carnegie Mellon University.

Iwamoto, M. 1994. A planner with quality goal and its speedup learning for optimization problem. In *Proceedings of the Second International Conference on AI Planning Systems*.

Joseph, R. L. 1992. *Graphical Knowledge Acquisition for Visually-Oriented Planning Domains*. Ph.D. Dissertation, Carnegie Mellon University, School of Computer Science.

Karinthi, R.; Nau, D. S.; and Yang, Q. 1992. Handling feature interactions in process planning. *Applied Artificial Intelligence* 6(4):389–415. Special issue on AI for manufacturing.

Knoblock, C. A. 1990. Learning effective abstraction hierarchies. In *Proceedings of the Eighth National Conference on Artificial Intelligence*.

McAllester, D., and Rosenblitt, D. 1991. Systematic nonlinear planning. In *Proceedings of the Ninth National Conference on Artificial Intelligence*, 634–639.

Minton, S.; Carbonell, J. G.; Knoblock, C. A.; Kuokka, D. R.; Etzioni, O.; and Gil, Y. 1989. Explanation-based learning: A problem-solving perspective. *Artificial Intelligence* 40:63–118.

Mitchell, T. M.; Keller, R.; and Kedar-Cabelli, S. 1986. Explanation-based generalization: A unifying view. *Machine Learning* 1(1).

Nau, D. S., and Chang, T.-C. 1985. Hierarchical representation of problem-solving knowledge in a frame-based process planning system. Technical Report TR-1592, Computer Science Department, University of Maryland.

Peot, M. A., and Smith, D. E. 1993. Threat-removal strategies for partial-order planning. In *Proceedings of the Eleventh National Conference on Artificial Intelligence*, 492–499.

Pérez, M. A., and Carbonell, J. G. 1993. Automated acquisition of control knowledge to improve the quality of plans. Technical Report CMU-CS-93-142, School of Computer Science, Carnegie Mellon University.

Pérez, M. A., and Etzioni, O. 1992. DYNAMIC: A new role for training problems in EBL. In Sleeman, D., and Edwards, P., eds., *Machine Learning: Proceedings of the Ninth International Conference (ML92)*. San Mateo, CA.: Morgan Kaufmann.

Pérez, M. A., and Veloso, M. M. 1993. Goal interactions and plan quality. In *Preprints of the AAAI 1993 Spring Symposium Series, Symposium on Foundations of Automatic Planning: The Classical Approach and Beyond*.

Pérez, M. A. 1994. The goal is to generate better plans. In *Working Notes of the AAAI 1994 Spring Symposium Series, Workshop on Goal-Driven Learning*.

Rosenbloom, P. S.; Laird, J. E.; McDermott, J.; Newell, A.; and Orciuch, E. 1985. *R1-Soar*: An experiment in knowledge-intensive programming in a problem-solving architecture. *IEEE Transactions on Pattern Analysis and Machine Intelligence* PAMI-7(5):561–569.

Ruby, D., and Kibler, D. 1990. Learning steppingstones for problem solving. In *Proceedings of the Darpa Workshop on Innovative Approaches to Planning, Scheduling and Control*, 366–373.

Shen, W.-M. 1989. *Learning from the Environment Based on Percepts and Actions*. Ph.D. Dissertation, Carnegie Mellon University, School of Computer Science. Available as technical report CMU-CS-89-184.

Simon, H. A. 1981. *The Sciences of the Artificial*. Cambridge, MA: The MIT Press, second edition.

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

Veloso, M. M. 1992. *Learning by Analogical Reasoning in General Problem Solving*. Ph.D. Dissertation, Carnegie Mellon University, School of Computer Science. Available as technical report CMU-CS-92-174.

Wang, X. 1994. Automatic acquisition of domain knowledge for planning by observation and practice. Thesis proposal, School of Computer Science, Carnegie Mellon University.

Wilensky, R. 1983. *Planning and Understanding*. Reading, MA: Addison-Wesley.