Utilization Filtering of Macros Based on Goal Similarity

Uri Keidar, Shaul Markovitch & Erez Webman
Computer Science Department
Technion, Haifa 32000
Israel
shaulm@cs.technion.ac.il

Abstract

Deductive learners acquire knowledge that is implicitly available to improve the performance of the problem solver. One of the most known form of deductive learning is the acquisition of macro operators. Macro-operators carry cost as well as benefits. When the costs outweigh the benefits, we face the utility problem. The vast number of macros available to the learner forces it to be selective to avoid the utility problem. The most common approach to selective macro-learning is using acquisition filters. Such filters try to estimate the utility of a macro before inserting it into the macro knowledge base. One problem with this approach is that the utility of a macro strongly depends on the problem being solved. In this work we suggest an alternative approach called *utilization filtering*. Instead of being selective when the macro is *acquired*, the learner is selective when the macro is *utilized*. We propose to use similarity-based filtering. A macro is considered as potentially useful for a particular problem if it proved to be useful for similar problems. Without further knowledge about the states in the search space, we suggest to use the heuristic function to determine similarity between states. Initial testing of this approach in the grid domain showed that indeed it is beneficial to delay selectivity to the utilization stage.

1 Introduction

One of the most known learning techniques for speeding up problem solvers is the acquisition of macro operators [6, 19]. While gaining experience in solving problems in a search space, the learning program memorizes sequences of transitions in the space. These sequences of transitions, called macro operators, are then used by the problem solver in future problem solving as if they were basic transitions.

Using macro operators carries obvious benefit: when using a sequence of transitions that eventually leads to the goal state, the problem solver saves the search from the start state to the end state of the sequence. However, macro operators carry also some costs due to the increase in the branching factor of the search graph. When the costs outweigh the benefits, the macros have negative *value* [10] and we face the so-called *utility problem*[10, 13, 18, 3].

The utility problem is more evident in macro-learning than in other types of learning. The main reason is the vast amount of macros that are available for acquisition. A uniform binary tree of depth n contains 2^{2n} different macros. To reduce the costs of macros the learner must be selective. Markovitch and Scott [10] analyze the learning information flow from the experience space through the acquisition procedure to the learned knowledge base and finally to the problem solver. They

identify five types of filters that a learner may employ: an experience filter, an attention filter, an acquisition filter, a retention filter and a utilization filter.

The most common filters employed in existing macro learning systems are acquisition filters and retention filters [4, 12, 13, 8, 2, 15, 16, 3]. The problem with acquisition and retention filters is that they must evaluate a macros on a global basis. However, the value of a macro strongly depends on the problem that the problem solver tries to solve. It is quite feasible that a macro has a negative utility when solving one problem and a positive utility when solving another. Such a macro may have a global zero utility. The filter that is most competent in dealing with such a problem is the *utilization filter*. Ideally, a utilization filter passes to the problem solver only macros that have positive utility for the current problem. In practice, the only way of *knowing* the utility of a macro for a particular problem is by solving it. Therefore, a utilization filter will typically *estimate* the likelihood of a macro to be harmful for a particular problem.

Markovitch and Scott [9, 10] describe a methodology for utilization filtering in the context of lemma learning. The filter is binary - it turns off macro utilization below nodes in the derivation tree that are likely to fail. The motivation behind this filter is the phenomenon where in backtracking search every lemma usage under a failure node is bound to be harmful.

In this work we propose a method for utilization filtering of macros in general search spaces which depends only on the heuristic function used by the problem solver. The method relies on a general assumption that a macro useful for solving one goal is likely to be useful also in solving "similar" goals. The problem is to define similarity in this context. Mooney [14] takes this approach to the extreme and only allows utilization of macros that actually solve the problem.

When features of the states in the search space are available, one could define similarity based on those features [2]. Without such features, we propose to use the heuristic metric used by the problem solver for defining similarity between states. Goals are similar if they reside in the same neighborhood, where distance is measured using the heuristic function. A record of goals for which the macro was useful is kept for each macro. When the problem solver expand function asks for the macro operators, the utilization filter passes only macros that were successful in solving similar goals.

In Section 2 we give some background on the problem of selective macro learning. Section 3 describes the methodology used for utilization filtering. Section 4 describes a set of experiments that test the methodology in the grid domain. Finally, Section 5 concludes.

2 Background

Let $G = \langle S, \varphi \rangle$ be a search space where S is a set of states and $\varphi : S \rightarrow 2^S$ is the successor function. A problem in G is a pair $\langle s_i, s_g \rangle$ where $s_i, s_g \in S$. A solution to a problem $\langle s_i, s_g \rangle$ is a sequence $\langle s_1, s_2, \ldots, s_k \rangle$ where $s_1 = s_i, s_k = s_g$ and $s_j \in \varphi(s_{j-1})$ for $1 < j \leq k$. A macro in G is a sequence of states $\langle s_1, \ldots, s_m \rangle$ where $s_j \in \varphi(s_{j-1})$ for $1 < j \leq m$. Let $G = \langle S, \varphi \rangle$ be a search space and M a set of macros defined over G. The extended search space is $G_M = \langle S, \varphi_M \rangle$ where $\varphi_M(s) = \varphi(s) \cup \{r \in S | \langle s, \ldots, r \rangle \in M$. A heuristic over G is a function $h : S \times S \rightarrow N$ where $h(s_1, s_2) = 0 \leftrightarrow s_1 = s_2$.

A *search* is a sequence of applications of the successor function starting from the initial state. Formally:

1. $\langle s_i \rangle$ is a search

2. Let $B = \langle s_1, \ldots, s_k \rangle$ be a search. Then, for each $s \in B$, $B \| \varphi(s)$ is also a search[1].

$Cost(B)$, the cost of the search B, is defined to be $|B|$ (the number of states generated).

A search strategy is a pair of functions $\tau = \langle \mu, \psi \rangle$ where μ takes a search and picks a state to expand and ψ is a predicate that decides when the search should be stopped. Given a space G and a search strategy τ, the search for a problem p is marked as $B_{\tau,G}(p)$. A *best-first* search strategy expands the unexpanded state with he lowest heuristic value and stops whenever the goal state is generated.

Given a search space S and a set of a macro M, the *utility* of a macro m (not necessarily in M) with respect to a problem p is defined as

$$U_{G,M,\tau,p}(m) = Cost(B_{\tau,G_M}(p)) - Cost(B_{\tau,G_{M-\{m\}}}(p)) \quad (1)$$

The utility of a macro subset is defined similarly. The utility of a macro with respect to a problem distribution D is the expected value of U for a problem randomly selected from D.

According to the information filtering framework [11], there are several filters that can be applied to select macros with high utility. A *selective acquisition* filter decides whether to add a macro to the existing macro set. A *selective retention* filter decides what subset of macros should be removed out of the existing set. A *selective utilization* filter decides what macros should be applied in the context of a given problem. The goal of all these filters is to increase the utility of the learned macro set.

In the following section we describe a general methodology for utilization filtering in search spaces with heuristic functions.

3 Utilization filtering based on goal-similarity

Assume that the search program tries to solve problem p using a strategy τ and a set of macros M. The role of the utilization filter is to show the problem solver only a part of M. Unlike retention filters, the utilization filter can utilize its knowledge of the particular problem being solved. Finding the best subset of macros is computationally hard since there is an exponential number of possible subsets. Utilization filters must be efficient since they cost in problem-solving time (versus retention filters which cost in learning time). Therefore it is quite reasonable to consider macros independently and to ignore possible dependencies.

In this work we propose a general strategy for utilization filtering: When trying to solve a problem p, use only macros that were useful for problems *similar* to p. A macro m is *useful* with respect to a problem p if there is a solution path that includes m. This information is collected during learning.

To determine the similarity of two problems one can look at the similarity of the goal states. Let M be a set of macros. For each $m \in M$ let $U(m)$ be the set of problems for which m proved to be useful (during the learning phase). Let $p = \langle s_i, s_g \rangle$ be the currently solved problem. Let $SIM : S \times S \rightarrow \{True, False\}$ a similarity predicate over the set of states. A similarity-based filter with respect to M, p and SIM is a function $f : M \times P \rightarrow \{True, False\}$ where

$$f(m, \langle s_i, s_g \rangle) = true \Leftrightarrow \exists s \in U(m)[SIM(s, s_g]$$

[1] We assume some arbitrary order on $\varphi(s)$

If the domain definition includes a set of features over states, the similarity can be defined in terms of these features. Finkelshtein and Markovitch [2] devised a utilization filtering method for search spaces where there is a set of features over states. However, they use these features to restrict the application of a generalized macro to states that are similar to the original state where the macro was acquired.

What similarity measure can be used in the general case where the only available information is the search space and a heuristic function? We propose to use in such a case the heuristic function itself to determine the similarity of states. Define the heuristic-based *irrelevance* r of a macro m to a problem $p = \langle s_i, s_g \rangle$ to be the minimal distance of s_g to goal states in $U(m)$:

$$r(m, \langle s_i, s_g \rangle) = \min_{q \in U(m)} h(q, s_g)$$

We define two alternative utilization filters that are based on the heuristic function used for the search:

$$f^k_{\text{k-thresh}}(m, p) = true \Leftrightarrow r(m, p) \leq k \quad (2)$$

$$f^k_{\text{k-best}}(m, p) = true \Leftrightarrow |\{j \in M | r(j, p) \leq r(m, p)\}| \leq k \quad (3)$$

The first filter selects macros whose relevance is smaller then a certain threshold k. The second filter selects the k most relevant macros.

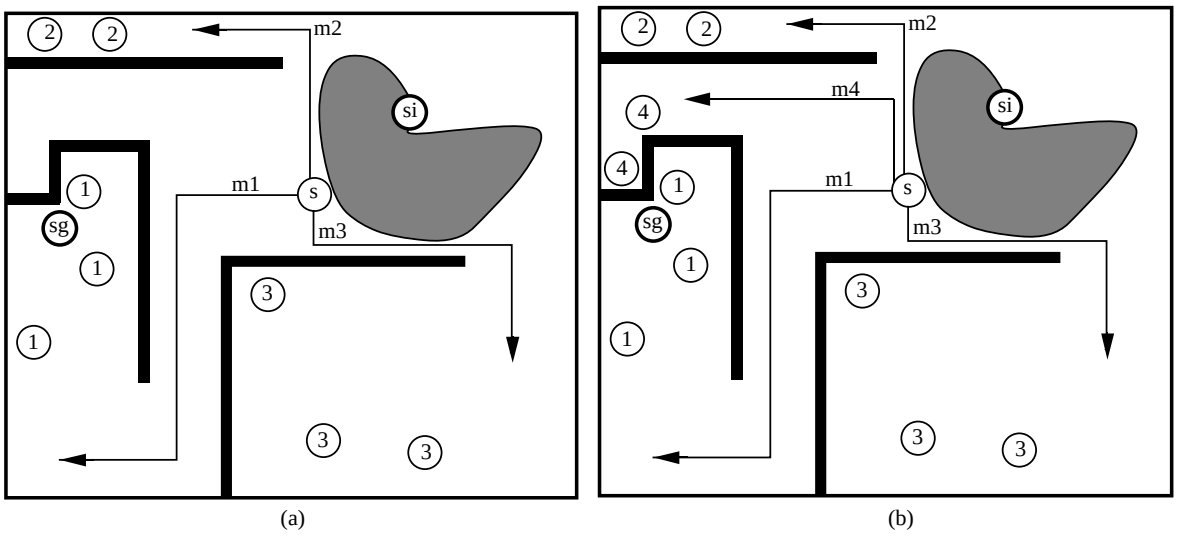

Figure 1: An example of heuristic-based utilization filtering. At state s in (a) there are 3 available macros: m_1, m_2 and m_3. While m_2 takes the searcher closer to the goal S_g, the utilization filter prefers macro m_1. The experience gained with m_1 showed that it was helpful in solving goals closer to s_g than the goals solved by m_2. Figure (b) demonstrates a pathology where the filter can make errors. m_4 looks as a promising macro while it is actually not.

The intuition behind these filters is clear. A macro that was helpful in bringing the search to goals that are close to the current goal is likely to be also helpful for the current goal. Figure 1(a) shows an example of a applying the heuristic based filters in a grid domain with the Manhattan-distance heuristic. The search procedure searches from state s_i and its goal is s_g. The gray area

is the currently searched area. At state s there are three macros available: m_1, m_2 and m_3. The states in $U(m_i)$ are marked with i. The filters will select m_1 since the states in $U(m_1)$ are close to s_g.

Note that macro m_2 is misleading. Its end point is heuristically closer to the goal than the end point of macro m_1. However, the experience gained with using both macros indicates that m_1 is more relevant to the current goal than m_2. Obviously, such rough evaluation is bound to fail in the presence of pathologies. For example, macro m_4 in Figure 1(b) also looks useful while it is actually not.

4 Experimental study

We have performed a set of experiments to test the effectiveness of the similarity-based utilization filter.

4.1 The grid domain

The experiments reported here were performed in the grid domain. There are two reasons for using this domain. In this domain every state has two coordinates. Every state has transitions to four neighbors, except those transitions that are blocked. The Manhattan-distance heuristic would have been a perfect heuristic in such a domain, but various "walls" within the grid causes it to make errors.

First, the grid domain allow us to generate many artificial domains easily, varying various properties of the domains. Second, navigation is such domains is an important problem for robotics and VLSI design.

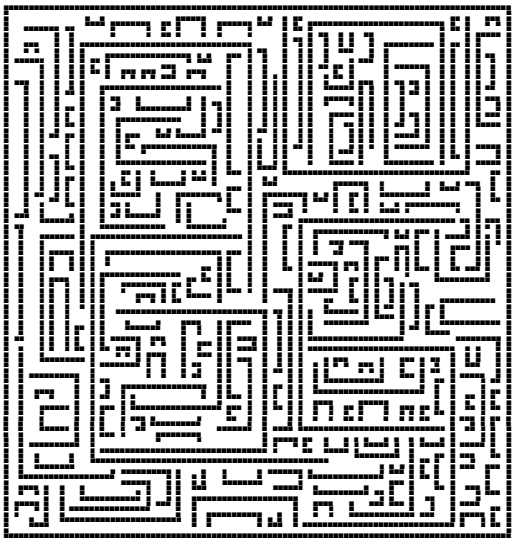

Figure 2: Example

We have devised an algorithm for generating random grids based on supplied parameters. All the grids generated by the algorithm are fully connected (i.e., there is a path between every pair of states) to avoid the problem of incorporating measurements of searches that fail [17, 1]. All the results reported in this paper were obtain when using the grid shown in figure 2. The grid size is

100×100. Thus, the number of states is 10000. We are currently in the process of repeating the experiments on other grids.

4.2 Acquisition filters

In this work we concentrate on utilization filtering. However, we must employ some acquisition filter during the search, otherwise an extremely large number of macros will be acquired. A very common restriction is to only acquire macros that are subpaths of the solution path. Even this restriction is not enough since a solution path of length n yields $O(n^2)$ macros. Iba [5] proposed the *peak-to-peak* strategy for acquisition filtering. This filter only allows macros that connect states whose heuristic values are peaks in the sequence of heuristic values from the initial to the goal states. Finkelshtein and Markovitch [2] propose the *minimum-to-better* acquisition strategy. The start state of a macro learned by this strategy is a local minimum. The end state is the first state that has a heuristic value better than the local minimum. The idea behind the *minimum-to-better* strategy is to acquire macros that are able to rescue the search program from local minima.

The *minimum-to-better* strategy high selectivity yields small and high quality macro databases. To test the utilization filter with larger macro databases, we also used the *dispersion* acquisition strategy. This strategy picks k random macros from each solution path. The idea behind this filter is to acquire macros that are well spread over the space.

4.3 Experimental methodology

A learning session was conducted by feeding the learner with 2000 randomly generated training problems. At various stages of the learning, we turned learning off and tested the learner using 500 randomly generated testing problems.

We have used several dependent variables for measuring the performance of the filters, but the main measurement used is the total number of nodes generated while solving the testing set.

There are many independent variables in such an experiment but due to the limited scope of this research we fixed most of them. The main independent variables are:

Search space We have used the grid shown in Figure 2. We tried other grids as well and got very similar results.

Search strategy Best-first search. Optimizing search such as A* is not suitable for macro-usage [7].

Heuristic function Manhattan-distance.

Acquisition filter We have tried the two filters.

Utilization filter We have tried the two filters and with various settings.

4.4 Results

Four main experiments were performed. Learning using *minimum-to-better* strategy with and without utilization filter and learning using the *dispersion* strategy with and without utilization filtering.

Using the *minimum-to-better* strategy without a utilization filter proved to be successful. The total number of generated nodes decreased from 317395 to 58268. After about 1000 training problems the graph becomes flat and there is no more improvement in performance. Using the utilization filter improved the performance only by 4%. Why didn't utilization filtering improve the performance? Since the *minimum-to-better* strategy is very selective, it learned relatively few macros. The number of macros learned was close to 2000, i.e., about one macro per problem on average. Since there are 10000 states, the average branching factor increased by 5%. Therefore, even the filter that selects only one best macro per state ($f^1_{\text{k-best}}$) did not affect the search.

The *dispersion* strategy includes a parameter that specifies how many macros can be learned for each training problem. A learning session with a limit of 100 macros per problem was conducted. The total number of macros learned was 20,000. Thus, the average branching factor increased by 50%. Indeed, when tested without a utilization filter, performance improved for the first 300 training problems, but started to deteriorate after that. At the end of the learning session the testing set was solved generating 81908 nodes.

We then repeated the tests with the utilization filter $f^1_{\text{k-best}}$ (The one that selects the macro that is the most *relevant* to the problem being solved.) The performance improved significantly. Instead of deterioration in performance, there was a steady progress. At the end of the learning session the testing set was solved using 23305 generated nodes. This result is 3.5 times faster than without the filter, and 2.5 times faster than with the *minimum-to-better* strategy (with or without filter).

Figure 3 shows the learning curves of the three experiments. We have omitted the curve for the experiment with *minimum-to-better* and utilization filter since it is almost identical to the curve without the filter.

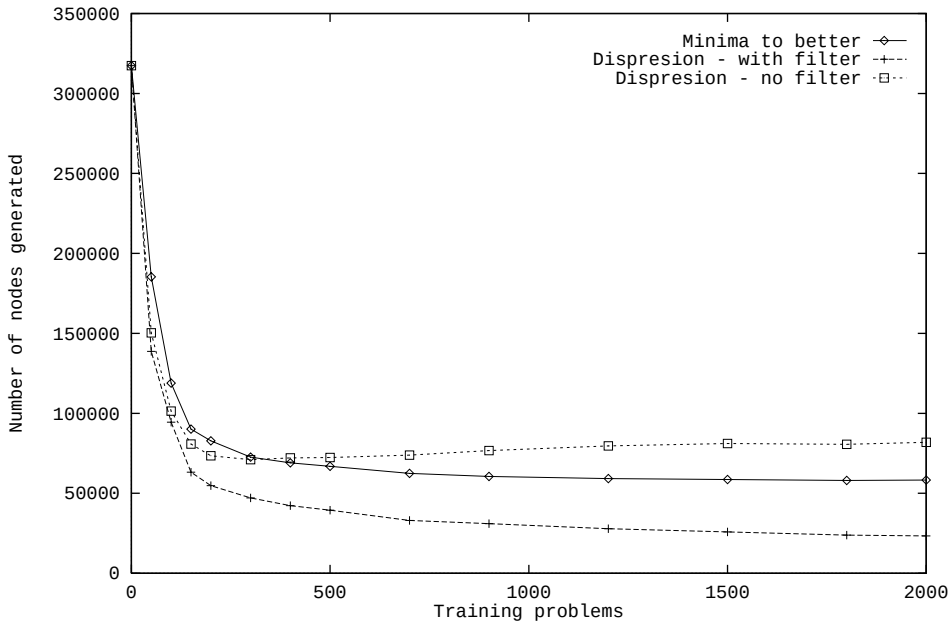

Figure 3: Learning curves of the three learning session. The best result was achieved when using a liberal acquisition filter and a restrictive utilization filter

5 Conclusions

Deductive learners, such as macro learning programs, use the inference rules of the deductive system to acquire sentences that are only implicitly available in the transitive closure of the basic axioms. Since the process of generating members of the transitive closure is so simple, most deductive learners suffer from information explosion. Thus, the key for successful deductive learning is an efficient filtering of the learned knowledge.

The most common filtering mechanism used is *acquisition filter*. Such a filter tries to estimate the utility of a learned macro before it is inserted into the knowledge base. Markovitch and Scott [10] pointed out that the utility of a macro (or another form of deductive rule) strongly depends on the problem being solved, and describe a method for *utilization filtering* of learned knowledge in backtracking search.

In this work we offer another method of *utilization filtering* that is most useful for a heuristic best-first search in search graphs where there is no information about the individual states. The assumption behind our method is that a macro is potentially useful for one problem when it proved to be useful for "similar" problem. Since we assume that no knowledge is available about the individual states, we use the heuristic distance to determine similarity between states.

We conducted initial experiments in the grid domain. The results are encouraging. The most interesting results suggests that, unless storage is a major concern, it is better to be less selective when *acquiring* macros and more selective when *using* them than the other way around.

One problem that we have not discussed here is the cost of utilization filtering. Since filtering is performed for each expansion of a node, it should carry a very low execution cost otherwise its benefit will be outweigh buy this cost. If we assume a non-generalized macros such as described here, we can index macros by states. Then, for each expansion the filter will only have to test the macros associated with the current state. Computing the distance between the goal state and the members of $U(m)$ is cheap. However, when the size of $U(m)$ increases, the cost of computing the similarity may become prohibitive. To avoid high costs we can limit the size of this set, or measure the distance to a centroid of the set.

There is still much work to be done to study the limitation and the generality of this method. We are now in the process of testing the effect of various parameters in the grid domain. We also intend to test the effectiveness of this approach to other known domains. In addition, we are investigating other variants of the method described.

References

[1] O. Etzioni and R. Etzioni. Statistical methods for analyzing speedup learning experiments. *Machine Learning*, 14:333–347, 1994.

[2] L. Finkelshtein and S. Markovitch. Selective acquisition and utilization of macro operators: A learning program solves the general n x n puzzle. Technical report CIS l9216, Computer Science Department, Technion, Haifa, Israel, 1992.

[3] J. Gratch and D. DeJong. COMPOSER: A probabilsitic solution to the utility problem in speed-up learning. In *Tenth National Conference on Artificial Intelligence*, pages 235–240, San Jose, California, 1992. American Association for Artificial Intelligence.

[4] G. A. Iba. A heuristic approach to the discovery of macro-operators. *Machine Learning*, 3(4):285–317, 1989.

[5] G. A. Iba. A heuristic approach to the discovery of macro-operators. *Machine Learning*, 3(4):285–317, Mar. 1989.

[6] R. E. Korf. Macro operators: A weak method for learning. *Artificial Intelligence*, 26:35–77, 1985.

[7] S. Markovitch and I. Rosdeutscher. Systematic experimentation with deductive learning: Satisficing vs. optimizing search. In *Proceedings of Knowledge Compilation and Speedup Learning workshop*, Aberdeen, Scotland, 1992.

[8] S. Markovitch and P. D. Scott. The role of forgetting in learning. In *Proceedings of The Fifth International Conference on Machine Learning*, pages 459–465, Ann Arbor, MI, 1988. Morgan Kaufmann.

[9] S. Markovitch and P. D. Scott. Utilization filtering: a method for reducing the inherent harmfulness of deductively learned knowledge. In *Proceedings of The Eleventh International Joint Conference for Artificial Intelligence*, pages 738–743, Detroit, Michigan, 1989.

[10] S. Markovitch and P. D. Scott. Information filtering: Selection mechanisms in learning systems. *Machine Learning*, 10(2):113–151, Feb. 1993.

[11] S. Markovitch and P. D. Scott. Information filtering: Selection mechanisms in learning systems. *Machine Learning*, 10:113–151, 1993.

[12] S. Minton. Selectively generalizing plans for problem-solving. In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, 1985. Morgan Kaufmann.

[13] S. Minton. *Learning Search Control Knowledge: An Explanation-Based Approach*. Kluwer, Boston, MA, 1988.

[14] R. Mooney. The effect of rule use on the utility of explanation-based learning. In *Proceedings of The Eleventh International Joint Conference for Artificial Intelligence*, pages 725–730, Detroit, Michigan, 1989.

[15] D. Ruby and D. Kibler. Learning subgoal sequences for planning. In *Proceedings of The Eleventh International Joint Conference for Artificial Intelligence*, pages 609–614, Detroit, Michigan, 1989.

[16] D. Ruby and D. Kibler. Ease: Integrating search with learning episodes. Technical Report 92-30, Information and Computer Science, University of California, Irvine, California, 1992.

[17] A. Segre, C. Elkan, and A. Russell. A critical look at experimental evaluations of ebl. *Machine Learning*, 6:183–195, 1991.

[18] D. Subramanian and S. Hunter. Measuring utility and the design of provably good ebl algorithms. In *Proceedings of The ninth International Workshop on Machine Learning*, Aberdeen, Scotland, 1992. Morgan Kaufmann.

[19] P. Tadepalli. A formalization of explanation-based macro-operator learning. In *Proceedings of International Joint Conference for Artificial Intelligence*, pages 616–622, Sydney, Australia, 1991.