- **Convergence towards the correct learned knowledge.** These experiments illustrate the effect from incrementally refinin the learned knowledge. The results show that HAMLET in *lazy* mode converges to the correct set of control rules, as a function of the number of training problems seen.

The following subsections describe the experiments and discuss the results. In all experiments, all problems were randomly generated, different sets of problems were used for learning and testing, and the distributions used for both sets of problems were the same. Also, the *optimality parameter* was always set to `true`, quality is measured as the number of operators in the resulting plan, and every training phase began with no control rules, $\mathcal{C} = \phi$.

### 5.1. *Lazy vs. Eager Learning*

This experiment compares HAMLET's performance in *lazy* and *eager* learning modes. To show the difference between the two learning modes, we carried out the following experiment in the logistics domain. We trained HAMLET with the same 400 randomly-generated problems in *eager* and in *lazy* modes.[14] The learning problems were 200 one-goal problems involving up to four packages, and 200 two-goal problems involving up to fiv packages. In *lazy* mode, 19 rules were learned, while in *eager* mode, it learned 98 rules.

Then we tested the performance of the learned rules on 475 randomly generated problems from the same distribution as the learning set. The testing problems were of increasing complexity, generated by varying the number of goals in the problems from 1 to 10, and the maximum number of packages from 5 up to 20. The time bound given for all experiments reported in this article is $150 \times (1+\text{mod}(\text{number-goals},10))$ seconds.[15] The results are shown in Tables 1, 2 and 3. In Table 1 the f rst column shows the number of problems solved by Base-Level PRODIGY (281), and using the learned control rules (239). The second column shows the total time spent, where the time of the unsolved problems is the time bound. The rest of the columns show the comparison using only the problems that were solved by both configu ations. For instance, the third column shows the number of problems in which the solutions provided by Base-Level PRODIGY were better (i.e., according the the quality measure used) than using the learned control rules (eight cases) compared to the number of problems in which using the rules achieved better solutions than Base-Level PRODIGY (31 cases). The same applies for the other two columns. Table 2 shows the results using the rules learned in *lazy* mode

[14] *Eager* mode means calling HAMLET'algorithm (Figure 4) with the parameter $L = eager$, while *lazy* mode means calling HAMLET with $L = lazy$.

[15] We also tested using several time bounds, and the results were similar to the ones presented here.

using the same set of problems, and Table 3 shows the comparison between the two learning modes.

*Table 1*. Comparison of performance between Base-Level PRODIGY4.0 and PRODIGY4.0 using HAMLET's learned rules in *eager* mode for the logistics domain.

| Rules Used | Solved Problems | Time (seconds) | Solved by both (197 problems) | | |
|---|---|---|---|---|---|
| | | | Better solutions | Solution length | Nodes explored |
| no rules | 281 | 58900 | 8 | 1208 | 7699 |
| eager (98 rules) | 239 | 72699 | 31 | 1164 | 5560 |

*Table 2*. Comparison of performance between Base-Level PRODIGY4.0 and PRODIGY4.0 using HAMLET's learned rules in *lazy* mode for the logistics domain.

| Rules Used | Solved problems | Time (seconds) | Solved by both (278 problems) | | |
|---|---|---|---|---|---|
| | | | Better solutions | Solution length | Nodes explored |
| no rules | 281 | 58900 | 2 | 2995 | 16366 |
| lazy (19 rules) | 334 | 44091 | 89 | 2763 | 13030 |

*Table 3*. Comparison of performance between PRODIGY4.0 using HAMLET's learned rules in *lazy* and *eager* modes for the logistics domain.

| Rules Used | Solved problems | Time (seconds) | Solved by both (228 problems) | | |
|---|---|---|---|---|---|
| | | | Better solutions | Solution length | Nodes explored |
| eager (98 rules) | 239 | 72699 | 12 | 1593 | 7810 |
| lazy (19 rules) | 334 | 44091 | 44 | 1474 | 6802 |

**Discussion.** In *lazy* learning mode, HAMLET performs more efficiently and yields higher quality solutions. With respect to efficiency, the number of solved problems increases from 50% up to 70%, while the time spent on solving the problems also decreases. In addition, the solutions provided are much better. More specifically, in *eager* mode, it performs worse for eight problems and better for 31. In *lazy* mode, it performs worse in only two problems, and better in 89! We believe HAMLET's lazy behavior is responsible for such good results, given that it learns only when it seems clear that a decision was the best one in any node in the search tree, and was not chosen by the default problem solver. That is, the goal of HAMLET is to provide the problem solver with a learned way of controlling when not to follow the

default procedure, thus letting the problem solver decide when the default decision is correct.

There is also an important difference in the behavior of both modes: the number of learned rules. Since the *eager* mode learns from every decision, it learns many more rules, and many more types of rules. For instance, suppose that operator $O_1$ appears in the domain description before operator $O_2$, and both are relevant to the same goal $g$. Then the default behavior of PRODIGY4.0 is that, when it does not have control knowledge to know which operator to use, it uses the first operator that appears in the domain description. Thus, it will always try $O_1$ before $O_2$. If $O_1$ is always the best operator to achieve the goal $g$, then the lazy mode of HAMLET will never learn a control rule for that decision, while the eager mode will learn a rule each time that it uses $O_1$ and succeeds in achieving a solution. Therefore, eager mode suffers more from the utility problem than the lazy mode as can be observed in these tables in the time columns.

With respect to learning time, learning in *lazy* mode is a relatively fast process, since it generates fewer rules, and, therefore, spends a short time learning. The time spent on learning after every PRODIGY4.0 run on a problem is between two and ten times less than PRODIGY's search time. In contrast, learning eagerly can be very slow, since the refinement procedure needs to match many more rules against each other. It might take up to 20 times PRODIGY's search time.

We have been working on many experiments to find additional evidence of these two slow down phenomena (i.e., number of rules and learning time), and we have found that the eager mode does not work for most domains. It overloads the system with control rules, not allowing HAMLET to finish the learning phase due to memory problems. A potential advantage of *eager* mode over *lazy* mode might be that the *lazy* mode requires the whole search tree to be expanded for learning control rules, while *eager* mode can learn as soon as any solution has been found. Therefore, *lazy* mode should be used for domains for which learning in easy problems scales well to complex problems. We have found empirically that this is true for most of our planning domains.

Finally, if we compare the solution length and the number of nodes of both modes against not using the rules, we see that in *eager* mode it saves 3.6% in solution length and 27.8% in number of nodes, while in *lazy* mode it saves 7.7% in solution length and 20.4% in number of nodes. The performance is almost doubled for *lazy* mode in solution length, and better for *eager* mode in number of nodes. The reason why the savings in number of nodes is bigger for *eager* mode is due to the number of control rules it has, which prune more of the search space.

## 5.2. *Improving Efficiency and Solution Quality*

To show that HAMLET improves not only the efficiency of the base-level problem solver, PRODIGY4.0, but also improves the quality of the solutions provided by PRODIGY4.0 with the learned knowledge, we have carried out experiments in the logistics transportation domain, described in Subsection 3.4, and in blocksworld. We use the blocksworld to show a domain in which HAMLET improves mostly the efficiency of the problem solver, and logistics as a domain in which it also greatly improves the quality of the solutions provided by the problem solver. In this last domain the quality of the solutions is an important aspect, due to the possible unnecessary movements of the carriers. HAMLET learns knowledge that allows the planner to generate effective solutions for transporting packages.

**Blocksworld results.** We trained HAMLET with 200 simple randomly generated problems of one and two goals, and up to ten blocks. HAMLET generated 11 control rules. Then, we randomly generated 375 testing problems of increasing complexity. Table 4 shows the results of those tests. We varied the number of goals in the problems from one up to ten, and the maximum number of blocks from five up to 50.

*Table 4*. Results on increasingly complex problems in the blocksworld domain.

| Test sets | | Solved problems | | Solved by both (174 problems, 46.4%) | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Better solutions | | Solution length | | Nodes explored | |
| Goals | Problems | without rules | with rules | without rules | with rules | without rules | with rules | without rules | with rules |
| 1 | 100 | 68 | 100 | 0 | 0 | 245 | 245 | 9227 | 1187 |
| 2 | 100 | 56 | 92 | 1 | 2 | 196 | 193 | 3924 | 1258 |
| 5 | 100 | 45 | 80 | 1 | 3 | 188 | 180 | 4085 | 1895 |
| 10 | 75 | 13 | 34 | 0 | 1 | 76 | 72 | 11048 | 313 |
| Totals | 375 | 182 | 306 | 2 | 6 | 705 | 690 | 28284 | 4653 |
| % | | 48.5% | 81.6% | 1.1% | 3.4% | | | Ratio | 6.1 |

**Discussion.** The results show a remarkable improvement on the Base-Level PRODIGY solver performance when using the learned control rules. As an example, it increases the number of solved problems from 48.5% up to

81.6%, and expands six times fewer number of nodes.[16] It also generates better solutions using the rules in six occasions, while Base-Level PRODIGY only generates better solutions in two occasions. These results on solution quality are not as impressive as in the logistics domain (detailed below) due to the fact that, in the blocksworld task, most problems have the same number of operators in all possible solutions to the same problem.

**Logistics results.** We trained HAMLET in *lazy* mode with 400 simple randomly generated problems of one and two goals, and up to three cities and fi e packages. HAMLET generated 26 control rules. Then, we randomly generated 525 testing problems of increasing complexity. Table 5 shows the results of those tests. We varied the number of goals in the problems from one up to 50, and the maximum number of packages from f ve up to 50. Problems with 20 and 50 goals pose very complex problems to f nd good or optimal solutions, even for humans.

*Table 5*. Results on increasingly complex problems in the logistics domain.

| Test sets | | Solved problems | | Solved by both (279 problems, 53.14%) | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | | | Better solutions | | Solution length | | Nodes explored | |
| Goals | Problems | without rules | with rules | without rules | with rules | without rules | with rules | without rules | with rules |
| 1 | 100 | 95 | 100 | 0 | 11 | 327 | 307 | 2097 | 1569 |
| 2 | 100 | 85 | 94 | 0 | 25 | 528 | 479 | 3401 | 2308 |
| 5 | 100 | 56 | 82 | 1 | 33 | 865 | 777 | 5170 | 3463 |
| 10 | 100 | 32 | 68 | 1 | 24 | 770 | 668 | 3482 | 2941 |
| 20 | 75 | 13 | 39 | 0 | 10 | 505 | 455 | 2216 | 1924 |
| 50 | 50 | 1 | 10 | 0 | 0 | 34 | 34 | 143 | 141 |
| Totals | 525 | 282 | 393 | 2 | 103 | 3029 | 2720 | 16509 | 12346 |
| % | | 53.7% | 74.9% | 0.7% | 36.9% | | | Ratio | 1.3 |

**Discussion.** The results show again a remarkable increase in the number of solved problems using the learned rules, as well as a large number of problems (103) in which the solution generated using the rules is of better quality, compared to the two problems in which Base-Level PRODIGY produced a

[16] In 10 goal problems using the learned rules, it expands 35 times fewer number of nodes in the search tree.

solution of better quality. The running times decreased using the rules, but not significan ly.

Learning control knowledge to improve problem solving performance is a potential source of inneff ciency due to the tradeoff between the savings obtained and the cost of using the learned knowledge. This is generally known as the *utility problem* in speedup learning (Minton 1988). Interestingly, HAMLET does not suffer substantially from the utility problem even with no special organization of the learned knowledge.[17] Due to its inductive step over the control rules of the same type, HAMLET keeps a small number of control rules, instead of retaining many variations of each rule type. This is one of the sources of power of the induction step applied to the bounded explained control rules. Nevertheless, we are currently developing effic ent methods for organizing and matching the learned control rules. We consider this organization essential to the overall learning process to avoid a potential utility problem due to inefficien matching (Doorenbos and Veloso 1993).

After analyzing why some problems were not solved, we concluded that some rules were not correct after the training phase. This fact led us to carry on the next set of experiments towards testing the convergence of the learning approach.

### 5.3. *Convergence to the Correct Control Knowledge*

The previous results are important in the sense that they show that the learned rules after 400 training problems perform well. But someone could ask if the learned knowledge improves over time when more training problems are given, or it begins to oscillate in a local maximum, adding and removing the preconditions of the control rules. This is an important issue for any machine learning system, especially for a *lazy* system as HAMLET, since HAMLET relies on incremental refine ent to correctly characterize the learned control rules. Therefore, it needs to converge as it sees more training examples.

To show that HAMLET improves with the number of training problems, we performed the following experiment: we trained HAMLET in *lazy* mode with 75 problems, 150 problems, and 400 problems in the logistics domain. Then, we tested the respective learned control rules on the same test set as before (525 problems of increasing complexity). Table 6 shows results that clearly illustrate the rules converge towards the correct behavior.

**Discussion.** After analyzing these results and the previous ones, we noted that there were problems in which HAMLET did not learn anything. These

[17] This is true even for *eager* mode that learns many more rules than *lazy* mode.

*Table 6.* Results on convergence in the logistics domain.

| Training problems | Solved problems | | Solved by both | | | | |
|---|---|---|---|---|---|---|---|
| | | | Better solutions | | Ratio Solution Length | Ratio Time | Ratio Nodes |
| | without rules | with rules | without rules | with rules | without/ with rules | without/ with rules | without/ with rules |
| 75 | 53.71% | 63.62% | 0.35% | 25.89% | 1.11 | 0.49 | 1 |
| 150 | 53.71% | 65.71% | 0.72% | 31.9% | 1.06 | 0.33 | 1.25 |
| 400 | 53.71% | 74.86% | 0.72% | 36.92% | 1.08 | 0.32 | 1.34 |

problems belong to two different groups. The first group are problems that were too easy. They were solved by PRODIGY with no search, or the default strategy made the right decisions in the first place. The second group of problems were too complicated, so PRODIGY could not expand the whole search tree in the given time bound (usually 100 seconds), and, therefore, HAMLET could not learn any control rule. Here, there is some underlying assumption (bias) that influence HAMLET's behavior: HAMLET **will learn better from medium difficulty problems**. Having this bias in mind, the training problems were randomly generated so that: they were not too easy, such as problems in which the goals are true in the initial state; and they were not too complicated, such as problems with more than two goals. We believe that this is not an over simplifying assumption: control knowledge learned from them will still be applicable to easier or more difficult problems. As we have shown, the learned knowledge, even coming from a "biased" training phase, transfers well to more complicated problems, and does not interfere in the default PRODIGY behavior with simpler problems.

Since training can benefit from well-suited training problems, we are currently developing a better training schema in which problems are not generated randomly, but are biased towards the kinds of problems from which HAMLET learns better. This domain-independent biased generation of training problems should improve the convergence of HAMLET, and also reduce the training effort. We also plan to analyze, for each domain, the number of training problems that will be needed to obtain a certain degree of accuracy, based on research from computational learning theory (Valiant 1984).

## 6. Related Work

Most previous lazy learning approaches have been inductive approaches, such as the work in instance-based (Aha et al. 1991), memory-based (Stanfill and

Waltz 1986), or exemplar-based learning (Porter et al. 1990). Only some of the work has been applied to planning, usually in the context of analogy or case-based reasoning (Hammond 1989; Hanks and Weld 1995; Kambhampati 1989; Kettler et al. 1994; Veloso 1994a, b). Most of this work concerns domain-specifi algorithms. Also, although these approaches demonstrated some useful lazy learning behavior, they did not, as we have, compare lazy and eager learning modes. The two main differences with these approaches are: control rules represent knowledge to control individual decisions, while cases chain multiple decisions together allowing therefore a global control of the planning process; and control rules fi e only if their antecedents fully match, while cases allow partial matching.

Many of the inductive systems require many examples for learning complex defin tions, since they do not use prior knowledge that can guide the search of generalized hypothesis. Some new techniques have been developed that use prior knowledge, but they are still mainly used for learning domain theories (e.g., (Quinlan 1990; Muggleton 1992)), instead of learning control knowledge.

Similar work on lazy learning includes Lazy Explanation-Based Learning, LEBL (Tadepalli 1989) and Lazy Partial Evaluation, LPE (Clark and Holte 1992). While LEBL refine the knowledge by introducing exceptions, HAMLET modifie the control rules themselves by adding or removing their applicability conditions. Also, LEBL applies to games, while we use HAMLET for general task planning. Finally, LEBL does not consider plan quality. In turn, LPE learns from all search paths, following a more *eager* approach than HAMLET's *lazy* mode, and has been used in a linear problem solver (Prolog) to solve contraint satisfaction problems, instead of applying it to nonlinear planning.

Most of the planners that learn follow an eager deductive approach. They try to eagerly and correctly explain the problem solving choices from a single episode or from a static analysis of the domain defin tion. These speedup learning systems are usually applied to problem solvers with the linearity assumption, such as the ones applied to logic programming problem solvers (Quinlan 1990; Zelle and Mooney 1993; Muggleton 1992), special-purpose problem solvers (Langley 1983; Mitchell et al. 1983), or other general-purpose linear problem solvers (Etzioni 1993; Fikes et al. 1972; Leckie and Zukerman 1991; Minton 1988; Pérez and Etzioni 1992). These problem solvers are known to be incomplete and incapable of find ng optimal solutions (Rich 1983; Veloso 1989).

If we remove the linearity assumption, we are dealing with nonlinear problem solvers. In this article we show that nonlinear problem solving offers new learning opportunities where domain-dependent control knowledge can be used to further improve not only the problem solver's performance but also

the quality of the solutions produced. Moreover, eagerly constructing correct explanations of the nonlinear problem solver's successes and failures *from a single example* is computationally expensive, since it is difficu t to defin the right language for describing the relations among goals when making decisions. Also, even if those needed predicates are kept, goal regression leads in nonlinear planning to control knowledge that is either overly general or overly specifi . Some approaches to learning in nonlinear planning are (Estlin and Mooney 1995; Bhatnagar 1992; Laird et al. 1986; Leckie and Zukerman 1991; Kambhampati and Kedar 1991; Pérez and Carbonell 1994; Ruby and Kibler 1992; Veloso 1994b; Veloso et al. 1995). The main difference with our approach is the lazy aspects of HAMLET, and the way in which learned knowledge is represented. While improving problem solving eff ciency has been studied frequently, learning to improve solution quality has only been recently pursued by some researchers, including (Pérez and Carbonell 1994; Ruby and Kibler 1992). We differ from Pérez and Carbonell's work in that HAMLET performs inductive refinemen of the control rules, and in the way positive examples are generated. Ruby and Kibler's approach differs in the knowledge representation of the learned control knowledge, since it is a case-based learner.

## 7. Conclusions

We have described a learning approach, HAMLET, that lazily acquires successful and failure control patterns for a nonlinear problem solver. Within HAMLET, *lazily* means the combination of three lazy aspects:

- It learns rules that are not provenly correct by bounding the explanation of the problem solving successes to a reduced set of features that explain why a certain decision is the best one. The explanation does not consider the whole search tree, nor does it try to prove that it is correct. Hence, this is a *lazy* learning approach.
- It incrementally refine learned control rules. Since the rules might not be correct, they might fail to (optimally) solve future problems. HAMLET does not eagerly try to f nd those erroneous applications of the control rules, but lazily waits until it fi ds a negative example. Also, it refi es on demand the descriptions of the target concepts, using only a subset of the examples found.
- These rules are learned only at the decision points that override the default behavior of the problem solver, instead of at all the decision points (i.e., as is done by other learning mechanisms). This is again a *lazy* learning approach for determining which rules should be learned.

The combination of these first two *lazy* aspects results in a system that can solve problems more efficiently, achieving better solutions than the heuristic-based problem solver.[18] Also, this lazy aspect allows HAMLET to learn useful control rules in nonlinear planning, which is a complex task to perform by *eager* approaches: it is difficult to describe the complete extra domain theory that *eager* approaches require in nonlinear planning (Minton 1988; Katukam and Kambhampati 1994). Furthermore, the results show that the third *lazy* aspect has several advantages over an *eager* learner since it achieves *better* solutions *more efficiently*, requires fewer learning resources (i.e., learning time and memory), and it suffers less from the utility problem.

Finally, HAMLET has been tested in a variety of experiments involving complex planning problems. The empirical results support the effectiveness of HAMLET's learning approach, in terms of improvement in planning efficiency, in the quality of plans generated, and in its incremental convergence towards the correct knowledge. In summary, HAMLET's learning power comes most directly from its overall lazy learning approach.

## Acknowledgements

This research for the first author was sponsored by Ministerio de Educación y Ciencia and Comunidad de Madrid. This research for the second author is 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 authors 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 authors would like to thank David Aha and the anonymous reviewers for their many useful comments.

## References

Aha, D. W., Kibler, D. & Albert, M. K. (1991). Instance-based learning algorithms. *Machine Learning* **6**(1): 37–66.

Bhatnagar, N. (1992). Learning by incomplete explanations of failures in recursive domains. In *Proceedings of the Ninth International Conference on Machine Learning*, pp. 30–36, Aberdeen, Scotland: Morgan Kaufmann.

[18] As stated in Section 2, the base problem solver already incorporates many useful domain-independent heuristics. It would be easier to outperform a system that performs completely uninformed search.

Borrajo, D., Caraç a-Valente, J. P. & Morant, J. L. (1992a). Learning heuristics in planning. In *Proceedings of the Sixth International Conference on Systems Research, Informatics and Cybernetics*, pp. 43–49, Baden-Baden, Germany: The International Institute for Advanced Studies in Systems Research and Cybernetics.

Borrajo, D., Caraç a-Valente, J. P. & Pazos, J. (1992b). A knowledge compilation model for learning heuristics. In *Proceedings of the First Workshop on Knowledge Compilation*, Aberdeen, Scotland.

Borrajo, D. & Veloso, M. (1994). Incremental learning of control knowledge for nonlinear problem solving. In *Proceedings of the European Conference on Machine Learning*, pp. 64–82. Catania, Italy: Springer Verlag.

Carbonell, J. G., Blythe, J., Etzioni, O., Gil, Y., Joseph, R., Kahn, D., Knoblock, C., Minton, S., Pérez, A., Reilly, S., Veloso, M. & Wang, X. (1992). PRODIGY4.0: The manual and tutorial. Technical Report CMU-CS-92-150, SCS, Carnegie Mellon University.

Carbonell, J. G., Knoblock, C. A. & Minton, S. (1990). Prodigy: An integrated architecture for planning and learning. In VanLehn, K. (ed.), *Architectures for Intelligence*, Erlbaum, Hillsdale, NJ. Also Technical Report CMU-CS-89-189.

Clark, P. & Holte, R. (1992). Lazy partial evaluation: An integration of explanation-based generalisation and partial evaluation. In *Proceedings of the Ninth International Conference on Machine Learning*, pp. 82–91, Aberdeen, Scotland: Morgan Kaufmann.

Cohen, W. W. (1990). Learning approximate control rules of high utility. In *Proceedings of the Seventh International Conference on Machine Learning*, pp. 268–276, Austin, TX: Morgan Kaufmann.

DeJong, G. F. & Mooney, R. (1986). Explanation-based learning: An alternative view. *Machine Learning* **1**(2): 145–176.

Doorenbos, R. B. & Veloso, M. M. (1993). Knowledge organization and the utility problem. In *Proceedings of the Third International Workshop on Knowledge Compilation and Speedup Learning*, pp. 28–34, Amherst, MA.

Estlin, T. A. & Mooney, R. (1995). Hybrid learning of search control for partial order planning. In *New Directions in AI Planning*, pp. 115–128. IOS Press.

Etzioni, O. (1993). Acquiring search-control knowledge via static analysis. *Artif cial Intelligence* **62**(2): 255–301.

Etzioni, O. & Minton, S. (1992). Why EBL produces overly-specif c knowledge: A critique of the Prodigy approaches. In *Proceedings of the Ninth International Conference on Machine Learning*, pp. 137–143. Aberdeen, Scotland. Morgan Kaufmann.

Fikes, R. E., Hart, P. E. & Nilsson, N. J. (1972). Learning and executing generalized robot plans. *Artif cial Intelligence* **3**: 251–288.

Hammond, K. J. (1989). *Case-based Planning: Viewing Planning as a Memory Task*. New York, NY: Academic Press.

Hanks, S. & Weld, D. (1995). A domain-independent algorithm for plan adaptation. *Journal of Artif cial Intelligence Research* **2**: 319–360.

Kambhampati, S. (1989). *Flexible Reuse and Modifica ion in Hierarchical Planning: A Validation Structure Based Approach*. PhD thesis, Computer Vision Laboratory, Center for Automation Research, College Park, MD: University of Maryland.

Kambhampati, S. & Kedar, S. (1991). Explanation based generalization of partially ordered plans. In *Proceedings of the Ninth National Conference on Artificia Intelligence*, pp. 679–685. Anaheim, CA: AAAI Press.

Katukam, S. & Kambhampati, S. (1994). Learning explanation-based search control rules for partial order planning. In *Proceedings of the Twelfth National Conference on Artificia Intelligence*, pp. 582–587. Seattle, WA: AAAI Press.

Kettler, B. P., Hendler, J. A., Andersen, A. W. & Evett, M. P. (1994). Massively parallel support for case-based planning. *IEEE Expert* **2**: 8–14.

Laird, J. E., Rosenbloom, P. S. & Newell, A. (1986). Chunking in SOAR: The anatomy of a general learning mechanism. *Machine Learning* **1**: 11–46.

Langley, P. (1983). Learning effective search heuristics. In *Proceedings of the Eighth International Joint Conference on Artificia Intelligence*, pp. 419–421, Los Altos, CA: Morgan Kaufmann.

Leckie, C. & Zukerman, I. (1991). Learning search control rules for planning: An inductive approach. In *Proceedings of the Eighth International Workshop on Machine Learning*, pp. 422–426, Evanston, IL: Morgan Kaufmann.

Minton, S. (1988). *Learning Effective Search Control Knowledge: An Explanation-Based Approach*. Boston, MA: Kluwer Academic Publishers.

Minton, S., Knoblock, C. A., Kuokka, D. R., Gil, Y., Joseph, R. L. & Carbonell, J. G. (1989). PRODIGY 2.0: The manual and tutorial. Technical Report CMU-CS-89-146, School of Computer Science, Carnegie Mellon University.

Mitchell, T. M., Keller, R. M. & Kedar-Cabelli, S. T. (1986). Explanation-based generalization: A unifying view. *Machine Learning* **1**: 47–80.

Mitchell, T. M., Utgoff, P. E. & Banerji, R. B. (1983). Learning by experimentation: Acquiring and ref ning problem-solving heuristics. In R. S. Michalski, J. G. Carbonell & T. Mitchell (eds.), *Machine Learning, An Artif cial Intelligence Approach*. Palo Alto, CA: Tioga Press.

Muggleton, S. (1992). *Inductive Logic Programming*. London: Academic Press Limited.

Pérez, M. A. & Carbonell, J. G. (1994). Control knowledge to improve plan quality. In *Proceedings of the Second International Conference on AI Planning Systems*, pp. 323–328, Chicago, IL: AAAI Press.

Pérez, M. A. & Etzioni, O. (1992). DYNAMIC: A new role for training problems in EBL. In *Proceedings of the Ninth International Conference on Machine Learning*, pp. 367–372, Aberdeen, Scotland: Morgan Kaufmann.

Porter, B. W., Bareiss, R. & Holte, R. (1990). Knowledge acquisition and heuristic classif cation in weak-theory domains. *Artif cial Intelligence* **45**: 229–263.

Quinlan, J. R. (1990). Learning logic def nitions from relations. *Machine Learning* **5**: 239–266.

Rich, E. (1983). *Artif cial Intelligence*. McGraw-Hill, Inc.

Ruby, D. & Kibler, D. (1992). Learning episodes for optimization. In *Proceedings of the Ninth International Conference on Machine Learning*, pp. 379–384, Aberdeen, Scotland: Morgan Kaufmann.

Stanf ll, C. & Waltz, D. (1986). Toward memory-based reasoning. *Communications of the Association for Computing Machinery* **29**: 1213–1228.

Stone, P., Veloso, M. & Blythe, J. (1994). The need for different domain-independent heuristics. In *Proceedings of the Second International Conference on AI Planning Systems*, pp. 164–169, Chicago, IL: AAAI Press.

Tadepalli, P. (1989). Lazy explanation-based learning: A solution to the intractable theory problem. In *Proceedings of the Eleventh International Joint Conference on Artif cial Intelligence*, pp. 694–700, San Mateo, CA: Morgan Kaufmann.

Valiant, L. (1984). A theory of the learnable. *Communications of the ACM* **27**(11): 1134–1142.

Veloso, M. & Blythe, J. (1994). Linkability: Examining causal link commitments in partial-order planning. In *Proceedings of the Second International Conference on AI Planning Systems*, pp. 170–175, Chicago, IL: AAAI Press.

Veloso, M. & Borrajo, D. (1994). Learning strategy knowledge incrementally. In *Proceedings of the Sixth IEEE International Conference on Tools with Artificia Intelligence*, pp. 484–490, New Orleans, LO: IEEE Computer Society Press.

Veloso, M., Carbonell, J., Pérez, A., Borrajo, D., Fink, E. & Blythe, J. (1995). Integrating planning and learning: The PRODIGY architecture. *Journal of Experimental and Theoretical AI* **7**: 81–120.

Veloso, M. M. (1989). Nonlinear problem solving using intelligent causal-commitment. Technical Report CMU-CS-89-210, School of Computer Science, Carnegie Mellon University.

Veloso, M. M. (1994a). Flexible strategy learning: Analogical replay of problem solving episodes. In *Proceedings of the Twelfth National Conference on Artificia Intelligence*, Seattle, WA: AAAI Press.

Veloso, M. M. (1994b). *Planning and Learning by Analogical Reasoning*. Springer Verlag.

Waldinger, R. (1981). Achieving several goals simultaneously. In Nilsson, N. J. & Webber, B. (eds.), *Readings in Artif cial Intelligence*, pp. 250–271. Los Altos, CA: Morgan Kaufmann.

Zelle, J. & Mooney, R. (1993). Combining FOIL and EBG to speed-up logic programs. In *Proceedings of the Thirteenth International Joint Conference on Artificia Intelligence*, pp. 1106–1113, Chambery, France: Morgan Kaufmann.