Search Control, Utility, and Concept Induction*

Brian Carlson, Jerry Weinberg, & Doug Fisher
Department of Computer Science
Box 1679, Station B
Vanderbilt University
Nashville, TN 37235

Abstract

Our research adapts incremental conceptual clustering (or concept formation) to the task of learning to guide search. We build on earlier research that uses concept induction techniques to learn search control, but our approach differs by virtue of its reliance on probabilistic, hierarchical classification schemes that increase certain aspects of search efficiency. The system also includes inductive strategies of 'noise tolerance' that mitigate problems of control knowledge 'utility'. A general lesson is that recently identified search 'utility' problems are synonymous with inductive problems of 'noise'; solutions to the problems of the latter type can be usefully adapted to the former.

1 Introduction

An important objective of machine learning research is to improve the *efficiency* of search. This includes the compilation of operator sequences into macro-operators and the adaptation of object concept learning methods to guide operator application (Mitchell, Utgoff, & Banerji, 1983; Langley, 1985). However, recent research has qualified the naive application of these techniques: learned search control knowledge varies in its *utility* (Minton, 1988). In the worst case, learned knowledge can have a detrimental effect on search since the search to find applicable learned knowledge can be more costly than the search that an uninformed system would require.

This paper illustrates that *incremental conceptual clustering* or *concept formation* can organize search control knowledge for efficient reuse. In particular, operator choices made during successful searches are clustered into 'similarity' classes that capture the shared context in which operators were applicable. Operator selection during later search is guided by classification of contextual information. However, reliance on classification is qualified by 'noise tolerant' strategies that demonstrably mitigate the 'utility' problem. In fact, a general observation of our work is that problems of 'noise' in inductive concept learning and problems of 'utility' in search control learning are closely linked in form and in solution.

*This research was supported by NASA Ames grant NCC 2-645.

2 Search and Concept Induction

There are two facets to the problem of learning search control knowledge: generating plausible abstractions of when operators should be applied and filtering these abstractions for their utility (Etzioni, 1988). These facets correspond to similar aspects of concept learning. This connection has been traditionally recognized with respect to the generation of plausible abstractions. Early work on systems such as SAGE (Langley, 1985) and LEX (Mitchell, Utgoff, & Banerji, 1983) applied empirical concept learning techniques to complete solution traces, thus inducing conditions under which an operator's application previously led to a goal state. Using information-theoretic methods, Rendell, Seshu, and Tcheng (1987) used PLS1, a 'utility' clustering system, to group problems with similar solution strategies. Purely analytic concept learning approaches to uncovering search control knowledge have also been investigated under the rubric of *explanation-based* learning (DeJong & Mooney, 1986; Mitchell, Keller, & Kedar-Cabelli, 1986). These techniques use analytic, typically deductive strategies to find conditions under which 'operators' should apply from a small number of 'solution' traces.

Until recently, work on learning search control has focused almost exclusively on the generation of plausible abstractions. However, recently there has been the observation that rule application varies in utility: the degree that rule application alters the number of steps (i.e., subproblems, states) encountered dur-

ing search.[1] Rule application in certain contexts may actually detract from search efficiency. Several approaches to eliminating harmful rule application have been examined. For example, related techniques by Hansson and Mayer (in press) and Wefald and Russell (1989) assess whether significant information is gleaned about eventual goal satisfaction from further state expansion. Probability estimates used in this assessment are learned by analyzing the state space expanded during search. With sufficient experience probability estimates terminate expansion at states from which it is unlikely that useful information can be found by further search (e.g., the system may be very certain about eventual goal achievement from the current state, thus diminishing the need for further search since it is unlikely to yield significant new insights about eventual outcomes).

Recently, research in explanation-based learning has employed similarly-intended, though differently-implemented methods for controlling search (Minton, 1988; Mooney, 1989). Research in this area has focused on the efficacy of exploiting learned rules versus simply using the primitive operators. For example, Markovitch & Scott (1989) learn probability estimates that subgoals can be satisfied. Learned rules are not used in proof attempts of subgoals that are not likely to be successful; only primitive rules are used in these cases, thus avoiding redundant search.

As we have noted, work with SAGE and LEX recognized the applicability of concept induction methods to generate plausible search control rules. Similarly, we believe that utility can be tested by noise-tolerant strategies of concept learning. For example, ID3 (Quinlan, 1986) generates decision trees from training data using a measure of the information transmitted about class membership (e.g., disease) by each attribute used to describe objects (e.g., patient case histories). The values of the most informative attribute label arcs of the tree and are used to divide the training set; the information-theoretic measure is used to recursively divide each training subset, thus forming a decision tree. During tree construction, an estimate of whether 'significant' information is gained about class membership is made by a chi-square heuristic. Similar to Wefald and Russell (1989) and Hansson and Mayer (in press), tree expansion is terminated at nodes where the divisive attribute does not transmit significant information about class membership; at this point, the most common class among the training subset is used to label the appropriate leaf of the decision tree. Subsequent data is classified by traversing appropriate paths of the tree to a leaf, where an appropriate class designation resides. Quinlan and others (Michalski, 1987) have shown that this general strategy eliminates the use of 'low utility' concepts (or portions thereof); classification accuracy is not adversely affected or is actually improved by the process.

In the following two sections we describe the application of empirical concept learning to the induction, organization, and exploitation of search control rules. Like earlier research, we are concerned with the generation of plausible concepts and control rules. Our work in this area is distinguished by the use of *probabilistic* concepts (i.e., control rules), a representation scheme that allows partial matching. Moreover, our work is further distinguished by its attention to the connection between post-generation tests of utility in traditional concept learning systems and search control learning.

3 The COBWEB Concept Formation System

Empirical concept learning is typically concerned with improving prediction accuracy. In search control learning this translates into a concern for accurately predicting the operator that should be applied under current conditions; more accurate predictions result in a more directed, efficient search. For example, the search-intensive task of language recognition is highly anticipatory; a parser expects (i.e., predicts) a particular symbol next on the input stream. If the next symbol is not as expected then the parser has made an incorrect prediction; this may actually reflect on an incorrect prediction that was made several symbols earlier but has only caused a contradiction after subsequent processing. In the case of the phrase *big blue bugle boy*, the subphrase *big blue* might be a nickname, *big blue bugle* might refer to a large blue brass instrument, but the intent of the phrase is that *big*, *blue*, and *bugle* all modify the noun *boy*. Of the relevant parsing operators, (PARSE adjective) and (PARSE noun), neither can be predicted with certainty at intermediate points in the phrase. As with any search-intensive system, the parser must backtrack to an indeterminate depth and try alternatives until contradictions are eliminated.

3.1 Hierarchy Generation

To reduce backtracking we wish to better predict the likelihood that an operator applies under current conditions. Our particular approach to improving search efficiency is through a conceptual clustering system, COBWEB (Fisher, 1987; 1989). This is an untutored system that incrementally builds classification trees from objects that are described by nominal attribute-value pairs. Stored at each node of the tree are the value distributions of each attribute over the objects classified under the node. For example, if 90% of the objects stored under a node, n, are blue, then the `blue` feature would be weighted accordingly: $P(\texttt{Color} = \texttt{blue}|n) = 0.9$. Each node is a *probabilistic concept*

[1] Recent work also points out that search control rules vary in their match cost – a test of a rule's applicability (Minton, 1988; Tambe & Rosenbloom, 1989). Our work thus far concentrates on reducing the states examined during search, although we will return to issues of match cost in Section 5.

(Smith & Medin, 1981); the classification tree is a *probabilistic concept tree.*

Each tree level contains sibling classes that collectively partition the observed objects. COBWEB can incrementally incorporate a new object into the class that best matches the object according to *category utility* (Gluck & Corter, 1985), a measure that rewards the formation of object classes that improve 'prediction ability'. More formally, the category utility of a class, C_k, is a function of the *expected number* of attribute values that can be correctly predicted about members of the class, $E(\#CorrectPredictions|C_k)$. This expectation can be further formalized in terms of conditional probabilities that are stored at tree nodes: $E(\#CorrectPredictions|C_k) = \sum_i \sum_j P(A_i = V_{ij}|C_k)^2$. Category utility has some additional complexities (Gluck & Corter, 1985; Fisher, 1987), but for our purposes it is sufficient to note that category utility is a function of:

$$P(C_k) \sum_i \sum_j P(A_i = V_{ij}|C_k)^2,$$

where $P(C_k)$ is the proportion of the observations to which the expectation applies; $P(C_k)$ is the probability that the benefits (i.e., expectations) of a class will be realized.

COBWEB incrementally filters objects into appropriate classes based on category utility. A new object is evaluated with respect to a class by tentatively placing the object in the class; each class's attribute-value distributions are tentatively updated to reflect the values of the new object. Probabilities are computed from the tentatively updated distributions and the category utility of the class is computed. The class that maximizes category utility after adding the new object is chosen to classify the object and appropriate distributions are updated permanently. This process is recursively applied to the subtrees rooted at the selected child until a leaf is reached. A leaf is a singleton class that represents a previously observed object. While objects are predominantly incorporated with respect to existing classes, operators also exist for new node (class) creation, node combination (merging), and node division (splitting). A more complete description of COBWEB can be found in Fisher (1987).

Object incorporation is easily adapted to allow object classification and prediction: category utility guides an object along a path of nodes to a 'best' matching leaf. If any value(s) are missing from the new observation, they may be predicted from the known values of the leaf. While COBWEB trees are reminiscent of decision trees, probabilistic concepts are *polythetic* in that multiple attributes guide classification. If an object has missing attribute values then category utility acts as a partial-matching function with summation limited to probabilities of known attributes.

3.2 Hierarchy Evaluation: Pruning and Utility

COBWEB's strategy of prediction at best matching leaves demonstrably yields good results in many domains, but like early versions of ID3 this strategy can often diminish prediction accuracy (cf., Section 2). More generally, all inductive learning systems require that a domain exhibit regularities (i.e., dependencies between attributes) if learning is to be beneficial. If the learning system is too persistent in trying to uncover regularities where no significant ones exist, then this can result in 'overfitting'. This is also the case in learning to search: if no or little correlation exists between the conditions of a state and the eventual success of an operator application, then persistence in trying to discover such a connection will result in overfitting. In search control tasks, overfitting reduces the accuracy with which operator applications are predicted, thus causing greater backtracking. In both concept learning and search control learning, the utility of certain 'rules' is negligible or detrimental.

A recent version of COBWEB (Fisher, 1989) employs a *past-performance* method for disposing of low utility rules. This method was inspired by Quinlan's (1987) *reduced error* pruning, but the general approach is also related to Hansson and Mayer's and Wefald and Russell's strategies for terminating search.[2] In particular, as COBWEB classifies an object it determines at each node whether an attribute would be correctly predicted at the node. To do this, it compares the object's actual value along this attribute with the most common (i.e., most probable) value of the attribute at the node. If the two values are equal, then COBWEB would have correctly predicted the attribute's value if it had been required to; in this case, a counter is incremented indicating that a correct prediction would have been made at this node. In addition, COBWEB also records whether the attribute would have been correctly predicted at a descendant of the node. Thus, each node holds two counts for an attribute: one of how often a correct prediction would have been made at the node, and one of how often a correct prediction would have been made at a descendent of the node. When a prediction of an attribute is actually necessary (i.e., the attribute's value is unknown), then classification descends to a node at which the unknown attribute is more accurately predicted relative to its descendants; the most common value of the attribute is used as COBWEB's prediction.

Our summary of this past-performance 'pruning'

[2] Fisher (1989) also experimented with a chi-square method of terminating classification that directly assesses the significance of the information gained by deeper classification. Gennari, Langley, and Fisher (1989) used a cut-off parameter to prune a classification subtree based on a user-supplied threshold of required 'information gain'. This method was not sensitive to differences between attributes and relied entirely on the user to specify 'significant' gain thresholds.

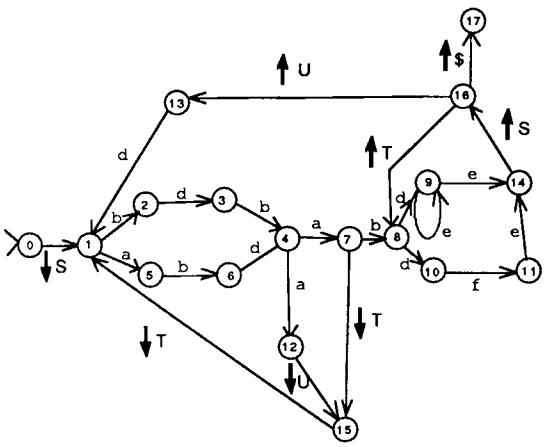

Figure 1: A NPDA transition diagram for simplified-English parsing.

strategy has been brief of necessity, but it does not add to the asymptotic cost of learning or classification and it considerably improves prediction accuracy over the initial strategy of always classifying an object to a leaf. With this in mind, we turn to the application of these concept formation strategies to our primary goal: the effective management of search control.

4 An Example: Search Control in Parsing

Our objective is to demonstrate that COBWEB (and by implication other conceptual clustering systems as well) can effectively direct search by predicting operators that are best applied under 'current circumstances'. Consider a detailed parsing example similar to the one given at the beginning of Section 3. Figure 1 shows a nondeterministic (i.e., backtracking) push-down automata (NPDA) for recognizing an artificial, but nontrivial language. Arcs have two types of labels: those preceded by an up/down arrow, and those that are a lower case letter. The lower case letters denote input symbols that are to be parsed. A down arrow, (↓), followed by a letter denotes a symbol to be pushed on a stack when the arc is crossed in parsing. An arc labeled by an up arrow, (↑), denotes that the symbol must reside at the top of the stack, and is popped from the stack. It is assumed that the stack contains only a $ when parsing begins. A sentence is accepted (i.e., successfully parsed) if there is at least one way to enter the *final state* of the machine (i.e., state 17) with an empty stack and an empty input stream.

Consider "a b d a b b d b a b d e e e e d e", which is a string accepted by the NPDA. A successful parse begins by pushing **S** on the stack in crossing from state 0 to state 1, parsing "a b d a b" leaving the system in state 7. At state 7, a **T** is pushed on the stack, then an **S** leaving the system in state 1. Next the symbols, "b d b a b d e e e e" are parsed, and then the **S** and **T** are popped off the stack. Next "d e" are parsed and the final **S** and $ are popped off the stack, leaving the system in state 17, the final state.

Our example illustrates the moves necessary for a successful parse, but the NPDA contains 4 points of nondeterminism: state 4 given an 'a', state 7 given a 'b', state 8 given a 'd', and state 9 given an 'e'. The arcs out of state 9 model precisely the dilemma of the noun/adjective example at the beginning of Section 3. This nondeterminism is the cause of search: each point of nondeterminism must be tried and may result in backtracking if the wrong choice is made. The NPDA is constructed so that certain incorrect guesses are discovered rather quickly, while others are not uncovered for several moves.

To reduce backtracking we present information about successful parses to COBWEB, in the hope that the system can use this information to guide future parsing. In particular, after a sentence is successfully parsed, a complete trace of the choices *required* from the start state through the final state is returned; this does not include the choices that were retracted via backtracking. Each choice is regarded as an operator to be predicted from decisions that were made previously. Thus, the complete trace is segmented into 'objects' (i.e., windows) of seven attributes: one object for each choice made in the trace. The values of four of these attributes are the four choices (operators) that were made just prior to the current choice; two of these attributes are the top two stack symbols at the time of the current choice; finally, the seventh attribute is the current choice. Thus, a single parse of ten choices will be segmented into ten distinct objects.[3] After sentence recognition, the successful parse is segmented and the constituent objects are incorporated into a COBWEB hierarchy. During subsequent parsing COBWEB is used as an oracle for predicting the current choice, given a window of four previous choices and the top-most stack symbols.

To test the savings provided by a COBWEB oracle, a COBWEB-enhanced parser was trained on successful parses. Training sentences were generated so as to usually adhere to two rules, although neither was followed 100% of the time, thus introducing some 'noise'. The first rule was that pushing a **T** or a **U** (leading into state 15) was dependent upon the path from state 1 to state 4. The second regularity was that roughly 4 consecutive *e*'s should occur when in state 9. The COBWEB-enhanced parser was trained on five randomly generated sentences (under the above men-

[3] Note that the initial choices – those without four predecessors – are still represented but without some initial attributes.

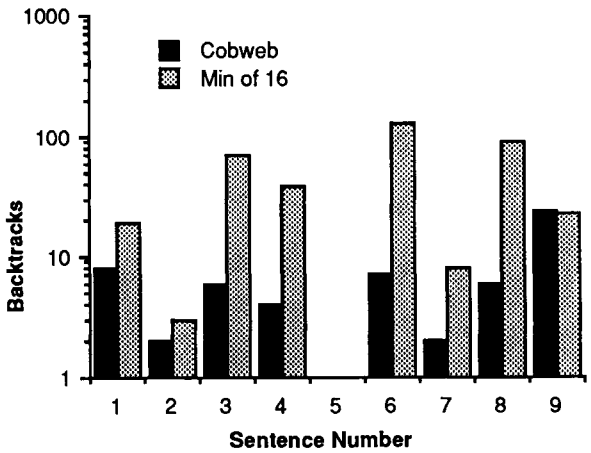

Figure 2: Backtracks required by COBWEB-guided parser and best of 16 alternative parsers.

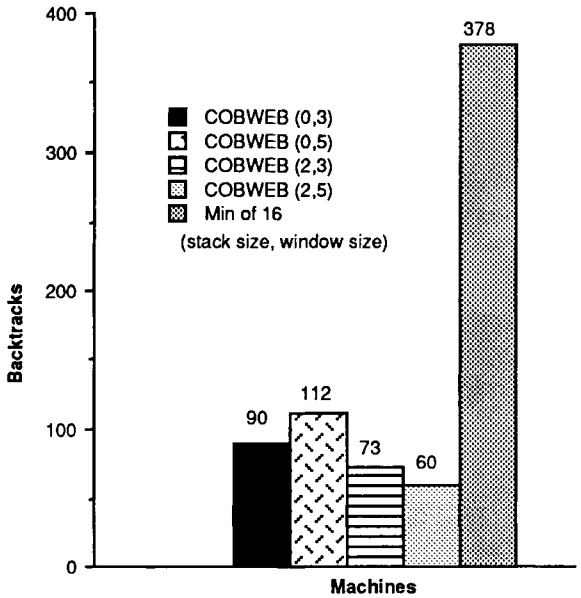

Figure 3: Sensitivity to input window and stack symbols.

tioned regularity constraints), which after windowing constituted a total of one to two hundred training objects. Four more sentences were randomly generated to use as additional test sentences.[4] For comparative purposes, the number of backtracks required by a COBWEB-enhanced parser was compared with sixteen *hard-coded* parsers: recall that the NPDA of Figure 1 has four points of nondeterminism – two choices per point. The sixteen hard-coded parsers correspond to the (2^4 =) 16 possible orderings on these choices. These orderings roughly correspond to all the possible ways that an 'expert' might order choice preferences with sufficient knowledge of individual choice frequencies. All nine (training and test) sentences were run against the 16 hard-coded machines; COBWEB was trained on the first 5 of these sentences and like the hard-coded machines was tested against all nine.

Figure 2 compares the number of backtracks required by the COBWEB-enhanced parser and the number required by the *best* hard-coded machine (per sentence) for each of the nine sentences (note the logarithmic vertical scale). The COBWEB-enhanced parser out performs the minimums of the hard-coded machines on all sentences, except one (sentence # 5) where the number of backtracks was one in each case. A nonparametric sign test reveals that the COBWEB-enhanced parser properly minimized backtracks over the collective minimum of the hard-coded machines in a statistically-significant (α = 0.01) number of cases (i.e., 8, with 1 tie); even if we could determine *a priori* the best hardcoded machine to parse each sentence, the enhanced parser would still win in terms of required backtracks. Overall, the COBWEB-enhanced parser requires 60 backtracks for all nine sentences, while the sum of the hard-coded machine minimums comes to 378 backtracks. If we compare the COBWEB-enhanced parser to the individual machines then the savings become even more pronounced: the backtracks required by the individual machines ranged from 643 to 2626, as compared to 378 for their collective minimum and 60 for our COBWEB parser.[5]

We also investigated the sensitivity of the COBWEB-enhanced parser to window size and number of stack symbols that were used during learning. In the results that we report above, we assumed a input window size of 5 and access to the top 2 stack symbols.[6] Figure 3 compares the total number of backtracks over all sentences for various window/stack sizes and the sum total of the minimum parse of each sentence for any of the static machines (Min of 16). For each of

[4] Originally, five additional test sentences were generated, but one was identical to a training sentence and was removed from consideration.

[5] There are several interesting issues that arise when we consider using concept formation techniques to guide language parsing. In particular, the relative performance merits of and conceptual links between our heuristically-guided parser and efficient parsers for such things as LR(k) grammars are of some interest, but are tangential to the objectives of this paper. Notably, our heuristic parser assumes that statistical correlations exist between parsing transitions; if such correlations exist then concept induction techniques can hope to provide some speedup over standard parsing methods, regardless of the language class. If no such correlations exist, then inductive techniques will provide no speedups over the standard parsing methods for a language. More generally, the objectives of this paper are to illustrate links between inductive concept formation and search control management, and thus we will defer discussion of parsing-specific issues.

[6] As we stated earlier, during testing, 4 of the 5 window symbols will correspond to the 4 previous choices, while the 5th will correspond to the choice to be predicted.

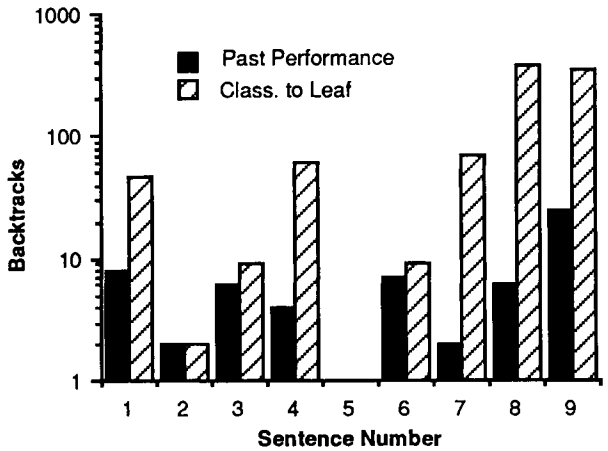

Figure 4: Backtracks required with and without noise-tolerant classification strategy.

the window/stack combinations that we investigated, the number of backtracks required by the COBWEB parser is considerably less than the hardcoded minimums. A second observation is that the performance of our parser appears to improve as the amount of information (input and stack symbols) available to guide classification increases. The exception to this trend occurs between (0,3) and (0,5), but we believe this to be an aberration caused by an exceptional sentence. In fact, the (0,5) properly minimized the backtracks relative to the (0,3) condition for 6 of the 9 sentences, with 1 tie and 2 cases in which (0,3) minimized backtracks. Though our experiments are not sufficient to make statistically-justified claims for the relative advantage of differing (stack, window) conditions at the $\alpha = 0.05$ level, we nonetheless believe that more extensive experiments will confirm the trend within certain limits.

Finally, the COBWEB results reported above assume the classification strategy designed to avoid overfitting that was described Section 3. To test our contention that this strategy avoids overfitting and the detrimental use of low-utility control knowledge, we compared these results to a parser that made predictions suggested by best-matching leaves of the learned concept tree. Figure 4 compares these alternative versions; classification to a leaf yields less reliable results and is often much more costly in terms required backtracking. The past-performance strategy outperforms classification to a leaf in 7 of the 9 cases, with 2 ties. This is significant using the nonparametric sign test at $\alpha = 0.05$. This supports our specific claim that our past-performance strategy gives a good measure of rule utility. More generally, a promising conjecture is that strategies designed to enhance noise tolerance in concept learning may be useful in mitigating utility problems in search control.

5 Concluding Remarks

We have demonstrated the efficacy of concept formation in the management of search control knowledge. COBWEB's probabilistic representation of search control facilitates greater flexibility in guiding search: in addition to hard-and-fast rules, tendencies ('hunches') also guide search. This characteristic is particularly important in tasks like parsing since it may be impossible to tell with certainty whether a particular operator is applicable prior to its invocation.

Although our approach was tested on a parsing task we believe that it can be adapted to other search-intensive tasks, though this may require that we overcome representational limitations. In particular, COBWEB requires nominal attribute-value representations. This representation is sufficient to deal with certain other domains that have been examined in search control research such as the 8-tile puzzle and other games (Wefald & Russell, 1989), but more complicated search tasks such as planning and those found in EBL research will require relational representations. In fact, extensions of COBWEB-like strategies that deal with relational representations are being investigated (Yoo & Fisher, 1990; Yang, Fisher, & Franke, in press).

A second, but more subtle, representation issue arises when we examine more deeply the meaning of rule 'utility'. In particular, we have ignored the 'match-cost' aspect of utility (Tambe & Rosenbloom, 1989); the COBWEB-enhanced parser uniformly requires greater execution time because of increased match cost. In part, this is due to 'uninteresting' factors such as (1) inefficiencies of the COBWEB implementation, which we have not tailored to this domain, and (2) the exceedingly low cost of backtracking with an NPDA. More fundamentally though, it appears that match costs are magnified by probabilistic concepts, which require that we match many attributes of a concept – even those that may be statistically irrelevant to class membership. Fortunately, the same noise-tolerance strategies that identify when an attribute is best predicted are also useful in determining when attributes are irrelevant for classification purposes. Thus, we believe that these strategies suggest a promising path for mitigating the match-cost factor of probabilistic-rule utility (Gennari, 1989).

Finally, we believe that a notable contribution of this work is that it illustrates a link between concept induction strategies for noise tolerance and search control issues of utility: classification/search should terminate at points that do not helpfully inform prediction. In Section 2 we alluded to a point that we more forcefully argue elsewhere (Fisher & Chan, in press; Yoo & Fisher, 1990) – that overfitting is also at the root of the utility problem as it pertains to EBL and domain theory search. Learned rules may represent logically possible, though statistically idiosyncratic connections between patterns of

operational predicates and target consequences. The application/consideration of such rules may actually have a detrimental effect on subsequent problem solving (Mooney, 1989; Markovitch & Scott, 1989). Our current work seeks to unify inductive issues of noise and explanation-based issues of utility in the context of concept formation over problem-solving experience (Yoo & Fisher, 1990) and plans (Yang, Fisher, & Franke, in press).[7] Our work tentatively suggests that EBL mechanisms of *selective retention* (Markovitch & Scott, 1989; Minton, 1988) and *selective utilization* (Markovitch & Scott, 1989; Mooney, 1989)[8] are synonymous in spirit to, and better implemented by, much studied inductive methods of *pruning* (Quinlan, 1986; Michalski, 1987) and preference identification (Fisher, 1989), respectively.

Acknowledgements

We thank the reviewers for helpful suggestions on content and exposition.

References

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

Etzioni, O. (1988). Hypothesis filtering: a practical approach to reliable learning. *Proceedings of the Fifth International Conference Machine Learning.* (416–428). Ann Arbor, MI: Morgan Kaufmann.

Fisher, D. H. (1987). Knowledge acquisition via incremental conceptual clustering. *Machine Learning, 2*, 139–172.

Fisher, D. H. (1989). Noise-tolerant conceptual clustering. *Proceedings of the International Joint Conference Artificial Intelligence* (pp. 825–830). Detroit, MI: Morgan Kaufmann.

Fisher, D., & Chan, P. (in press). Statistical guidance in symbolic learning. *Annals of Mathematics and Artificial Intelligence.*

Gennari, J (1989). Focused concept formation. *Proceedings of the Sixth International Workshop on Machine Learning* (pp. 379–382). Ithaca, NY: Morgan Kaufmann.

Gennari, J., Langley, P., & Fisher, D. (1989). Models of concept formation. *Artificial Intelligence, 40*, 11–62.

Gluck, M. A., & Corter, J. E. (1985). Information, uncertainty, and the utility of categories. *Proceedings of the Seventh Annual Conference of the Cognitive Science Society* (pp. 283–287). Irvine, CA: Lawrence Erlbaum.

[7] Unification appears to be a similar, though implicit, intention behind the work of Etzioni (1988).

[8] We believe that these terms were coined by Markovitch and Scott.

Hansson, O., & Mayer, A. (in press). Probabilistic heuristic estimates. *Annals of Mathematics and Artificial Intelligence.*

Langley, P. (1985). Learning to search: from weak methods to domain-specific heuristics. *Cognitive Science, 9*, 217–260.

Markovitch, S. & Scott, P. (1989). Information filters and their implementation in the syllog system. *Proceedings of the International Workshop on Machine Learning* (404–407). Ithaca, NY: Morgan Kaufmann.

Michalski, R. (1987). How to learn imprecise concepts: a method for employing a two-tiered knowledge representation in learning. *Proceedings of the Fourth International Workshop on Machine Learning* (50–58). Irvine, CA: Morgan Kaufmann.

Minton, S. (1988). Qualitative results concerning the utility of explanation-based learning. *Proceedings of the Seventh National Conference on Artificial Intelligence* (pp. 564–569). St. Paul, MN: Morgan Kaufmann.

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

Mitchell, T., Utgoff, P., & Banerji, R. (1983). Learning problem solving heuristics by experimentation. In R. Michalski, J. Carbonell, & T. Mitchell (Eds.) *Machine Learning: An Artificial Intelligence Approach*, Palo Alto, CA: Morgan Kaufmann.

Quinlan, J. R. (1986). Induction of decision trees. *Machine Learning, 1*, 81–106.

Quinlan, J. R. (1987). Simplifying decision trees. *International Journal of Man-machine Studies.*

Rendell, L., Seshu, R., & Tcheng, D. (1987). More robust concept learning using dynamically-variable bias. *Proceedings of the Fourth International Workshop on Machine Learning* (pp. 66–78). Irvine, CA: Morgan Kaufmann.

Smith, E. E., & Medin, D. L. (1981). *Categories and concepts.* Cambridge, MA: Harvard University Press.

Tambe, M. & Rosenbloom, P. (1989). Eliminating expensive chunks by restricting expressiveness. *Proceedings of the International Joint Conference Artificial Intelligence* (pp. 731–737). Detroit, MI: Morgan Kaufmann.

Wefald, E. & Russell, S. (1989). Adaptive learning of decision-theoretic search control knowledge. *Proceedings of the International Workshop on Machine Learning* (408–411). Ithaca, NY: Morgan Kaufmann.

Yang, H., Fisher, D., & Franke, H. (in press). Improving planning efficiency by conceptual clustering. *The Third International Conference on Industrial & Engineering Applications of Artificial Intelligence & Expert Systems.* Charleston, SC: ACM Press.

Yoo, J., & Fisher, D. (1990). Concept formation over explanations and problem-solving experience. Technical Report. Department of Computer Science, Vanderbilt University.