The General Utility Problem in Machine Learning

Lawrence B. Holder
University of Illinois
Beckman Institute
405 North Mathews, Urbana, IL 61801

(In the Proceedings of the Seventh International Conference on Machine Learning, 1990, pp. 402--410.)

Abstract

Experiments have revealed that uncontrolled application of the analytical learning paradigm results in knowledge having low utility. Because the performance element must consider low utility knowledge along with high utility knowledge, the proliferation of low utility knowledge eventually defeats the goal of improved performance. Experiments in empirical learning have demonstrated a similar phenomenon. Uncontrolled application of an empirical learning paradigm may result in inaccurate knowledge, and a post-processing stage is typically needed to repair the degradation in performance. The results from experimentation in both analytical and empirical learning imply a general utility problem in machine learning. This paper presents evidence for such a perspective and recommends a closer dependence between the learning paradigm and the performance goals for which it is designed. A new approach is presented along with experimentation that illustrates the applicability of the approach to the general utility problem.

1 Introduction

One of the main goals for research in machine learning is the eventual integration of learning methods with knowledge-based systems. Learning methods offer the ability to transform the knowledge of the system to improve performance on the tasks using the knowledge. The ability to transform knowledge will reduce the dependency of system performance on the quality of the knowledge initially entered by the knowledge engineer.

Recent experimentation with machine learning methods has uncovered a new obstacle to their integration with knowledge-based systems. Experiments with several analytical learning systems demonstrated an eventual degradation in performance due to uncontrolled application of the learning paradigm. This obstacle has been named the *utility problem* [Minton88]. Experiments with empirical learning methods are also uncovering this phenomenon of performance degradation due to unconstrained application.

Two additional obstacles block the integration of current learning methods with knowledge-based systems. First, the performance goals for the tasks using the knowledge may change over time. Knowledge transformations made by machine learning methods must adapt to changes in the performance goals for the desired tasks. Second, the knowledge may support different tasks from multiple domains. Knowledge transformations to improve performance on one task must preserve the performance goals of other tasks using the knowledge. These additional constraints along with the original utility problem combine to form the general utility problem. The general utility problem in machine learning is the degradation of performance for tasks using the knowledge due to the unconstrained transformation of the knowledge by machine learning methods.

Recent solutions to the utility problem have generally followed the trend of applying the learning method to every task and then pruning away the knowledge that eventually turns out to degrade performance. This research offers a different approach called performance-driven knowledge transformation that selectively applies learning methods only when necessary to achieve a desired performance goal for some task. The approach acquires knowledge for controlling the application of multiple learning methods.

The next section reviews work related to the utility problem in analytical learning and casts recent work in empirical learning in the context of the utility problem. Section 3 defines the general utility problem for machine learning and discusses alternative solutions. Section 4 describes the performance-driven knowledge transformation approach to solving the general utility problem. Section 5 illustrates experiments performed with an implementation of the approach in the Peak system. Section 6 concludes with plans for further im-

provements to the proposed approach.

2 Utility Problem in Learning

Research in both empirical and analytical learning has uncovered deficiencies in the employed methodologies. The major deficiencies stem from the naive view that the methodology in question is always applicable to the learning task and therefore should always be applied to the data. In a performance-driven system, one methodology is rarely sufficient to handle the variety of learning tasks.

2.1 Analytical Learning

Research on analytical (explanation-based) learning techniques began to focus more attention on performance with the appearance of Keller's work on the definition of operationality [Keller88]. Analytical techniques learn from a single example by proving the example is an instance of the concept to be learned. The proof terminates when the leaves of the proof tree are all operational predicates. The proof tree is then generalized, yielding an operational description of the concept. Earlier work on explanation-based learning defined an operational concept as one whose description is composed from a set of predicates deemed easy to evaluate [DeJong86, Mitchell86]. Keller pointed out that operationality is more intimately related to the performance element and the desired performance improvement. This view of operationality was used in the MetaLex system that learns heuristics for solving calculus problems. MetaLex defines an operational concept as one that improves the performance element's (problem solver's) run-time efficiency on a set of benchmark calculus problems, while maintaining effectiveness so that some percentage of the problems are still solved correctly. The increased attention on performance has led to the reevaluation of several analytical learning systems and the observation that performance may degrade with repeated application.

2.1.1 PRODIGY

In experimentation with the Morris analytical learning system, Minton found that performance degrades as the number of rules grows large [Minton85]. In order to learn a concept, the system acquires several rules whose disjunction forms the system's understanding of the concept. As the number of rules increase, the cost of determining the applicability of a rule may outweigh the benefits of applying, and thus, retaining the rule. Minton calls this phenomenon the *utility problem* and offers the Prodigy system as a solution [Minton88]. Prodigy maintains empirical estimates of match costs, application savings and frequency of application for each rule. These estimates are used to compute a utility value for the rule. If this value becomes negative, the rule is no longer considered. Minton found that maintenance of a rule's utility value and compression of the rule's conditions result in a substantial performance improvement. These results indicate that a system should be sensitive to the cost and savings of the learned descriptions.

2.1.2 SOAR

Experimentation on the Soar system has uncovered similar results [Tambe88]. Instead of monitoring the cost and benefits of rules, Tambe and Rosenbloom restrict the expressiveness of the learned rules so that the complexity of the match is kept linear in the number of matching conditions [Tambe89]. Results of using this technique within Soar indicate a greater number of less expressive rules are needed to attain the generality of the more expressive rules, but the match cost is no longer exponential. However, the results are unclear on whether an exponential number of simpler rules will be needed to achieve the generality of the more expressive rules. Also, the trend toward generating ground instances of the general rules seems contradictory to the purported benefits of analytical learning.

2.1.3 EGGS

Despite the aforementioned evidence for degrading performance in analytical learning systems, other such systems have demonstrated improved performance without concern for the number or form of the learned rules. Looking at systems by O'Rorke [O'Rorke87] and Shavlik [Shavlik88] Mooney recently uncovered the reason for the contradictory results [Mooney89]. The performance element for Mooney's experiments was Eggs [Mooney86], which includes a Horn-clause theorem prover and standard explanation-based learning techniques [DeJong86, Mitchell86] for generalizing the proofs.

Experiments with Eggs revealed that limited use of the learned rules provided greater performance in accuracy and speed than full use. Because Shavlik constrained the proofs to be no longer than a specified depth bound, his system was making only limited use of the learned rules (i.e., only those rules that required limited chaining).

Mooney also demonstrated that using a breadth-first search for theorem proving, instead of depth-first, also forced a limited use of learned rules. Learned rules that would have required deep sub-goaling to reach a solution are circumvented by the simultaneous consideration of proofs from the original domain theory. The use of breadth-first search in O'Rorke's system accounts for much of the favorable performance. Mooney concludes that limited use of learned rules is advis-

able until the system has learned the rules necessary to solve the more common problems.

2.1.4 Summary

Experimentation on analytical learning systems demonstrates performance degradation with uncontrolled application of the paradigm. In response, many researchers have opted for more specific instances of the learned rules. The level of specificity of the knowledge should not be arbitrary, but determined by the desired performance. In fact, with performance-directed learning the original domain theory may perform within desired performance thresholds, in which case, the application of analytical learning may be unnecessary.

2.2 Empirical Learning

Empirical learning methods have traditionally been designed to achieve the best classification performance possible. However, experimentation described below indicates that classification performance can actually degrade with repeated application of the empirical learning method.

2.2.1 AQ

During experimentation with the AQ system (specifically, AQ15 [Michalski86]), Michalski found that repetitive application of AQ can yield less accurate concepts than a more conservative application strategy combined with a simple inference mechanism [Michalski87]. The AQ methodology finds a conjunctive description that covers as many positive examples as possible without covering any negative examples. Positive examples not covered by the first description are used as input for another execution of AQ. This procedure continues until a concept in disjunctive normal form is produced covering all the positive examples and none of the negative examples. Michalski compared the accuracy of the DNF concept with that of the concept consisting of only the single disjunct covering the most positive examples. Using a simple matching procedure, the truncated concepts out-performed the original concepts in both accuracy and speed. This observation illustrates the need for systems to be more selective in their own behavior when such selectivity is sufficient to achieve the desired performance goals.

2.2.2 ID3

Similar results have been obtained with the decision trees generated by Quinlan's ID3 program [Quinlan86]. Quinlan found that *pruning* the rules extracted from a decision tree can improve the accuracy of the rules on unseen examples [Quinlan87]. ID3 builds decision trees by selecting an attribute from the training examples providing the best split (according to an information theoretic criterion) between positive and negative examples. The program continues by descending each branch and recursively applying itself to the examples satisfying the attribute value for that branch. ID3 halts when all the nodes at the frontier of the tree contain all positive or all negative examples. The pruning stage removes rules from the decision tree until accuracy on a set of test examples begins to decrease. Compared to the original rules, the pruned rules performed better on a set of unseen test examples. Although the success of pruning is due to noise, missing values, and the decreased number of examples available at higher depths in the tree, this stage might have been unnecessary if the desired accuracy had been taken into account during the initial generation of the decision rules.

2.2.3 Summary

Research on empirical learning has shown that inexact rules combined with a simple matching procedure can be less expensive and more accurate than exact rules. The tradeoff between accuracy and completeness of the learned rules should be decided by the desired performance. Modifying the AQ algorithm to return only the one disjunct covering the most positive examples may avoid generation of low utility knowledge. Likewise, constraining the ID3 algorithm to stop when the leaves have reached a desired level of accuracy may prevent inaccuracies at greater depths in the decision tree and avoid the need for pruning.

Typically, empirical learning methods are invoked to achieve the best classification accuracy possible. To avoid a degradation in classification performance, empirical learning methods should be invoked only when necessary to achieve a violated performance goal. Furthermore, repair of the performance goal violation may require only modest generalization as opposed to the large inductive leaps made by most empirical learning methods. In the extreme case (e.g., small number of instances in the concept), rote learning may be preferable to more powerful empirical learning methods.

3 General Utility Problem

The generation of low utility knowledge by both analytical and empirical learning methods indicates a utility problem in machine learning more widespread than that identified in the analytical learning literature. The general utility problem encompasses not only the performance degradation on one task due to uncontrolled application of learning methods, but also adaptation to changing task performance goals and maintenance of performance on other tasks using the knowledge. Thus, the general utility problem is informally defined as follows:

General Utility Problem: performance degradation on one or more tasks due to the transformation of knowledge.

In order to address the general utility problem, the role of machine learning methods must be viewed from a purely performance-based perspective. This perspective is similar to that used by Keller in the METALEX system [Keller87]. METALEX transforms its knowledge base by retaining a new concept only if the concept improves the efficiency and effectiveness of the performance element.

Markovitch and Scott address the general utility problem by filtering the information flow from instances, to the knowledge base, and then to the performance element [Markovitch89]. One filter, the *utilization* filter, removes harmful rules from the knowledge used by the performance element.

Learning control knowledge for the application of learning methods to different tasks is addressed by Rendell's variable-bias management system (VBMS) [Rendell87]. VBMS maps different tasks to points in a bias space. Each point in bias space represents a choice of inductive algorithm, representation language and any relevant parameters for the algorithm or language.

The next section describes an approach to the general utility problem called performance-driven knowledge transformation. This approach differs from the approach in METALEX by making performance goals more explicit and incrementally adapting to changes in desired performance. In contrast to the knowledge filtering approach, performance-driven knowledge transformation constrains the initial generation of knowledge as directed by failure performance goals. This approach differs from the approach in VBMS in that the emphasis is on selecting learning methods to repair violations in desired performance, not to achieve maximum possible performance on an isolated learning task.

4 Performance-Driven Knowledge Transformation

Performance-driven knowledge transformation controls the application of learning methods based on their ability to achieve desired performance goals on one task while preserving the performance on other tasks. Each task for the knowledge base defines a performance space. The dimensions of the performance space are the performance goals (e.g., completeness, correctness, response time) to be maintained by the knowledge base for that task. The current state of the knowledge base is represented by a point in the performance space for each task. A knowledge transformation can be viewed as a move of the current knowledge base from one point in the performance space of each task to another. Figure 1 shows the performance spaces for two tasks. Task A (Figure 1a) consists of three performance goals G_1, G_2 and G_3. Task B (Figure 1b) consists of two performance goals G_4 and G_5. The location of two knowledge bases K_1 and K_2 are shown for each task.

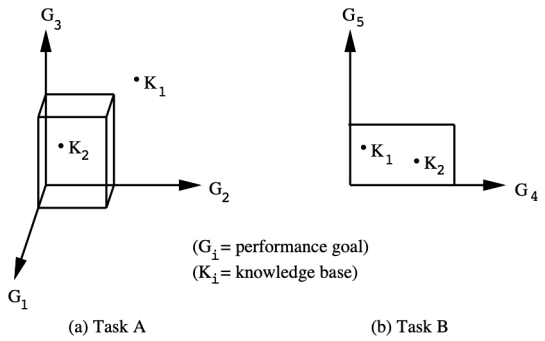

Figure 1: Performance Spaces for Two Tasks

The desired performance for each task defines a hyper-rectangle in that task's performance space. When the knowledge base moves outside the desired-performance hyper-rectangle in some performance space, performance-driven knowledge transformation selects a learning method to transform the knowledge base so that the corresponding point in the performance space for the current task moves back inside the desired-performance hyper-rectangle without moving the point outside the desired hyper-rectangle in the performance spaces for other tasks. Referring to Figure 1, knowledge base K_1 has satisfactory performance for task B, but violates the performance goals of task A. Transforming knowledge base K_1 to K_2 achieves the performance goals of task A and preserves the satisfactory performance for task B.

This research proposes an approach to performance-driven knowledge transformation implemented in the PEAK system. When a performance goal violation is detected while solving a problem from some task, the PEAK system uses information about the context of the goal violation (e.g., the difference between desired and actual performance) to select a transformation operator for reducing this difference while maintaining other performance levels. Application of the operator yields a new knowledge base. If the new knowledge base achieves the violated performance goal and preserves other performance goals, then the current knowledge base is replaced by the new knowledge base. Otherwise, another transformation operator is selected for application. Verification of the new knowledge base

is accomplished by using the knowledge to solve previously seen problems from the same task. For each operator, PEAK retains information about the applicability of the operator in a given context based on the success of the operator in reducing the goal violation. As more performance goal violations are repaired, PEAK demonstrates more intelligent selection of transformation operators and quicker convergence to a knowledge base within desired performance thresholds.

In the following discussion, certain assumptions have been made about the knowledge in the knowledge base and the performance element using this knowledge. The knowledge base is a set of Horn clause rules. The performance element is a deductive retriever similar to Prolog. Performance is measured while the performance element attempts to solve a query posed by the user. Attached to the query are the performance goals to be maintained during the solution of the query. Performance goal violations occur when the measured performance exceeds the desired thresholds.

4.1 Performance Perspective

Using performance goals as a means of guiding the maintenance and repair of a knowledge base requires a precise definition of performance. The definition of performance depends on the perspective. Four perspectives are applicable for describing the performance of a knowledge base:

- **External** performance is the performance measured from outside the knowledge base, regardless of any internal knowledge transformations.
- **Current** performance is the performance the system currently maintains for the previously seen queries.
- **Expected** performance is the performance the system expects to demonstrate on future queries. Expected performance is usually the same as current performance.
- **Absolute** performance is the performance that the current state of the knowledge would support if given every possible query.

When the user specifies a threshold for some performance measure, the proper perspective must be used to evaluate the performance of the knowledge base. *Absolute* performance is rarely available due to a lack of knowledge about the instance space. *Absolute* performance is inappropriate, because the distribution over the entire instance space may not give equal probability to each instance. *External* performance provides information about the rate of convergence towards absolute performance. Changes in *external* performance indicate the need for an increase or decrease in the extent of the knowledge transformations. *Current* performance evaluates the knowledge only on previously seen queries. *Expected* performance is the best measure of the current state of the knowledge base, because the objective of the knowledge base is to maintain its expected ability to perform the task within desired thresholds on possibly unseen queries.

Both *expected* and *external* performance should be measured by the performance-driven knowledge transformation process. Knowledge transformations are triggered only when *expected* performance falls below desired levels. *External* performance should then be used in the selection of an appropriate transformation operator. The greater the difference between *external* and *expected* performance, the more drastic a transformation should be recommended by the system.

4.2 Information on Goal Violations

Once a goal violation has been detected, several pieces of information are available for selecting an appropriate knowledge transformation operator. First, as described in the previous section, the difference between expected and external performance indicates the extent of the necessary transformation.

Second, after the performance element attempts to solve a query, the violated and preserved goals are known. Each goal contains information about the performance measure that this goal constrains, the desired threshold on the measure, the observed value of the measure on previously seen queries (including the query just processed), and the difference between the observed and desired performance (the error). The performance measure constrained by a violated goal is useful for selecting transformation operators capable of improving this performance measure. The magnitude of the error indicates the extent of the transformation. The performance measure constrained by a satisfied goal is useful for selecting transformation operators capable of preserving this performance measure. The magnitude of the error indicates the extent to which the selected operator may degrade performance on the satisfied goals in order to achieve performance on the violated goals.

A third source of information that will be available upon detection of a performance goal violation is the *task history*. Each task known to the knowledge base maintains a task history of previously seen queries from the task. The task history serves two purposes. First, the task history represents an empirical estimate of the distribution over the possible queries of the task. This distribution can be used to verify the achievement of violated performance goals in transformed knowledge. Second, an entry in the task history contains information about the query-solving episode. One useful

piece of information about a query-solving episode is the trace of the knowledge accessed during the solution.

The *knowledge trace* is an and/or tree that records the knowledge accessed during the solution of the query and indicates which rules (if any) support the response to the query. Information about the shape of a task's knowledge traces constrains the selection of knowledge transformations. For example, wide, shallow knowledge traces indicate that the knowledge consists of specific instances of the task; whereas narrow, deep knowledge traces indicate a more general set of rules for proving queries from the corresponding task.

Finally, past success of the transformation operators provides information upon performance goal violation. As the knowledge base transforms to meet performance goals, a record is kept of the old and new knowledge bases along with the operator responsible for the transformation. If the new knowledge base achieves a violated goal while preserving non-violated goals, then the system increases the operators applicability for achieving and preserving the appropriate goals. Over time, collection of this information will allow the system to make a more informed operator selection based on past experience.

4.3 Verification of Knowledge Base

Because no operator application is guaranteed to achieve the desired results, the system must verify that the knowledge base resulting from an operator application achieves the desired performance. Verification can be accomplished by re-solving the queries in the task history. The size of the task history can be changed to tradeoff performance convergence rates for transformation speed. As the system learns operator applicability, there is less chance that several alternative operator applications must be tried before finding one that achieves the violated goal. Because each transformation attempt requires verification of the resulting knowledge base, the fewer attempts necessary implies fewer verifications; thus, the task history size can increase over time.

5 Experimentation

This section illustrates the application of PEAK on two tasks from diverse domains. The first experiment involves learning to improve response time, completeness and correctness while determining whether to land the space shuttle manually or automatically depending on environmental conditions. The second experiment involves learning to improve response time while constructing plans to build towers in the blocks-world domain. Together, the two experiments demonstrate the ability of performance-drive knowledge transformation to selectively apply appropriate learning methods to achieve desired performance goals.

5.1 Shuttle Domain

This experiment executes the PEAK system on the shuttle landing control database available from the machine learning databases maintained by University of California at Irvine. The problem is to determine whether to land the shuttle manually or automatically based on environmental attributes. The corresponding task is labeled the *landing* task, and the queries are of the form `landing(ENV,?x)`. The `ENV` in the query represents the environmental situation to be evaluated. The performance element attempts to fill in the `?x` with the recommended landing control: `auto` or `noauto`.

Prior to query answering, the user inputs the performance thresholds to be maintained by the knowledge base while answering *landing* queries using the performance element (a backward-chaining deductive theorem prover for Horn clauses). For this experiment, three performance goals are specified: *correctness*, *completeness* and *response time*. The correctness goal specifies that the answers to queries must be correct 90% of the time. The completeness goal specifies that the query must be answered 95% of the time. That is, the answer should be either `auto` or `noauto` and not *"I don't know"*. The response time goal specifies that the performance element must respond within 10 seconds.

Two knowledge transformation operators are available: rote learning and empirical learning. Application of the rote learning operator asks the user for the correct answer to the query. A new rule is added to the knowledge base having the instantiated query as the consequent, and the facts defined before query execution as the antecedent. The empirical learning operator utilizes the ID3 program to build a decision tree from examples in the knowledge base. The examples are rules such as those learned by the rote operator. Each path in the resulting decision tree is converted to a rule. The examples are replaced by the new rules in the transformed knowledge base.

Starting with an empty knowledge base, PEAK attempts to solve *landing* queries, while maintaining the performance goals. Figure 2 plots the three performance goals for 200 randomly chosen queries from the shuttle landing control domain.

Figure 2a illustrates how PEAK maintains response time performance below 10 seconds. For the first 30 queries, response time increases as the number of rote-learned rules increases. Eventually, the large number of rules in the knowledge base cannot be traversed within the response time threshold.

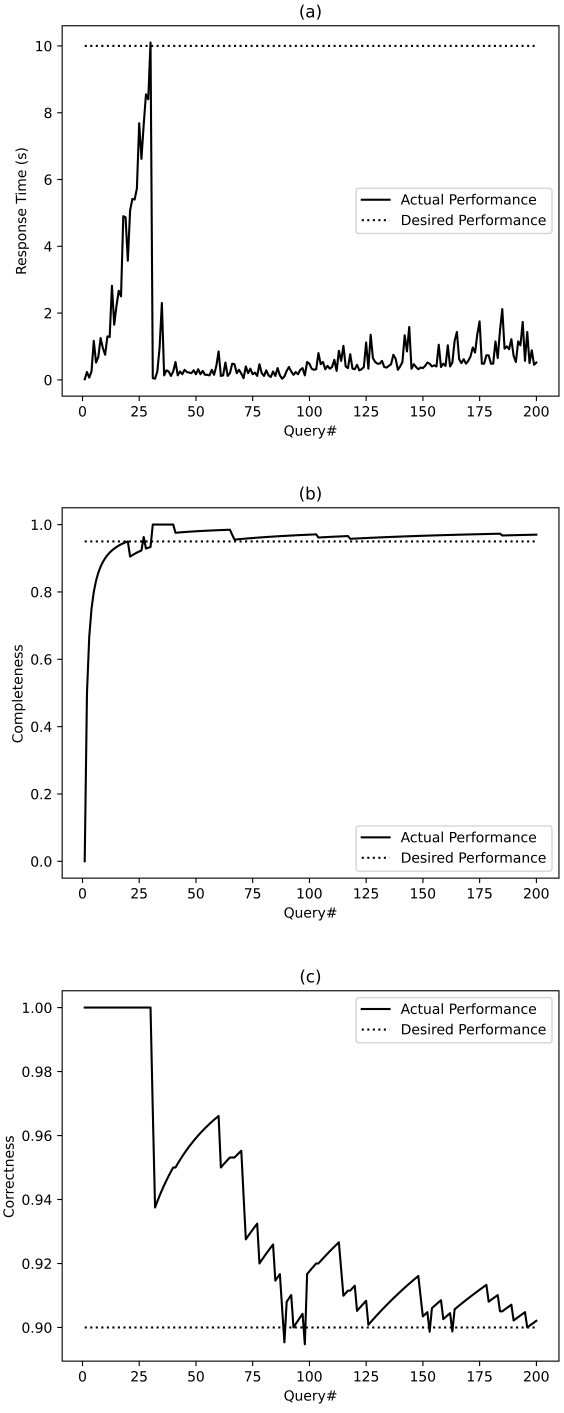

Figure 2: Plots of Performance for Shuttle Domain

While processing the 30th query, PEAK was unable to solve the query, generating a completeness failure. PEAK first trys to transform the knowledge base by rote-learning a new rule. However, verification of the new knowledge base uncovers a response time failure. Because the rote learning operator was ineffective, PEAK chose to apply the ID3 operator. ID3 generalized the 29 learned instances into 8 general rules. As Figure 2a indicates, the resulting transformation drastically improves response time performance.

The plot of completeness performance in Figure 2b illustrates how PEAK quickly learns the initial query knowledge. After the ID3 transformation, completeness remained above the 95% threshold for the remainder of the 200 queries.

The correctness plot in Figure 2c shows how performance starts at 100% and converges to the desired 90% threshold. The initial values of 100% for correctness are due to the fact that many of the initial queries could not be answered. Correctness performance only measures the correctness of answered queries. Immediately following the application of ID3, correctness falls to 94% due to the next two queries being incorrectly answered according to the new knowledge base. As query answering continues, the over-generalization in the rules eventually brings correctness down below the 90% threshold. Correctness violations occur at queries 89, 98, 153 and 163. In each case, PEAK uses the rote-learning operator to memorize the incorrectly answered query and restore 90% correctness performance.

The final knowledge base after completion of the 200 queries consists of the 12 rules shown in Figure 3. Rules 5-12 are the general rules learned by ID3. Rules 1-4 are the specific instances learned to repair the over-generalization in ID3's rules. After 200 queries, the knowledge base converged to 8 general rules describing major trends in the shuttle landing domain and four specific rules for special cases not handled correctly by the general rules.

One final observation from Figure 2 is the convergence of the performance towards the desired thresholds and not towards the maximum possible performance. This indicates how performance-driven knowledge transformation utilizes flexibility in one dimension of performance to improve performance in another dimension.

5.2 Blocks-World Domain

In the task from the blocks-world domain, the user asks the performance element to construct a plan for building a tower of blocks. The queries are of the form `tower(A B C ?state)`, where `A`, `B` and `C` are blocks, and `?state` is a variable to be instantiated with the plan for achieving the tower.

Prior to query answering, the user inputs the performance thresholds to be maintained by the knowledge base while answering *tower* queries. For this experiment, one performance goal is specified: response-time < 10 seconds. Performance goals for completeness and correctness are inappropriate, because the domain theory is assumed complete and correct.

In addition to the rote learning and ID3 operators used with the first experiment, an explanation-based generalizer, EGGS [Mooney86], is included in the PEAK system. EGGS applies standard explanation-based techniques [DeJong86, Mitchell86] to generalize the proofs obtained by the performance element. When EGGS is applied to a proof, the result is a general rule that is added to the knowledge base.

Starting with the blocks-world domain theory, PEAK attempts to solve *tower* queries, while maintaining the response time performance goal. The initial state of the blocks world contained six blocks. Figure 4 shows the response time obtained by PEAK for 100 semi-randomly chosen *tower* queries. Semi-random means that the first ten queries were all towers of height two (the two blocks to be in the tower were chosen randomly from the six blocks in the initial state). The second ten queries were all towers of height three, and so on for the first 50 queries. The second 50 queries repeat the above sequence to show the effects on response time of the rules learned during the first 50 queries.

As shown in Figure 4, the first 20 queries (towers of height two and three) are solved by the original domain theory within the response time threshold. However, the domain theory is unable to maintain the response time performance goal while processing the 21st query (tower of height four). At this point, ID3 cannot be applied due to the lack of examples in the knowledge base. The EGGS operator is chosen over rote learning due to the knowledge trace for the query. The deep, wide proof tree suggests that EGGS is more likely to succeed than ID3.

Application of EGGS yields a general rule that builds any tower of height four in one step. Thus, the remainder of the *tower* queries for height four are completed within the response time threshold. Similar rules are learned for the 31st query (tower of height five) and 41st query (tower of height six). Figure 4 shows that retrying the towers of heights two through six (queries 50-100) results in no response time performance violations due to the previously learned rules.

This experiment demonstrates PEAK's ability to constrain the application of the EGGS analytical learning algorithm. Application of EGGS was unnecessary for towers of height two and three, because the original

1. landing(x,noauto) ← sign(x,nn) & wind(x,head) & stability(x,xstab) & error(x,MM) & magnitude(x,Medium) & visibility(x,yes)
2. landing(x,noauto) ← sign(x,pp) & wind(x,tail) & stability(x,xstab) & error(x,MM) & magnitude(x,Low) & visibility(x,yes)
3. landing(x,noauto) ← sign(x,nn) & wind(x,head) & stability(x,stab) & error(x,MM) & magnitude(x,OutOfRange) & visibility(x,yes)
4. landing(x,noauto) ← sign(x,nn) & wind(x,tail) & stability(x,xstab) & error(x,MM) & magnitude(x,Low) & visibility(x,yes)
5. landing(x,auto) ← error(x,MM) & visibility(x,yes)
6. landing(x,auto) ← stability(x,stab) & error(x,SS) & magnitude(x,Strong) & visibility(x,yes)
7. landing(x,auto) ← visibility(x,no)
8. landing(x,noauto) ← error(x,XL) & visibility(x,yes)
9. landing(x,noauto) ← error(x,LX) & visibility(x,yes)
10. landing(x,noauto) ← stability(x,xstab) & error(x,SS) & magnitude(x,Strong) & visibility(x,yes)
11. landing(x,noauto) ← error(x,SS) & magnitude(x,OutOfRange) & visibility(x,yes)
12. landing(x,noauto) ← error(x,SS) & magnitude(x,Low) & visibility(x,yes)

Figure 3: Shuttle Domain Knowledge Base After 200 Queries

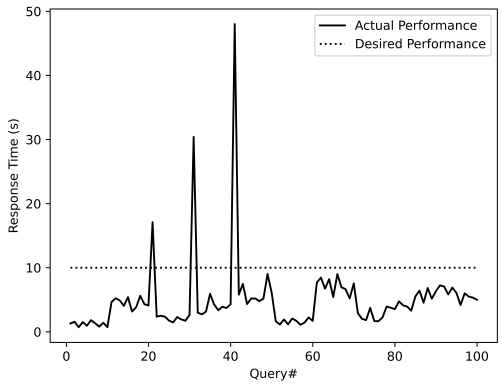

Figure 4: Plot of Response Time for Tower Domain

domain theory was able to solve these queries within the desired performance thresholds. However, the domain theory was unable to support the desired performance for towers of height four, five and six, requiring three applications of EGGS to learn general rules for these specific cases. As the performance on the second 50 queries indicates, the original domain theory plus the three learned rules was sufficient to maintain the desired performance for the tower-building task.

6 Conclusions

In order to integrate machine learning methods with knowledge-based systems, the general utility problem in machine learning must be addressed. Evidence for the general utility problem has been found in experimentation on both analytical and empirical learning methods. Unconstrained application of these methods has been shown to degrade the performance they were designed to improve. Learning methods should be invoked only after a performance failure has been detected, and then, only if the learning method is applicable to the properties of the failure. Furthermore, learning methods must permit transformation of the knowledge to achieve performance goals without violating the performance goals associated with other tasks using the knowledge. The learning methods must also have the ability to adapt to changing performance goals.

Performance-driven knowledge transformation offers an approach that addresses the general utility problem. Learning methods are invoked only when necessary to improve performance, and in accordance with previous success in repairing the violated performance goal. The performance-driven knowledge transformation approach has been implemented in the PEAK system. Experimentation with PEAK demonstrates the ability to control the application of learning methods to achieve desired performance goals.

More experimentation is necessary to validate the use of performance-driven knowledge transformation. Experiments with modified versions of the AQ and ID3 algorithms will indicate the usefulness of these transformation methods to avoid inaccuracies due to unconstrained application of the paradigms. Experiments with the interaction of multiple learning methods will indicate the usefulness of current control knowledge and suggest the need for other knowledge to control the performance-driven knowledge transformation process.

7 Acknowledgments

I would like to thank Robert Stepp for his insightful comments on this work. I would also like to thank Diane Cook, Brad Whitehall and Robert Reinke for their helpful suggestions. The ID3 and EGGS operators were adapted from versions written by Ray Mooney. The blocks-world domain theory was adapted from rules written by Jude Shavlik and Ray Mooney.

8 References

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

[Keller87] R. M. Keller, "Concept Learning in Context," *Proceedings of the 1987 International Machine Learning Workshop*, Irvine, CA, June 1987, pp. 91-102.

[Keller88] R. M. Keller, "Defining Operationality for Explanation-Based Learning," *Artificial Intelligence 35*, 2 (June 1988), pp. 227-241.

[Markovitch89] S. Markovitch and P. D. Scott, "Utilization Filtering: A Method for Reducing the Inherent Harmfulness of Deductively Learned Knowledge," *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI, August 1989, pp. 738-743.

[Michalski86] R. S. Michalski, I. Mozetic, J. Hong and N. Lavrac, "The Multi-Purpose Incremental Learning System AQ15 and its Testing Application in Three Medical Domains," *Proceedings of the National Conference on Artificial Intelligence*, Philadelphia, PA, August 1986, pp. 1041-1047.

[Michalski87] R. S. Michalski, "How to Learn Imprecise Concepts: A Method for Employing a Two-Tiered Knowledge Representation in Learning," *Proceedings of the 1987 International Machine Learning Workshop*, Irvine, CA, June 1987, pp. 50-58.

[Minton85] S. Minton, "Selectively Generalizing Plans for Problem-Solving," *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, Los Angeles, CA, August 1985, pp. 596-599.

[Minton88] S. Minton, "Quantitative Results Concerning the Utility of Explanation-Based Learning," *Proceedings of the National Conference on Artificial Intelligence*, St. Paul, MN, August 1988, pp. 564-569.

[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.

[Mooney86] R. J. Mooney and S. W. Bennett, "A Domain Independent Explanation-Based Generalizer," *Proceedings of the National Conference on Artificial Intelligence*, Philadelphia, PA, August 1986, pp. 551-555.

[Mooney89] R. J. Mooney, "The Effect of Rule Use on the Utility of Explanation-Based Learning," *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI, August 1989, pp. 725-730.

[O'Rorke87] P. V. O'Rorke, "LT Revisited: Experimental Results of Applying Explanation-Based Learning to the Logic of Principia Mathematica," *Proceedings of the 1987 International Machine Learning Workshop*, Irvine, CA, June 1987, pp. 148-159.

[Quinlan86] J. R. Quinlan, "Induction of Decision Trees," *Machine Learning 1*, 1 (1986), pp. 81-106.

[Quinlan87] J. R. Quinlan, "Generating Production Rules from Decision Trees," *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, August 1987, pp. 304-307.

[Rendell87] L. Rendell, R. Seshu and D. Tcheng, "Layered Concept Learning and Dynamically-Variable Bias Management," *Proceedings of the Tenth International Joint Conference on Artificial Intelligence*, Milan, Italy, August 1987, pp. 308-314.

[Shavlik88] J. W. Shavlik, "Generalizing the Structure of Explanations in Explanation-Based Learning," Ph.D. Thesis, Department of Computer Science, University of Illinois, Urbana, IL, January 1988.

[Tambe88] M. Tambe and A. Newell, "Some Chunks Are Expensive," *Proceedings of the 1988 International Machine Learning Workshop*, Ann Arbor, MI, June 1988, pp. 451-458.

[Tambe89] M. Tambe and P. Rosenbloom, "Eliminating Expensive Chunks by Restricting Expressiveness," *Proceedings of the Eleventh International Joint Conference on Artificial Intelligence*, Detroit, MI, August 1989, pp. 731-737.