Simple Selection of Utile Control Rules in Speedup Learning

Lawrence B. Holder and Anurag Chaudhry
Department of Computer Science Engineering
University of Texas at Arlington
Box 19015, Arlington, TX 76019-0015
Email: holder@cse.uta.edu, chaudhry@cse.uta.edu

Abstract

Many recent approaches to avoiding the utility problem in speedup learning rely on sophisticated utility measures and significant numbers of training data to accurately estimate the utility of control knowledge. Empirical results presented here and elsewhere indicate that a simple selection strategy of retaining all control rules derived from a training problem explanation quickly defines an efficient set of control knowledge from few training problems. This simple selection strategy provides a low-cost alternative to example-intensive approaches for improving the speed of a problem solver.

1 INTRODUCTION

Initial attempts to solve the utility problem in speedup learning using empirical evaluations of control rules met with limited success due to a limited understanding of the problem solver's behavior [Etzioni and Minton, 1992]. More recent work applies statistical measures to learn control rules for which there is a high certainty of utility [Gratch and DeJong, 1992; Greiner and Jurisica, 1992]. However, these approaches require a large number of training problems to estimate the distribution and ensure utile control rules. Preliminary results in [Holder, 1992a; Holder, 1992b] and more recent results reported here suggest that a simple intermediate approach may yield sufficient speedup with fewer training problems and without specific utility measures.

In this work, control rules take the form of preference rules combined with a counter of the number of times the control rule has been extracted from solutions to previously-encountered training problems. The simple control-rule selection strategy investigated here is to retain every control rule, maintaining counters for repeated control rules, and selecting the control rule, having the highest count, that is applicable to the current goal.

After generating learning curves (match cost on a set of testing problems as a function of training problems) in several domains using the simple control rule selection strategy, we find that the learning curves have a single global minimum (lowest cost) that occurs after only a few training problems (< 5 for the domains studied). This suggests a greedy strategy may be effective for selecting control rules which learns every control rule until performance degrades on a separate set of testing problems. Of course, this strategy transfers to the testing set the need for large numbers of problems to estimate the problem distribution. However, the main advantage of this approach is the empirical identification of a typically small number of training problems needed to inject utile control rules into the problem solver. After empirically identifying the minimum of the learning-cost curve, the next step is to develop a theory for predicting the number of training examples corresponding to the minimum based on characteristics of the domain.

The next section discusses the general utility problem of degrading performance due to increasing amounts of learned knowledge. This problem is not isolated to speedup learning, but affects other learning paradigms as well. Section 3 describes other approaches to solving the utility problem in speedup learning. Section 4 illustrates the trend in the learning-cost curve for several domains, and Section 5 discusses the implications of these results. Section 6 presents our conclusions.

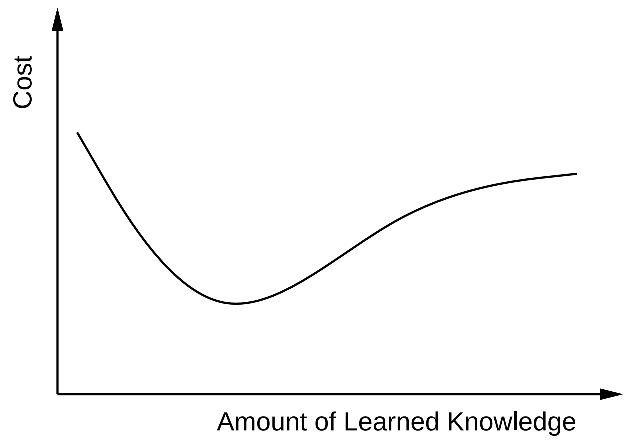

Figure 1: The relationship between cost and knowledge exhibited by a learning method suffering from the general utility problem.

2 GENERAL UTILITY PROBLEM

Both inductive and speedup learning methods suffer from the general utility problem: the eventual degradation of performance due to increasing amounts of low utility learned knowledge. In inductive methods the performance degradation is due to overfit. Overfit occurs when the learning method identifies errant patterns in the training data. Inductive learning methods typically use a set of training examples to generate knowledge for reducing classification error on unseen examples. Errant patterns may arise due to noise in the training data or inadequate stopping criteria of the method. As inductive methods generate more knowledge, they may increase the complexity of the learned hypothesis. For example, some inductive learning methods adapt a parameterized model to training data. As the number of parameters in the model becomes a sizable fraction of the data, the method fits the parameters according to trends in the training data that may not occur in unseen examples. Thus, a typical cost (classification error) curve as a function of the amount of learned knowledge will be similar to that in Figure 1.

The utility problem has been verified in several speedup learning systems [Minton, 1988b; Tambe and Rosenbloom, 1989; Mooney, 1989; Markovitch and Scott, 1989]; however, the underlying cause is less obvious than in inductive learning. From a high-level perspective, the utility problem is also caused by overfit. The learned knowledge applies to the solution of the training examples, but not necessarily to unseen examples. At a more detailed level, several factors contribute to the performance of a speedup learning system. First, the speedup learning system must search the learned knowledge for pieces applicable to solving the problem. As the amount of learned knowledge increases, so does this cost. Second, application of this knowledge may either increase cost by searching futile paths in the search space or decrease cost by preferring more applicable paths in the search space. The exact interaction of these factors is difficult to predict. However, the overall trend also follows the behavior in Figure 1, where cost is typically problem-solving time.

The *general utility problem* in machine learning refers to the degradation of performance due to increasing amounts of learned knowledge [Holder, 1990]. This term derives from the *utility problem* used by Minton [1988a] to describe this phenomenon in speedup learning, but generalizes to other machine learning paradigms. Other researchers have observed the ubiquity of the utility problem in machine learning paradigms and compare the utility problem in speedup learning to the problems of noise and overfit in inductive learning [Yoo and Fisher, 1991]. This work suggests that individual methods for avoiding the general utility problem may derive from a general model of the relationship between learned knowledge and performance that applies to several learning paradigms. Empirical instances of this model also indicate that minimizing the utility problem in speedup learning may involve fewer training examples than predicted by other approaches to avoiding the utility problem.

3 RELATED WORK

Most approaches to avoiding the utility problem rely on training examples to empirically evaluate the utility of learned knowledge. The Prodigy system [Minton, 1988a] evaluates the utility of problem-solving control knowledge by estimating the application cost, frequency and savings afforded by the control knowledge based on the training problems.

Eskey and Zweben [1990] describe a plausible approach to speedup learning when several examples are needed to support an instance of the target concept. They employ a decision function that evaluates control knowledge based on the explanation's probability of correctness, expected cost to match search control rule, and expected degradation in solution quality. Several examples are needed to support an explanation with high confidence and adopt the corresponding control rules.

The Composer system [Gratch and DeJong, 1992] embodies a probabilistic solution to the utility problem. Composer defines the utility of a planner as the sum of the utility of each problem in the distribution weighted by its probability of occurrence. A candidate control rule is evaluated in the context of the existing planner. If there is high confidence that a rule will benefit the planner, then the rule is added to the planner control knowledge. Other candidate rules are then re-evaluated in the context of the new planner. Composer incrementally adds control rules to its control strategy. The utility of the rule depends on the current control strategy. A rule is added only after demonstrating benefit to a pre-specified confidence level. Higher confidence levels require larger numbers of examples.

The PALO (Probably Approximately Locally Optimal) [Greiner and Jurisica, 1992] approach adopts a hill-climbing technique that evaluates transformations to the performance element (as effected by control knowledge) using a statistical method. Learning terminates when PALO has identified (with high probability) a near-local maximum in the transformation space. PALO provides stronger guarantees than Composer (and Prodigy/EBL) at the cost of more examples. Harmful rules are not discarded in PALO as quickly as they are in Composer. This results in a larger candidate (control rule) set in PALO which increases the cost to solve each training example. On the other hand, while Composer uses utility analysis when climbing to performance elements with superior performance, the analysis does not guarantee to produce optimal performance elements.

The above systems depend on the training examples for the distribution of problems in the domain. Typically, a large number of training examples are necessary to accurately estimate the problem distribution and the utility of control knowledge. Moreover, the task of finding the optimal set of control knowledge, even knowing the distribution, is intractable most of the time [Greiner and Jurisica, 1992]. On the other end of the spectrum, simply limiting the *amount* of the learned knowledge (while ignoring utility) may be advantageous in terms of learning time saved. PALO tries to estimate the unknown distribution, but the learning time is extremely high. Hence, concentrating on the amount of learned knowledge rather than the utility of the learned knowledge might be feasible. Excessive knowledge degrades performance. Limiting learned knowledge, without utility evaluation, may save learning time and eliminate degradation.

4 EXPERIMENTATION

This section presents empirical results on learning control knowledge in several domains. The results indicate that an efficient set of control rules (relative to the cost of the original domain theory) can be learned using a simple selection strategy and a small number of training examples.

4.1 Methodology

The experiments use a backward-chaining Prolog-like deductive retriever with proof tree and control rule generation capabilities. A depth bound of 20 was used during proof tree generation. Explanation-based generalization [Mitchell *et al.*, 1986] is used to generalize the proofs. The generalized proof is used to generate the search control knowledge (control rules) for guiding the retrieval process.

The number of matches (unifications) serves as a performance criteria for monitoring performance changes with increase in the amount of learned knowledge, i.e., control rules. The database rules are Horn clauses, and control rules are of the form:

if goal
then database rule

The control rules are preference rules which choose a particular rule for solving the current goal. If the antecedent of the control rule matches (unifies with) the current goal then the consequent of the control rule is preferred over other rules to solve the current goal. The antecedent of the control rule points to a rule in the database. A weight is associated with each control rule. The weight represents the total number of times the control rule has been successfully used for solving problems (i.e., proving goals and subgoals). The weight is incremented by one, each time the control rule is used. If more than one control rule can help in solving the current goal then the rule with the maximum weight is chosen.

In the control rule store, control rules whose consequents have the same predicate are clustered together. The predicate of the goal (subgoal) is used to hash into the control rule store, leading to the control rule cluster having the same predicate as the goal (subgoal). Any of these rules (in the cluster) could (potentially) help in proving the goal. Hence the cost of using a control rule includes the unification cost of finding the maximum weighted rule (in this cluster), whose consequent unifies with the goal.

The average cost of using a control rule is the average number of unifications required to match a goal with the consequent of the control rules having the same predicate. The cost also includes the match cost of using wrong rules and facts from the database as a result of choosing a bad control rule. The savings in terms of number of matches from using a good control rule is the savings resulting from not trying useless rules and facts from the database for proving the goal.

Since the cost of unification of facts (with goals) is usually less than that of rules (with the cascading effect of proving antecedents of the rule), facts are preferred over rules in our deductive retriever.

If the control rule store (for a particular domain) includes rules like

1. The consequent is:
   abcd(X, obj1)
   The weight is: 4
   The antecedent is:
   ```
   abcd(X0,Y0) <-
           read(X0), my(X0, Y0),
           lips(Y0).
   ```

2. The consequent is:
   abcd(obj2, Y)
   The weight is: 4
   The antecedent is:
   ```
   abcd(X0,Y0) <-
           ur(X0), so(X0, Y0),
           kool(Y0).
   ```

3. The consequent is:
   abcd(X, Y)
   The weight is: 5
   The antecedent is:
   ```
   abcd(X0,Y0) <-
           make(X0), my(X0, Y0),
           day(Y0).
   ```

4. The consequent is:
   abcd(obj1, Z)
   The weight is: 7
   The antecedent is:
   ```
   abcd(X0,Y0) <-
           quid(X0), pro(X0, Y0),
           quo(Y0).
   ```

then the cost of using a control rule for proving a goal *abcd(obj2, obj1)* will include the cost of unification of the consequents of the rules enumerated above, with *abcd(obj2, obj1)*. Note the consequent of control rule 4 does not unify with the goal and hence the corresponding antecedent (a database rule) will not be used to prove the goal. However the cost of unifying *abcd(obj1, Z)* with *abcd(obj2, obj1)* contributes to the cost of using a control rule. The consequents of control rules 1, 2 and 3 unify with the goal (the cost of using a control rule includes this unification cost). However, the antecedent of rule 3 will be used to solve the goal because it has the highest weight. If this rule successfully solves the goal then the savings due to this control rule includes the savings resulting from not searching (potential) futile paths – futile paths resulting from the use of antecedents of control rules 1, 2 and/or 4 and any other rule in the database whose consequent unifies with the goal. If this control rule fails to solve the goal then the other control rules are tried (using the same procedure). The use of the control rule, in this case, leads to the exploration of futile paths which contributes to the cost. If all the control rules (namely 1, 2 and 3 for the example shown above) fail, a rule from the database (different from the antecedents of the control rules 1, 2 and 3 and whose consequent unifies with the goal) is chosen randomly to solve the goal. In this situation, contol rules have contributed only to the cost.

The results in the next section are generated according to the following learning loop:

- Control_Rule_Store = Nil
- Solve a list of testing problems and record performance;
- For each training example in the training set
  - Pick a training example and solve it
  - Append new control rules to the Control_Rule_Store
  - Solve the list of testing problems and record performance

4.2 Results

Three different domains were used in these experiments. The sentence domain consisted of 14 rules implementing a simple natural language parser. The artificial domain consists of 24 rules for determining family relationships combined with 21 artificial rules increasing the number of alternative rules applicable to certain goals. The blocks domain contains 8 rules for a situational calculus implementation consisting of one operator for transferring blocks and building towers.

Figure 2 shows the cost (averaged over 90 trials) of solving 9 testing problems in the sentence domain after learning control rules from each of 18 training problems sampled randomly from a set of 28 problems (queries). After an initial increase due to the control rules learned from the first training problem, the match cost decreased to a point below the cost of the initial rules, but then increased steadily with more training problems. The learning-cost curve follows the

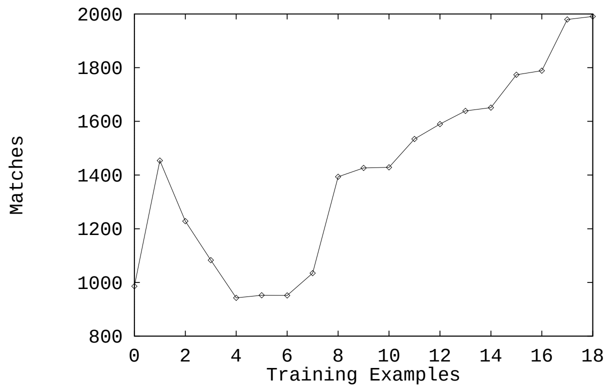

Figure 2: Sentence domain with general control rules. Match values averaged over 90 trials consisting of 18 training and 9 testing sampled from 28 queries.

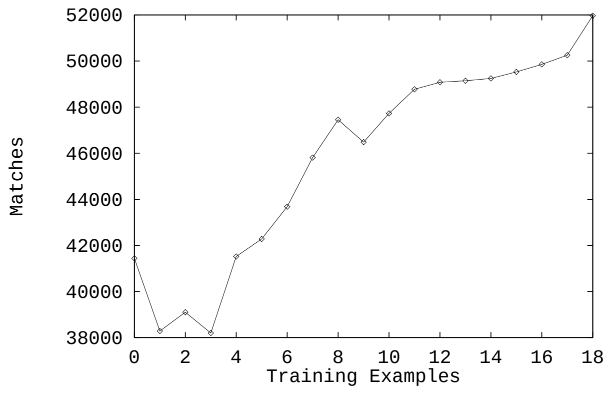

Figure 3: Artificial domain with general control rules. Match values average over 90 trials consisting of 18 training and 9 testing sampled from 28 queries.

trend of Figure 1. The minimum cost occurred after the fourth training problem.

Figure 3 shows the cost (averaged over 90 trials) of solving 9 testing problems in the artificial domain after learning control rules from each of 18 training problems sampled randomly from a set of 28 problems (queries). Again, the learning-cost curve follows the trend of Figure 1, and the minimum cost occurred after 3 training problems.

Figure 4 shows the cost (averaged over 30 trials) of solving 5 testing problems in the blocks domain after learning control rules from each of 10 training problems sampled randomly from a set of 15 problems (queries). The problems all involved building towers of height 2 from 6 blocks initially on the table. Since each

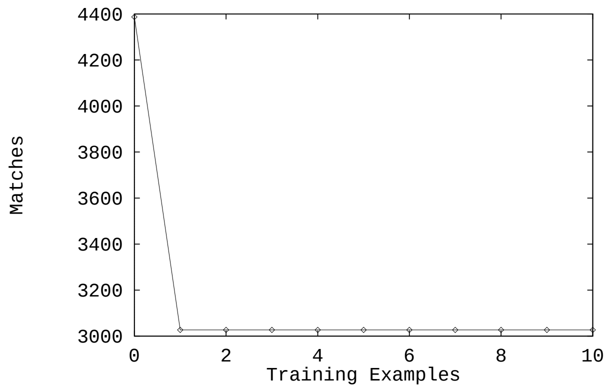

Figure 4: Blocks domain with general control rules. Match values average over 30 trials consisting of 10 training and 5 testing sampled from 15 queries consisting of towers of height 2.

query was essentially the same, the necessary control rules were learned after the first training example and remained fixed thereafter.

Figure 5 shows the cost (averaged over 30 trials) of solving 10 testing problems in the blocks domain after learning control rules from each of 20 training problems sampled randomly from a set of 30 problems (queries). The 30 queries consisted of 18 towers of height 2, 9 towers of height 3, and 3 towers of height 4. Once again, the familiar trend of the general utility problem is evident. The control rules preferred by the harder queries degrade the search for control rules effective for the easier queries.

5 DISCUSSION

Although the experimental results indicate that few training examples are necessary to learn a utile set of control rules, the main issue is how to predict the number of training examples corresponding to the minimum of the learning-cost curve. First, as is evident from Figure 3, there can be local minima in the curve. However, a single global minimum can be argued based on two factors affecting the performance behavior of the problem solver during the course of control-rule learning. One factor is the time spent testing the applicability of the control rules and following futile paths in the search space not explored by the original domain theory. The second factor is the cost savings due to the avoidance of futile paths explored by the original domain theory.

Initially, as the system learns control rules generated

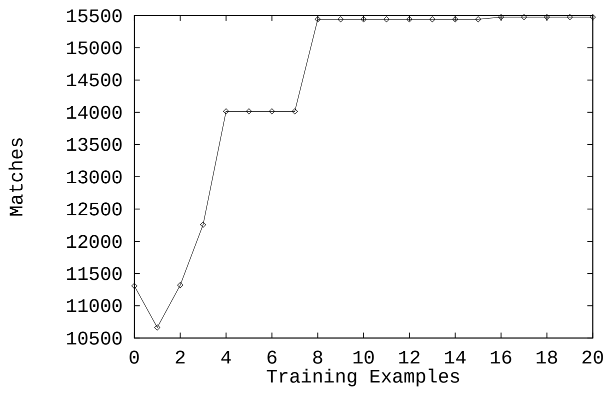

Figure 5: Blocks domain with general control rules. Match values average over 30 trials consisting of 20 training and 10 testing sampled from 30 queries consisting of towers of height 2 (18), height 3 (9) and height 4 (3).

from randomly-sampled training problems, cost increases slightly with the inclusion of low-utility control rules. However, the learning curve quickly turns downward (lower cost) as control rules are learned from training problems containing goals that are prevalent in the problem distribution. Eventually, after the utile control rules have appeared, subsequent control rule learning follows statistically insignificant trends in the problem distribution that drive up the cost of solving the testing problems. These factors combine to form a single minimum in the learning-cost curve.

Another difficulty in identifying the minimum is that several control rules are learned per training example. Finer control may be possible by limiting the number of control rules instead of training examples. Since more general control rules seem to have higher utility, control rules learned from higher levels in the explanation structure may be preferable to those learned from lower levels.

Although empirical results predict a learning-curve minimum at small numbers of training examples, no theory is available to predict this number. Currently, this point is determined empirically based on the sample of the problem distribution contained in the testing set. One direction is to consider the testing set as a pruning set and deriving lower bounds on the size of the pruning set to ensure proper identification of the minimum. However, this approach would be similar to current statistical approaches and would probably require a similar number of problems in the pruning set.

A second approach to predicting the number of training examples corresponding to minimum of the learning-cost curve is to use characteristics of the domain to predict the number of training problems necessary to ensure control rules are learned for traversing the search space in an efficient manner. We are attempting to relate domain characteristics (e.g., size and shape of the search space, size of the problem space, recursive versus non-recursive) to the probability of seeing a majority of training problems that follow a certain, highly-efficient path through the search space that is also followed by a large number of other problems prevalent in the problem distribution.

6 CONCLUSION

The simple control-rule selection strategy lies at the opposite end of the spectrum from approaches to the utility problem dependent upon large numbers of training problems to estimate the problem distribution. Empirical results indicate that few training problems are needed to learn a utile set of control rules minimizing the learning-cost curve. The next step is to compare the minimum of the learning-cost curve to the speedup obtained using other approaches to the utility problem, and compare the number of training examples at the minimum to the number suggested by statistical approaches. Eventually, a similar statistical approach will be developed for accurately predicting the necessary number of training problems based on domain characteristics. If the empirical results are indicative of behavior in other domains, there should be no need for large numbers of training problems, and a utile set of control rules can be learned with less cost.

References

Eskey, M. and Zweben, M. (1990). Learning search control for constraint-based scheduling. In *Proceedings of the Eighth National Conference on Artificial Intelligence*, 908–915.

Etzioni, O. and Minton, S. (1992). Why EBL produces overly-specific knowledge: A critique of the PRODIGY approaches. In *Proceedings of the Ninth International Conference on Machine Learning*, 137–143.

Gratch, J. and DeJong, G. (1992). COMPOSER: A probabilistic solution to the utility problem in speed-up learning. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, 235–240.

Greiner, R. and Jurisica, I. (1992). A statistical approach to solving the EBL utility problem. In *Pro-*

ceedings of the Tenth National Conference on Artificial Intelligence, 241–248.

Holder, L. B. (1990). The general utility problem in machine learning. In *Proceedings of the Seventh International Conference on Machine Learning*, 402–410.

Holder, L. B. (1992a). Empirical analysis of the general utility problem in machine learning. In *Proceedings of the Tenth National Conference on Artificial Intelligence*, 249–254.

Holder, L. B. (1992b). Unifying empirical and explanation-based learning by modeling the utility of learned knowledge. In *Proceedings of the ML92 Workshop on Knowledge Compilation and Speedup Learning.*

Markovitch, S. and Scott, P. D. (1989). Utilization filtering: A method for reducing the inherent harmfulness of deductively learned knowledge. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 738–743.

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

Minton, S. (1988b). Quantitative results concerning the utility of explanation-based learning. In *Proceedings of the Seventh National Conference on Artificial Intelligence*, 564–569.

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

Mooney, R. J. (1989). The effect of rule use on the utility of explanation-based learning. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 725–730.

Tambe, M. and Rosenbloom, P. (1989). Eliminating expensive chunks by restricting expressiveness. In *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, 731–737.

Yoo, J. and Fisher, D. (1991). Concept formation over problem-solving experience. In *Proceedings of the Twelfth International Joint Conference on Artificial Intelligence*, 630–636.