COMPOSER: A Probabilistic Solution to the Utility Problem in Speed-up Learning

Jonathan Gratch and Gerald DeJong
Beckman Institute for Advanced Studies,University of Illinois
405 N. Mathews, Urbana, IL 61801
e-mail: gratch.cs.uiuc.edu

Abstract

In machine learning there is considerable interest in techniques which improve planning ability. Initial investigations have identified a wide variety of techniques to address this issue. Progress has been hampered by the *utility problem*, a basic tradeoff between the benefit of learned knowledge and the cost to locate and apply relevant knowledge. In this paper we describe the COMPOSER system which embodies a probabilistic solution to the utility problem. We outline the statistical foundations of our approach and compare it against four other approaches which appear in the literature.

1 Introduction

Machine learning is often entertained as a mechanism for improving the efficiency of planning systems. Researchers have proposed a wide array of techniques to modify planning behavior, including macro-operators [DeJong86, Fikes72, Mitchell86], chunks [Laird86], and control rules [Minton88, Mitchell83]. With these techniques comes a growing battery of successful demonstrations in domains ranging from 8-puzzle to Space Shuttle payload processing. Unfortunately, in what is now called the utility problem, learned knowledge can hurt performance [Minton88]. This is underscored by a growing body of demonstrations where learning *degrades* planning performance [Etzioni90b, Gratch91a, Minton85, Subramanian90].

In an earlier paper we elaborated the limitations in a particular learning approach — [Gratch91b]. That paper sketched the COMPOSER system as one solution to these limitations. COMPOSER is intended as a general probabilistic solution to the utility problem. In this paper we detail our approach and report on a series of empirical evaluations. These tests compare COMPOSER's learning criterion against the approaches adopted by PRODIGY/EBL [Minton88], STATIC [Etzioni90b], a hybrid of PRODIGY/EBL and STATIC [Etzioni90a], and PALO [Greiner92]. The results substantiate our earlier analyses. They also cast doubt on the efficacy of nonrecursive control knowledge. This is significant as nonrecursive control knowledge has received considerable attention in the literature [Etzioni90b, Letovsky90, Subramanian90].

2 Learning as Search

Learning can be viewed as a transformational process in which the learning system applies a series of transformations to a performance element (see [Buchanan77, Gratch92]). The transformations available to a learner define its *vocabulary of transformations*. These are essentially learning operators and collectively they define a *transformation space*. For instance, acquiring a macro-operator can be viewed as transforming the initial system (the original planner) into a new system (the planner operating with the macro-operator). A learning system must explore this space for a sequence of transformations which result in a better planner.

To evaluate different learning approaches we must clarify our intuitive notions of when one planner is more efficient than another. We characterize planners through a numeric *utility function* which ranks the behavior of a planner over a fixed distribution of problems. We equate efficiency with minimizing planning time. Other measures are possible and our approach could apply to them as well. For any given problem, utility increases as the time to solve the problem decreases. The utility of a planner is defined as the sum of the utility of each problem in the distribution weighted by its probability of occurrence:

$$UTILITY(planner_i) = -\sum_{prob \in Distribution} Cost(planner_i, prob) \times Pr(prob)$$

Note that higher utility does not entail that the planning time of any particular problem is reduced. Rather, the expected cost to solve any representative sample of problems is less.

Utility is a preference function over planners. It is also useful to discuss the utility of individual transformations. The *incremental utility* of a transformation is defined as the change in utility that results from applying the transformation to a particular planner (e.g. adopting a control rule). This means the incremental utility of a transformation is conditional on the planner to which it is applied. We denote this as: ΔUTILITY(TransformationlPlanner).

3 COMPOSER

COMPOSER uses the previous definition of utility to evaluate and adopt control knowledge which, with high probability, improves planning performance. Its design was motivated by deficiencies in PRODIGY/EBL. Another paper illustrates how these deficiencies are shared by many other speed-up learning techniques [Gratch92]. In this paper we focus on the implementation of COMPOSER. In

[Gratch91b] we note that PRODIGY/EBL adopts two heuristic simplifications to identify beneficial control rules. First, aspects of the problem distribution are learned from a single example. Secondly, control rules are treated as if they do not interact. These simplifications have the unfortunate consequence that PRODIGY/EBL can learn control strategies which yield planners which are up to an order of magnitude *slower* than the original planner. We replace this heuristic approach with a rigorous alternative.

3.1 Algorithm

COMPOSER is implemented within the PRODIGY 2.0 architecture. This system includes the PRODIGY planner, a STRIPS–like system. The learning component of PRODIGY/EBL analyzes solution traces and proposes control rules to correct inefficiencies observed during planning. These control rules are condition–action statements which inform the PRODIGY planner to delete or prefer certain node, operator, or variable binding choices. COMPOSER primarily utilizes selection and rejection rules. This is discussed further in Section 5.

COMPOSER differs from PRODIGY/EBL in how statistics are gathered and how control rules are introduced into the PRODIGY planner. We implement a hill–climbing approach to the utility problem. The basic algorithm is sketched in Figure 1. We assume the user has provided a training set which is drawn randomly according to a fixed distribution of problems.

Input: TRAINING_EXAMPLES

CONTROL_STRATEGY = ∅
CANDIDATE_SET = ∅
While more training examples
  solve problem with Planner+CONTROL_STRATEGY
  learn new rules and add them to CANDIDATE_SET
  acquire statistics for all rules in CANDIDATE_SET from trace
  POSITIVE_RULES = ∅
  Forall rules ∈ CANDIDATE_SET
    If ΔUTILITY(rule|PRODIGY+CONTROL_STRATEGY)
      significantly negative
      Then remove rule from CANDIDATE_SET
    If ΔUTILITY(rule|PRODIGY+CONTROL_STRATEGY)
      significantly positive
      Then add rule to POSITIVE_RULES
  If POSITIVE_RULES
    add rule with highest utility to CONTROL_STRATEGY
    remove this rule from CANDIDATE_SET
    discard all statistics on rules in CANDIDATE_SET

Output: CONTROL_STRATEGY

Figure 1: The COMPOSER algorithm

Learning occurs with a single pass through the training examples. The algorithm incrementally adds control rules to a currently adopted control strategy. A rule is added only if it has demonstrated its benefit to a pre–specified confidence level. Once added, the rule changes how the planner behaves on subsequent training examples. New rules are proposed, and statistics gathered, with respect to the current control strategy. In this manner a control strategy is "grown" one rule at a time until the training set is exhausted.

3.2 Acquiring Utility Statistics

Gathering incremental utility statistics is the one aspect of COMPOSER which ties it to a particular representation for control knowledge — control rules. Other transformations would require analogous data gathering procedures.

A control rule should only be adopted if it improves the average efficiency of the problem solver. This average can be estimated by determining how the rule performs on individual problems and combining information from several problems. The next section discusses how to combine information. But first we will describe how COMPOSER extracts incremental utility values on individual problems.

How can we determine the incremental utility of a control rule on a particular problem? Ideally we have access to an efficient analytic model of the problem solver which can predict incremental utility. Unfortunately it is difficult to provide such a model for a non–trivial problem solver like PRODIGY. Instead we can measure utility empirically. The obvious approach is to solve the same problem multiple times — once for the current control strategy without the rule in question, and once using the strategy augmented with a candidate rule. The difference in problem solving cost between these runs is the incremental utility of the control rule on that problem. This problem must be repeatedly solved for each candidate rule. Clearly this approach is too expensive in practice.

COMPOSER uses a more efficient approach for gathering incremental utility values. It extracts a utility value for each candidate rule simultaneously from a single solution trace. While PRODIGY/EBL also derives multiple estimates from a single example, its technique is rendered inaccurate by the interactions which occur among rules (see [Gratch91b]). COMPOSER solves the interaction problem by extracting estimates without allowing the candidate rules to change the search behavior of the planner. Control rules only effect search behavior if they are *adopted* into the control strategy.

In contrast to adopted rules, the actions of *candidate* rules are not acted upon. They are simply noted in the problem solving trace. After a problem is solved, COMPOSER analyzes the annotated trace, and identifies the search paths which would have been avoided by each rule. The time spent exploring these avoidable paths indicates the savings which would be provided by the rule. This savings is compared with the recorded precondition match cost, and the difference is reported as the incremental utility of the rule for that problem.

It should be noted that this procedure is more expensive than the heuristic approach adopted by PRODIGY/EBL. This is because COMPOSER pays the penalty of matching preconditions without acquiring any of the benefit of candidate control rules. We are not aware of a reliable technique which avoids this additional cost.

3.3 Commitment Criterion

The incremental utility of a transformation across the problem distribution is estimated by averaging utility values from successive problems. The estimates should be accurate but based on as few examples as possible. In the field of statistics this is referred to as a *sequential analysis problem* (see [Govindarajulu81]). Observations are gathered until some stopping criterion is satisfied. As this criterion will commit COMPOSER to adopting or discarding a transformation we refer to this as a *commitment criterion.* In this case we are estimating the incremental utility of transformations to some specified confidence.

Formally, COMPOSER must choose among two hypotheses for each candidate:

H_0: ΔUTILITY(rule|planner+control_strategy) <= 0,

H_1: ΔUTILITY(rule|planner+control_strategy) > 0

In general there will be a discrepancy between the average of a sample and the true population mean. If the rule is negative, we wish to bound the probability that it will appear positive, and vice versa. It suffices to restrict the probability that the difference between the true utility and the estimate is larger than the magnitude of the true utility:[1]

$$Pr(\,|\text{ESTIMATE} - \Delta\text{UTILITY}| > |\Delta\text{UTILITY}|\,) = \delta$$

We use a distribution–free commitment criterion developed by Nádas [Nadas69]. The technique dynamically determines the number of examples sufficient to *approximately* achieve the specified confidence level. By approximate we mean that if the specified error level is δ the observed error rate may be slightly more or less than δ. The discrepancy is a function of the underlying distribution, but this type of approximation is very close in practice (see [Woodroofe82]). Examples must be taken until the following inequality holds:

$$(V_{r,n}/\overline{X}_{r,n})^2 < n(1/a)^2$$

where $\overline{X}_{r,n}$ is the average utility of the rule r over n problems, $X_{r,i}$ is the utility of r on the ith problem, $V_{r,n}{}^2 = n^{-1} + n^{-1}\sum(X_{r,i} - \overline{X}_{r,n})^2$ is the variance of the current sample, and n is greater than some small constant n_0 (we adopt a value of $n_0 = 3$ recommend by Adam Martinsek (personal communication) and evaluated in [Woodroofe82]). The parameter a satisfies the constraint that $\Phi(-a) = (1 - \delta)/2$, where Φ is the cumulative distribution function of the standard normal distribution.

The commitment criterion permits COMPOSER to identify when transformations are beneficial with some pre–specified probability. After each problem solving attempt, COMPOSER updates the statistics and evaluates the commitment criterion for each control rule in the candidate set. If no control rule has attained the confidence requirement, another problem is solved. If the commitment criterion identifies control rules with positive incremental utility (there may be more than one), COMPOSER adds the control rule with highest positive incremental utility to the current strategy, and removes it from the candidate set.[2] Statistics for the remaining candidates are discarded as they are conditional on the previous control strategy. If the commitment criterion identifies candidate rules with negative incremental utility, they are eliminated from the candidate set. Eliminating a candidate does not affect the current strategy, so the statistics associated with the remaining candidate control rules are not discarded. This cycle is repeated until the training set is exhausted. Each time a transformation is adopted the expected efficiency of the PRODIGY planner is increased, giving COMPOSER an *anytime* behavior [Dean88].

4 Evaluation

We evaluated COMPOSER's commitment criterion against several other commitment criteria. Before discussing the experiments we review these other criteria.

4.1 PRODIGY/EBL'S Utility Analysis

PRODIGY/EBL adopts transformations with a heuristic utility analysis. As control rules are proposed they are added to the current control strategy. The savings afforded by each rule is estimated from a single example and this value is credited to the rule each time it applies. Match cost is measured directly. If the cumulative cost exceeds the cumulative savings, the rule is removed from the current control strategy. The issue of interactions among transformations is not addressed.

4.2 STATIC's Nonrecursive Hypothesis

STATIC utilizes a commitment criterion based on Etzioni's structural theory of utility. The criterion is grounded in the *nonrecursive hypothesis* which states that transformations will have positive incremental utility, regardless of problem distribution, if they are generated from nonrecursive explanations of planning behavior (i.e. no predicate in a subgoal is derived using another instantiation of the same predicate). The issue of interactions between transformations is not addressed. STATIC applies this criterion to control rules but the issue is important in macro–operators as well [Letovsky90, Subramanian90].

STATIC has out performed PRODIGY/EBL's on several domains. The nonrecursive hypothesis is cited as the principle reason for this success [Etzioni90b]. This claim is difficult to evaluate as these systems can generate very different control rules. We clarify this issue by testing the nonrecursive hypothesis with the NONREC system, a re–implementation of STATIC's nonrecursive hypothesis within the PRODIGY/EBL framework. NONREC replaces PRODIGY/EBL's commitment criterion with a criterion which only adopts nonrecursive control rules.

1. This is a two–sided statistical test and thus is overly conservative. We are not aware of a good one–sided test for this problem.

2. If each candidate rule is estimated to a particular error level, the strategy of adopting the *first* positive rule to reach significance can result in a higher overall error level. The discrepancy is a function of how many candidate rules with negative utility are estimated to have positive utility. This discrepancy has not proven significant in our experience and there are ways to bound it at the cost of more training examples.

4.3 A Composite System

Etzioni has suggested that the strengths of STATIC and PRODIGY/EBL could be combined into a single system [Etzioni90a]. He proposed a hybrid system which embodies several advancements including a two layered utility criterion. The nonrecursive hypothesis acts as an initial filter, but the remaining nonrecursive control rules are subject to utility analysis and may be later discarded.

We implemented the NONREC–UA system to test this hybrid criterion. As control rules are proposed by PRODIGY/EBL's learning module, they are first filtered on the basis of the nonrecursive hypothesis. The remain rules undergo utility analysis as in PRODIGY/EBL.

4.4 PALO'S Chernoff Bounds

Greiner and Cohen have proposed an approach similar to COMPOSER's [Greiner92]. The Probably Approximately Locally Optimal (PALO) approach also adopts a hill–climbing technique and evaluates transformations by a statistical method. PALO differs in its commitment criteria and and that it incorporates a criterion for when to stop learning. PALO terminates learning when it has (with high probability) identified a near–local maximum in the transformation space. We will focus on the different commitment criteria which is based on Chernoff bounds.

The difference is that PALO provides stronger guarantees at the cost of more examples. This means that if the user specifies an error level of δ, the true error level will never exceed δ, and may in fact be much lower.[3] Our PALO–RI system evaluates this approach. Like COMPOSER, PALO–RI uses a candidate set of rules. In this case the size of the set is fixed before learning begins. A candidate is adopted when the following inequality holds:

$$\sum_{i=1}^{n} X_{r,i} > \Lambda_r \sqrt{2n \ln(Cs^2\pi^2/(3\delta))}$$

where $X_{r,i}$ is the incremental utility of rule r on problem i, C is the maximum size of the candidate set, s is one plus the number of rules in the current control strategy, and Λ_r is a is the maximal per problem ΔUTILITY of rule r. The parameter C is the size of the candidate set. We discuss the setting of the various parameters in the next section.

4.4 Experiments

We compare the STRIPS domain from [Minton88], the AB–WORLD domain from [Etzioni90a] for which PRODIGY/EBL produced harmful strategies, and the BIN–WORLD domain from [Gratch91a] which yielded detrimental results for both STATIC and PRODIGY/EBL's learning criteria. Results are summarized in Figure 2. In each domain the systems are trained on 100 training examples drawn randomly from a fixed distribution. The current control strategy is saved after every twenty training examples.[4] The graphs illustrate learning curves where the independent measure is the number of training examples and the dependent measure is execution time for 100 test problems drawn from the same distribution. This process is repeated eight times, using different but identically distributed training and test sets. Values in Figure 2 represent the average of these eight trials. "Rules Added" indicates the average number of rules learned by the system; "Train Time" is the number of seconds required to process the 100 training examples; "Test Time" is the number of seconds required to generate solutions for the 100 test problems.

COMPOSER and PALO–RI require confidence parameters which were set at 90%. PALO–RI's behavior is strongly influenced by parameters whose optimal values are difficult to assess. We tried to assign values close to optimal given the information available to us.[5]

As mentioned, COMPOSER does not implement a general approach to evaluating preference rules. In particular, it cannot properly evaluate the incremental utility of preference rules in the AB–WORLD and STRIPS domains. To ensure that differences reflect the commitment criteria and not the vocabulary of transformations, we disabled the learning of preference rules for every system in the STRIPS and AB–WORLD domains. We evaluated the ramifications of this change by comparing PRODIGY/EBL with and without preference rules and found that, in both domains, more efficient strategies resulted when preference rules were *disabled.* This is consistent with statements made by Minton concerning preference rules [Minton88 p. 129].

It was quickly apparent that PALO–RI would not adopt any transformations within the 100 training examples. We tried to give the system enough examples to reach quiescence but this proved too expensive. The problem is twofold — first, too many training examples were required; secondly, and as a consequence of the first problem, the candidate set grew large since harmful rules were not discarded as quickly as in COMPOSER. This increased the cost to solve each training example. To collect statistics on PALO–RI we only performed one instead of five learning trials. Furthermore, we terminated PALO–RI after the first transformation was adopted or 10,000 examples, whichever came first.

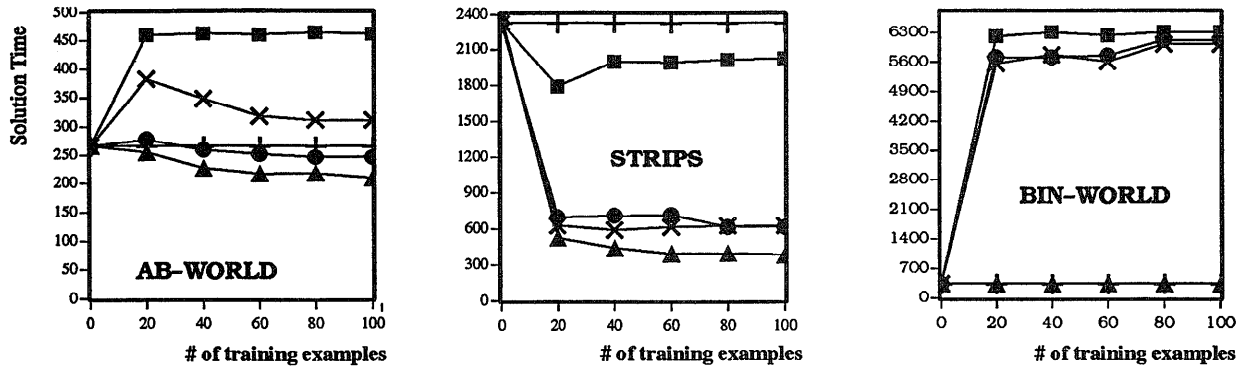
4.5 Discussion

The results illustrate several interesting features. COMPOSER exceeded the performance of all other approaches in every domain. In AB–WORLD and STRIPS, COMPOSER identified beneficial control strategies. In BIN–WORLD the system did not adopt any transformations. It does not ap-

3. PALO adopts three conservative refinements over COMPOSER: 1) chernoff bounds replace our approximate Nádas technique. 2) the worst case discrepancy from footnote 2 is bounded. 3) instead of bounding the error of adopting a bad rule at each step, PALO bounds the sum of all errors in the transformation sequence.

4. PRODIGY/EBL's utility analysis requires an additional settling phase after training. Each control strategy produced by PRODIGY/EBL and NONREC+UA received a settling phase of 20 problems following the methodology outlined in [Minton88].

5. C was fixed based on the size of the candidate list observed in practice. In the best case a rule can save the entire cost of solving a problem, so for each domain, lambda for each rule was set at the maximum problem solving cost observed in practice. AB–WORLD – C=30 lambda=15; STRIPS – C=20, lambda=100; BIN–WORLD – C=5, lambda=150.



SYSTEM	AB-WORLD Rules Added	AB-WORLD Train Time	AB-WORLD Test Time	STRIPS Rules Added	STRIPS Train Time	STRIPS Test Time	BIN-WORLD Rules Added	BIN-WORLD Train Time	BIN-WORLD Test Time
No Learning	—	—	266	—	—	2323	—	—	346
COMPOSER	1	1667	210	4	4133	382	0	3425	346
PRODIGY/EBL	11	1253	311	20	3775	622	2	6383	6020
NONREC	17	1253	462	25	4012	2021	4	6710	6305
NONREC+UA	9	1259	247	10	3821	614	2	6359	6110

Performance after first rule adopted	PALO-RI Train Exmpls	PALO-RI Train Time	PALO-RI Test Time
AB-WORLD	6069	104,387	182
STRIPS	1182	41,370	1223
BIN-WORLD	10,000+	—	346

Figure 2: Summary of empriical results

pear that any control rule improves performance in this domain. It should be stressed that all systems utilized the same learning module. Therefore the results represent differences in commitment strategies rather than in the vocabulary of transformations.

As expected, COMPOSER and PALO-RI had the highest learning times as they incur the precondition cost of candidate control rules without gaining the benefit of their recommendations. The one exception was BIN-WORLD where COMPOSER quickly discarded a very expensive control rule which PRODIGY/EBL, NONREC, and NONREC+UA retained. An encouraging result is that COMPOSER's learning times were not substantially higher than the non-statistical systems. PALO-RI's learning times were significantly higher.

The results cast doubts on the nonrecursive hypothesis. NONREC yielded the worst performance on all domains. Even in conjunction with utility analysis the results are mixed — benefit on the AB-WORLD, slightly worse than utility analysis alone in STRIPS, and worse than no-learning in BIN-WORLD. A post-hoc analysis of control strategies did indicate that the best rules were nonrecursive, but many nonrecursive rules were also detrimental. The slow-down on BIN-WORLD primarily results from one nonrecursive control rule. Thus it appears that nonrecursiveness may be an important property but is insufficient to ensure performance improvements. These results are interesting since Etzioni reports that STATIC outperforms PRODIGY/EBL and No Learning in AB-WORLD. The nonrecursive hypothesis cannot completely account for this difference. We attribute the difference to the fact that STATIC and NONREC entertain different sets of control rules. NONREC was constrained to use the vocabulary which was available to PRODIGY/EBL while STATIC has its own rule generator.

Finally, although PALO-RI did not improve performance within the 100 training examples, we believe that if it were given sufficient examples it would out perform all other systems. With extended examples it did exceed COMPOSER's performance in AB-WORLD. This is because the PALO approach commits to transformations with highest incremental utility while COMPOSER balances incremental utility against variance. Unfortunately the cost of PALO's performance improvement is very high, both in terms of examples and learning time. Thus, while COMPOSER may identify somewhat less beneficial strategies, it achieves much faster convergence.

5 Future Research

Our investigations have exposed two important issues for future research. First, there are difficulties in extending COMPOSER's utility gathering approach to preference rules. It is easy to record the match cost for these rules. The problem stems from determining how much a rule would save if it were added to the control strategy. This is straightforward in the case of rules which delete alternatives. The search space explored by the planner using such a rule will always be a subset of the search search space explored without the rule. This is not necessarily the case with preference rules. A candidate preference rule can suggest search paths which are not explored in the solution trace. Determining the savings of a preference rule under these circumstances is expensive. The learning system must re-invoke the planner and explore the alternative path. This need may arise many times in one problem.

This discussion points to a general issue that some transformation vocabularies may be easier to implement within the COMPOSER framework than others. Perhaps the issue can be resolved by identifying alternative means to gather utility values. This problem disappears if we are willing to solve training problems multiple times — with and without the candidate transformation — but this is unlikely to be feasible in practice.

A second issue is that our commitment criteria needs further investigation. The PALO approach, compared to COMPOSER's, provides stronger guarantees and can yield better control strategies but at a higher learning cost. Neither technique directly accesses the tradeoff between the improvement due to learning and the cost to achieve that improvement. Currently we are investigating ways to apply decision theoretic methods to resolve this tradeoff in a principled way.

6 Conclusions

Learning shows great promise to extend the generality and effectiveness of planning techniques. Unfortunately, many learning approaches are based on poorly understood heuristics. In many circumstances a technique designed to improve planning performance can have the opposite effect.

In this paper we discussed one general approach to the utility problem which gives probabilistic guarantees of improvement through learning. Our implementation is restricted to control rules but could be extended to other representations of control knowledge. We contrasted COMPOSER with four other learning techniques — three which do not provide guarantees, and one which does. The utility analysis method of PRODIGY/EBL, the nonrecursive hypothesis of STATIC, and even a combination of both can produce substantial performance degradations. Greiner and Cohen's PALO approach should yield somewhat better performance improvements than COMPOSER but at a substantially higher learning cost.

Acknowledgements

This research is supported by the National Science Foundation, grant NSFIRI 87–19766. We benefited from many discussions with Adam Martinsek and Russ Greiner. Thanks to Oren Etzioni and Nick Lewins for their comments.

References

[Buchanan77] B. G. Buchanan, T. M. Mitchell, R. G. Smith and C. R. Johnson, "Models of Learning Systems," in ***Encyclopedia of Computer Science and Technology, Vol. 11***, J. Belzer, A. G. Holzman, & A. Kent (ed.), Marcel Dekker, New York, NY, 1977, pp. 24–51.

[DeJong86] G. F. DeJong and R. J. Mooney, "Explanation–Based Learning: An Alternative View," ***Machine Learning 1***, 2 (April 1986), pp. 145–176.

[Dean88] T. Dean and M. Boddy, "An Analysis of Time–Dependent Planning," ***AAAI88***, Saint Paul, MN, August 1988

[Etzioni90a] O. Etzioni, "Why Prodigy/EBL Works," ***AAAI90***, Boston, MA, August 1990, pp. 916–922.

[Etzioni90b] O. Etzioni, "A Structural Theory of Search Control," Ph.D. Thesis, Department of Computer Science, Carnegie–Mellon University, Pittsburgh, PA, In preparation, 1990.

[Fikes72] R. E. Fikes, P. E. Hart and N. J. Nilsson, "Learning and Executing Generalized Robot Plans," ***Artificial Intelligence 3***, 4 (1972), pp. 251–288.

[Govindarajulu81] Z. Govindarajulu, ***The Sequential Statistical Analysis***, American Sciences Press, INC., Columbus, OH, 1981.

[Gratch91a] J. Gratch and G. DeJong, "A Hybrid Approach to Guaranteed Effective Control Strategies," ***ML91***, Evanston, IL, June 1991.

[Gratch91b] J. M. Gratch and G. F. DeJong, "On comparing operationality and utility," Technical Report UIUC–DCS–R–91–1713, Department of Computer Science, University of Illinois, Urbana, IL, 1991.

[Gratch92b] J. Gratch and G. DeJong, "A Framework of Simplifications in Learning to Plan," ***First International Conference on Artificial Intelligence Planning Systems***, College Park, MD, 1992.

[Greiner92] R. Greiner and W. W. Cohen, "Probabilistic Hill–Climbing," ***Proceedings of Computational Learning Theory and 'Natural' Learning Systems***, 1992.

[Laird86] J. E. Laird, P. S. Rosenbloom and A. Newell, ***Universal Subgoaling and Chunking: The Automatic Generation and Learning of Goal Hierarchies***, Kluwer Academic Publishers, Hingham, MA, 1986.

[Letovsky90] S. Letovsky, "Operationality Criteria for Recursive Predicates," ***AAAI90***, Boston, MA, August 1990

[Minton85] S. Minton, "Selectively Generalizing Plans for Problem–Solving," ***IJCAI85***, Los Angeles, August 1985, pp. 596–599.

[Minton88] S. N. Minton, "Learning Effective Search Control Knowledge: An Explanation–Based Approach," Ph.D. Thesis, Department of Computer Science, Carnegie–Mellon University, Pittsburgh, PA, March 1988.

[Mitchell83] T. M. Mitchell, P. E. Utgoff and R. Banerji, "Learning by Experimentation: Acquiring and Refining Problem–solving Heuristics," in ***Machine Learning: An Artificial Intelligence Approach***, R. S. Michalski, J. G. Carbonell, T. M. Mitchell (ed.), Tioga Publishing Company, Palo Alto, CA, 1983, pp. 163–190.

[Mitchell86] T. M. Mitchell, R. Keller and S. Kedar–Cabelli, "Explanation–Based Generalization: A Unifying View," ***Machine Learning 1***, 1 (January 1986), pp. 47–80.

[Nadas69] A. Nadas, "An extension of a theorem of Chow and Robbins on sequential confidence intervals for the mean," ***The Annals of Mathematical Statistics 40***, 2 (1969), pp. 667–671.

[Subramanian90] D. Subramanian and R. Feldman, "The Utility of EBL in Recursive Domain Theories," ***AAAI90***, Boston, MA, August 1990, pp. 942–949.

[Woodroofe82] M. Woodroofe, ***Nonlinear Renewal Theory in Sequential Analysis***, SOCIETY for INDUSTRIAL and APPLIED MATHEMATICS, Philadelphia, PA, 1982.