LEARNING CONTROL INFORMATION IN RULE-BASED SYSTEMS: A WEAK METHOD

Usama M. Fayyad
Kristina E. Van Voorhis
Mark D.Wiesmeyer

Reprinted from PROCEEDINGS OF THE FOURTH CONFERENCE
ON ARTIFICIAL INTELLIGENCE APPLICATIONS,
San Diego, CA, March 14-18, 1988

The Computer Society of the IEEE
1730 Massachusetts Avenue NW
Washington, DC 20036-1903

Washington • Los Alamitos • Brussels

THE INSTITUTE OF ELECTRICAL AND ELECTRONICS ENGINEERS, INC.

IEEE COMPUTER SOCIETY PRESS

Learning Control Information In Rule-Based Systems: A Weak Method

Usama M. Fayyad[†]
Kristina E. Van Voorhis[‡]
Mark D. Wiesmeyer[††]

[†]Robot Systems Division, Department of E.E.C.S., The University of Michigan, Ann Arbor
[‡]Environmental Research Institute of Michigan, Ann Arbor
[††]Department of E.E.C.S., The University of Michigan, Ann Arbor

Abstract

Rule-based systems constitute one of the most successful classes of AI systems. This paper presents a weak method for learning control information in rule-based systems. The method proposes establishing "connections" between rules each time the system successfully performs a task. Connections have associated strengths which code the history of the success/failure of sequences of rule firings during problem solving. Connection strength is used to provide guidance to the inference engine in two important ways: 1. selection of the next rule to consider for matching, and 2. a basis for a conflict resolution scheme. The connection strength is also used in guiding the generation of new rules through composition. It insures that composition will only occur on sequences that have established their utility to the system through problem solving experience. We believe that the method proposed in this paper will, in general, result in significantly decreasing the matching and search efforts. The method has been implemented in a program called STAC and some favourable results have been obtained in the test domain of Euclidean geometry proofs. This paper presents the method used as well as the results obtained. At this stage, no formal analysis of the performance or the results is provided.

1 Introduction

One of the areas in which Artificial Intelligence has met with some success is that of rule-based systems. There are several advantages to such systems. They are easy to modify and they provide modularity in both task and control knowledge[Wat70]. Newell and Simon[New72] consider production systems to be a good candidate for modelling human cognitive processes. Neches, Langley, and Klahr[Nec87] point out further advantages and argue that such systems provide a suitable framework for modelling learning processes.

The representation of domain knowledge in the form of modular independent rules causes the system's efficiency to rely heavily on the control scheme. Control knowledge is particularly difficult to formulate. It is significantly more difficult than formulating the task/domain rules. Not only does it require knowledge of how a task can be executed, but also a thorough understanding of why a task is accomplishable, the properties of operators, and why one operator is favoured over another. In this paper, we propose a weak method for the acquisition of control knowledge in production systems.

1.1 Production Systems and The Control Problem

A production system consists of:

- A *production memory* (PM), which contains the rules of the system in condition/action format.
- A *working memory* (WM), which holds state information during problem solving. Conditions of rules are matched against WM contents. The action part of a rule modifies parts of WM.
- An *inference engine*, which performs the functions of matching rules against WM contents, deciding which rules are applicable, and finally selecting a single rule to apply. These three phases are described in more detail below.

Production systems are therefore characterized by the separation of rules, data, and the control schema[Wat78].

Basic operation of a production system follows a *recognize-act cycle*[McD78a] after WM has been initialized to a given state. Execution takes place in three phases[Nec87], the first corresponding to the *recognize* stage and the last two to the *act* stage. In the first, known as the *matching phase*, the system matches its rules against WM contents to determine the *conflict set*, the set of rules which are currently applicable. Next is the *conflict resolution phase* in which the system selects the rule to fire from the conflict set. The third phase is the rule application phase during which the selected rule is *fired* by executing its action part. The system is also provided with a set of *goal conditions*, which when satisfied indicate attainment of the goal. The recognize-act cycle repeats if the goal conditions are still not satisfied.

The control problem in production systems may be defined as:

Which of its applicable rules in a given situation should a production system select as the one to fire?

Several examples of general conflict resolution schemes appear in the literature.[1] The WM *recency* heuristic[Lan87,McD78b] favours rule instantiations that use elements most recently added to WM. *Distinctiveness rules* favour rules on the basis of their similarity or dissimilarity to previously executed instantiations of rules in the current conflict set[McD78b]. Rule *specificity* is also used in conflict resolution to favour more specific rules[McD78b,Hol86]. Some schemes impose an explicit priority ordering on the rules in PM[McD78b,Bar81]. Such ordering is determined by the programmer's *a priori* knowledge of rule interactions during task execution. In most systems that use it, the rule ordering is fixed and does not change during run time.

Another factor affecting the efficiency of a production system is the matching phase. The matching operation can be time consuming, as it may involve instantiating many variables. Special complex matching techniques can significantly improve the over-

[1] We assume *refractoriness*[McD78b,Nec87] is the first step of conflict resolution and is necessary for proper execution. For this reason it is not discussed as an option. Refractoriness means the prevention of a rule from firing more than once when matched to the *same* set of WM elements. This prevents "trivial looping"[Lan87].

CH2552-8/88/0000/0188$01.00 © 1988 IEEE

all efficiency of the system[Wat78]. Sophisticated matching techniques include the use of hierarchies, state transitions, meta-rules, or complex control schemes like Petri nets to narrow the set of rules considered[Wat78]. Some systems use special filters to eliminate many rules from consideration[McD78b]. For OPS5[For81], Forgy employed a discrimination network that minimizes matching time for computing the entire conflict set at the cost of space complexity. The disadvantage of such schemes is that they impose limitations on the form and the actions of rules, and they make the modification of production memory more problematic. However, flexibility is an aspect of particular importance to learning systems. For the proposed weak method, we assume a fairly simple matcher that matches rules individually. The control scheme provides guidance to the matcher in selecting the next rule to match thus avoiding the expensive computation of the entire conflict set.

1.2 The Proposed Approach

We propose the addition of an efficient learning mechanism to the inference engine. The goal is to improve the matcher's efficiency by reducing the amount of rules it has to consider during problem solving and to improve the quality of the problem solving by providing a good conflict resolution scheme. The fact that the general conflict resolution schemes mentioned in section 1.2 do not exploit domain specific knowledge often results in unsatisfactory performance[Nec81]. On the other hand, domain-dependent approaches often demand too much effort on the part of the system designer. Not only must the basic rules be provided, but control heuristics must be discovered and implemented—this may not always be easy to do. An alternative to explicitly coding control heuristics would be for the system to *discover* them during normal operation. The method we propose attempts to discover such heuristics in a manner which is efficient and general. While the information coded is domain/experience dependent, the learning mechanism has no ties to the domain itself. This conforms with Langley and Simon's emphasis on the importance of finding general learning mechanisms through which domain-specific knowledge is acquired[Lan81].

In our method, the history of problem solving is coded in "connections" which are established between rules. A *connection strength* is associated with each connection. The connection strength from $Rule_A$ to $Rule_B$, is a measure of the success of firing $Rule_B$ after $Rule_A$ during problem solving. It is increased whenever the two appear in sequence on the solution path. It is decreased if $Rule_A$ is on the solution path and $Rule_B$ is not, but was fired *immediately* after $Rule_A$. A connection between two rules need only be established if they fired in sequence. Thus a high connection strength between $Rule_A$ and $Rule_B$ corresponds roughly to the following heuristic assertion:

> Whenever $Rule_A$ fires, then based on experience, the most likely candidate rule to be fired *and eventually lead to success* is $Rule_B$.

Connection strength is used as the basis for conflict resolution. After firing a rule, the matcher first considers rules connected to the fired rule in order of decreasing strengths. The first rule to match will be fired; thus, there is never a conflict set. Traditional matching methods compute the entire conflict set before invoking the conflict resolution algorithm. We advocate the use of conflict resolution to guide the matcher in order to avoid matching some rules that will not be applied. If our scheme provides "good" conflict resolution, then the matcher will avoid a lot of extra work resulting in increased overall efficiency.

In addition to providing a control learning scheme, "connections" are also used to determine when composition of rules is warranted. Only rules with strong connections between them are composed. A high connection strength between two rules is a strong indication of their being repeatedly used *successfully, and in sequence.* It is therefore beneficial to the system to compose them. This provides a degree of immunity to creating a large number of rules that have low utility and whose presence would only serve to degrade the matcher's performance.

Before giving an overview of the implementation of this approach, STAC[2], we provide some background on machine learning and present previous work on learning control in production systems. Later, we present the results obtained by studying STAC's performance in the test domain of Euclidean geometry proofs.

2 Background

One of the earliest attempts at learning control is Samuel's checker player program[Sam63]. Samuel's approach required the programmer to identify control domain attributes which would be used by the system in subsequent learning of control heuristics. Such control domain knowledge is in general very difficult to formulate.

Our notion of maintaining connections with varying strengths is an extension of the use of *rule strength* in previous systems. The strength of a rule is a measure of the success of a rule[Hol78,Hol86], or the frequency with which it is used[Lan87]. Holland[Hol78, Hol86] used rule strengths for both conflict resolution and as a metric for purging low utility rules from his system. Rules (classifiers) "bid" a portion of their strength away during conflict resolution. Successful rules are awarded strength either from the environment or from bids by subsequent rules. Langley[Lan87] has also used rule strengths as part of a multi-component conflict resolution scheme for PRISM. Anderson[And83,And83a] has used rule strengths and activation strengths in the nodes of a semantic net corresponding to WM elements for conflict resolution. The emphasis of his work, however, was on *proceduralization.* The information coded in our learning scheme is at a finer level of granularity than that of rule strengths since connection strengths code a measure of "relatedness" or association between pairs of rules.

Rule composition was originally proposed by Lewis[Lew78] to account for speedup as the result of practice. Korf introduced a weak method for learning control in a restricted class of domains[Kor85]. A macro-operator is created for every sequence of operators that the system uses in solving a problem. This leads to the creation of a large number of macros that are very specific to the state and goal or subgoal for which they were created. Iba[Iba86] has developed a more sophisticated composition algorithm which employs a structure proposer, a static filter for judging rules on the basis of their simplicity, power and applicability, and a dynamic filter for judging rules on the basis of the frequency of their applicability and success. Vere[Ver77,Ver78] has developed a system which learns rules from situation sequences and "before-and-after" pairs. Laird's[Lai84,Lai86] SOAR system, employs a weak learning method called Chunking. Chunking is a general learning technique similar to composition and is used to learn new rules as well as control knowledge. One problem that can affect SOAR is that many formed chunks may not be useful in the future. What makes STAC unique among all of the aforementioned systems is that rule composition is *connection-strength-based.* One advantage to this, is

[2]The name STAC is an acronym for Solve, Trace, And Connect — a basic outline of the operation of the system.

that composition occurs only among rules which have been proven, through experience, to lead to success when fired in sequence. Thus, STAC avoids some of the dangers that arise in single-trial learning systems.

Mitchell's [Mit83] LEX system learns heuristics for deciding the usefulness of an operator in a given situation. Domain knowledge in the form of a hierarchy of concepts is needed to guide generalization and specialization in the *version space*. Domain-specific heuristics coded in the form of *meta-rules* have also been used to perform conflict resolution[Wat70,Mos78,Sil86]. Control knowledge in all of the above systems takes the form of symbolic rules. Thus part of the domain performance effort — matching, rule application — is expended on computing control strategies. Hayes-Roth[Hay85] advocated such explicit control mechanisms and claimed that they are actually *necessary* for intelligent control. However, she admits that such an approach is very inefficient and is likely to be used only during development of the system and not during task execution. In contrast, control knowledge in STAC is encoded in terms of connection strengths. In addition to low overhead, the advantage to this is that new control knowledge does not interfere with the performance aspects of the system.

2.1 Motivation

Simon[Sim83] has provided the following functional definition of machine learning which effectively describes the intended behavior of STAC:

> *Learning denotes changes in [a] system that are adaptive in the sense that they enable the system to do the same task or tasks drawn from the same population more efficiently and more effectively the next time.*

We take the view that the goal of any kind of "useful" learning is to improve performance in a given domain. This suggests that a primary evaluation criterion for a learning system is its efficiency. The control information learned using our approach does not interfere with the operation of the system except in providing guidance. The space complexity is not high. In the worst case, a matrix of connections of size n^2 may have to be maintained, where n is the number of rules in PM. From our experience with the system in the domain of Euclidean geometry, we discovered that the matrix of connections is quite sparse. The results and improvement in problem solving performance are discussed in section 3. Our belief in the usefulness of this method for learning control has so far been based only on intuitive arguments and observations. One goal of our future research is to establish a formal framework for evaluating the usefulness of this method for arbitrary problems. As an initial intuitive guess for the type of characteristics possessed by a domain that make this approach useful we propose that:

1. Little knowledge of the domain, beyond the primitive operators, is available.
2. The solution of most tasks should take more than one step.
3. The system should have a relatively large number of rules— domains where very few general operators are enough to solve most problems are not well suited for this method.
4. Each rule codes a small amount of knowledge.

The last two conditions conform with Davis and King's characterization of domains for which production systems may be useful[Dav77]. As in all learning methods, the STAC architecture assumes that exploitation of regularity in the domain of application is possible. Absence of regularity in a domain is unusual and learning in such a domain would not be fruitful[Hol86]. One goal of our future research is the determination of the degree of regularity required in a domain for this method to yield useful learning.

2.2 Why Euclidean Geometry?

> *Nothing but Geometry can furnish a thread for the labyrinth of the composition of the continuum...and no one will arrive at a truly solid metaphysic who has not passed through that labyrinth [Rus37].*
>
> *G. W. Leibniz (1646-1716).*

Geometry is a domain which possesses all of the properties stated in the preceding section. It contains many rules which may naturally be coded to contain very little knowledge. In addition, most geometry proofs take more than one step. The problems used in testing STAC's performance were taken from Anderson's work [And83,And83a] (problems labelled P1-P7, L1, and L2). Another set of problems was taken from a high school geometry text[Jur65] (problems labelled 198, 201, 209, 200-1,-2,-5,-6,-7,-8). The problems selected were mainly about triangle congruence. The reason for restricting testing to problems of one type was to test conflict resolution in isolation. If the problems were of differing types (squares, circles, rectangles,...) then most rule selection would be done by the *matcher*. By restricting them to triangle congruences, most rules would have their conditions satisfied in most situations and conflict resolution would dominate in its effect on performance.

3 STAC: Description and Test Results

3.1 The Implementation

STAC is a production system augmented with a learning system. The latter consists of three distinct modules: the tracer, connection adjuster/builder, and the composer. Figure 1 gives an overview of the STAC architecture. The problem solver is a basic forward searching production system. The matcher selects rules for matching in order of their connection strengths to the last rule fired. The first rule to match is fired since it will, by definition, be the most favoured rule according to the conflict resolution scheme.

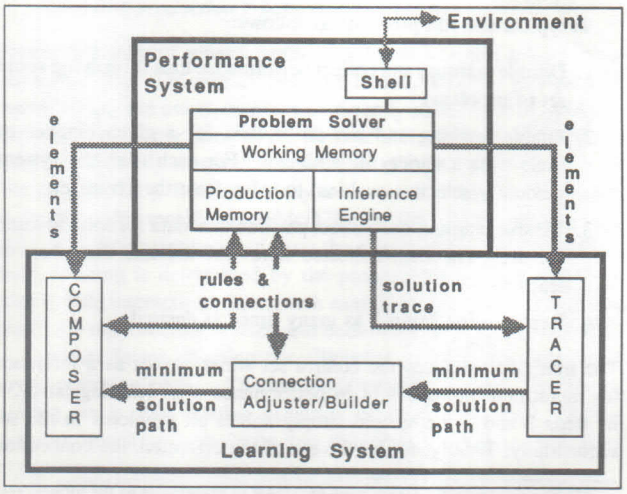


Figure 1. The STAC Architecture

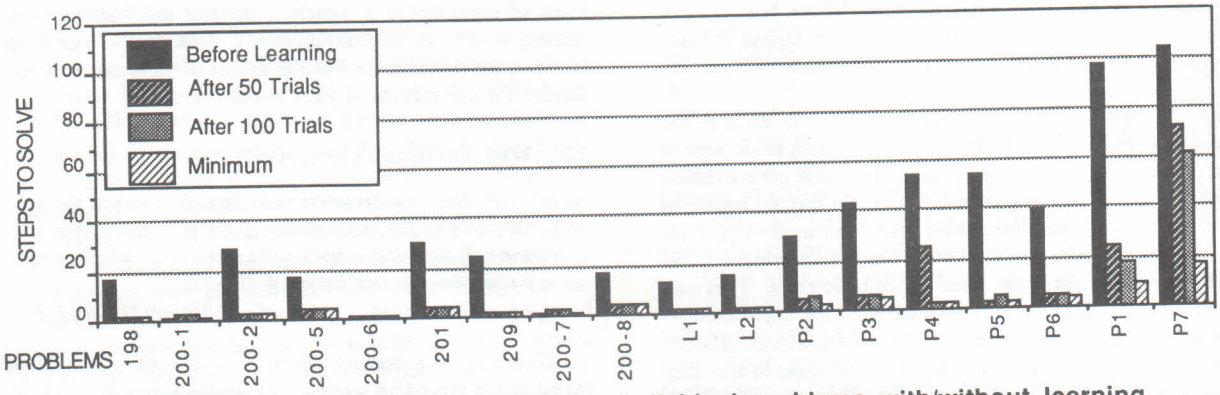


Figure 2. Number of steps to solve individual problems with/without learning.

Once the problem solver reaches a solution, it invokes the tracer module whose function is to determine the *minimal solution path* — the minimum length subsequence of the obtained solution sequence necessary to derive the goal.[3] This is achieved by traversing the solution sequence in reverse, marking only those rules that contributed to the goal. The process involves some search, since at each step several instantiations of a rule or some productions may have been applied that are irrelevant to achieving the goal.

The connection adjuster/builder simply increases the connection strengths between all the rules on the minimal solution path. If a connection does not exist, it is created. This module is also responsible for weakening connections. A connection between a rule on the minimum path and another that caused search to deviate from that path is weakened. Thus only rules that are in direct contact with rules on the minimal solution path are affected.

Rule composition creates macro-productions which code several problem solving steps into one. Two rules on the solution path are composed if the connection strength between them exceeds some user-defined threshold. Once the conditions and actions of the new rule are determined, it is variablized. Next, the newly composed rule is placed in PM, at which time it inherits the incoming and outgoing connections of its parent and child constituent rules respectively. The newly composed rule is given priority[4] over its parent since it is both more powerful and more specific[Wat70].

3.2 The General Testing Method

A typical test run proceeds as follows:

1. Disable learning and collect performance data for solving some set of problems.
2. Enable learning and run the system for a given number of trials over a variety of problems. For each trial, the system randomly selects a problem to solve from the given set.
3. Disable learning and collect performance data for solving same the set. The data collected here will indicate the effect of learning.
4. Repeat steps 2 and 3 as many times as desired.

The first step produces the control set which serves as a reference for measuring improvement in performance after learning (step 3). In steps 1 and 3 the system simply solves all problems in the set sequentially. Finally, for all the test runs performed, the connection strengthening factor was twice the weakening factor. We have not yet investigated the effects of varying these factors on learning. The argument for setting the weakening factor lower than the strenghtening factor is intuitive: usually the system has good reason for strengthening a given connection—it just solved a problem. However, stronger evidence is needed for weakening a connection since this is usually not due to the fact that a connection is "bad", but rather that interference is occuring between two "good" connections established for two different problems.

Upon solving a given problem, STAC provides reports on its performance via the following measures:

1. *Steps to Solve* : The actual number of steps needed to solve a problem (determined by the problem solver).
2. *Minimum Steps* : The number of steps in the minimum solution path (determined by the tracer).
3. *Rules Matched* : The number of rules matched during problem solving.
4. *Rules Considered* : The number of rules considered by the matcher.

Note that Steps to Solve $\leq$ Rules Matched $\leq$ Rules Considered.

3.2.1 Effect of Learning Control on Performance

Figure 2 shows the performance data for individual problems. Note that for most problems learning had a positive effect on performance — some of them were solved with minimal effort. Consider, as a more comprehensive performance measure, the total effort expended for solving the entire set of problems. Figure 3 shows the total effort needed in each of the above four categories to solve the

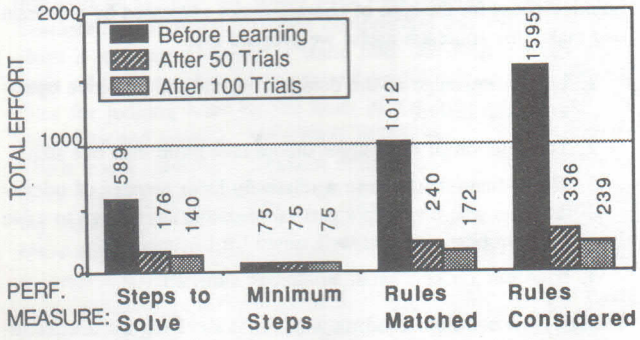


Figure 3. Performance data for solving entire set of problems with/without learning.

[3]This is similar to the search done by SOAR in preparation of the formation of a new chunk[Lai84,Lai86].

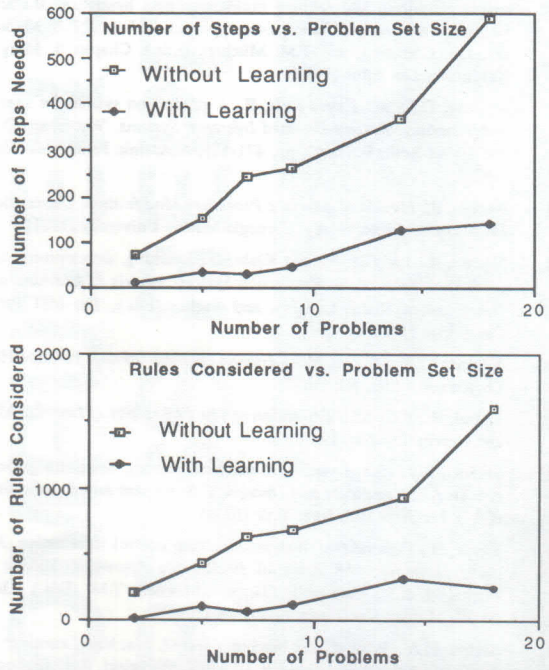
[4]i.e. the order by which the matcher considers rules having equal connection strength

entire set of problems.[5] Note that effort decreases monotonically with learning. The decrease is quite significant. After learning, the number of steps needed to solve the entire set decreased by 77%, the number of rules matched showed an 83% reduction, and the number of rules considered for matching was reduced by 85%. After learning, 50% of the problems were actually solved with minimal effort while most others exhibited very significant effort reduction.

3.2.2 Effect of Problem Set Size on Learning

Does the size of the problem set being used affect the learning method? STAC was run on several problem sets having different sizes.[6] On each problem set, the system was reset to an initial state, and then trained on that set. After training, learning was disabled, and the system was made to solve all problems in the set. The total number of steps needed to solve all the problems and the total number of rules considered during the problem solving were recorded for each set.

The graphs of these data versus size of the problem set (Figures 4 & 5) indicate that learning is fairly immune to the size of the training set. The corresponding data without any learning are shown on each graph to provide comparison. The performance data with learning are also significantly better than without.



Figures 4 & 5. Effect of problem set size on learning

3.2.3 Testing Knowledge Transfer

If trained on one set of problems then tested on another, would STAC perform on the second set better than it would have without any learning? STAC was run on a problem set that does not contain P1 and P7—the two most difficult problems. Without learning, the system required 72 steps to solve P7. After training on the easier subset, it solved P7 in 44 steps (the minimal solution path for P7 is only 8 steps). Thus it did improve, but was still far from reaching the minimal solution. Similar behavior occured with problem P1.

[5] Simply take the sum of performance data in each category over all the problems in the set.

[6] These were monotonically increasing subsets of the probelms.

In general, we would expect to see some knowledge transfer due to the fact that learning causes the system to discover connections between generally related steps. For example, after proving that two triangles are congruent, the most reasonable subsequent step is to apply the rules that derive the consequences of congruence—equality of all sides and angles—as opposed to deriving other facts. Indeed, this kind of behavior was exhibited by STAC. Some other tests supported this conclusion. However, it is very difficult to judge this particular aspect of learning since the system operates in a depth-first search strategy and is thus liable to actually perform worse after learning on a different set of problems. Without further formal analysis, or the collection of large amounts of data, no judgements regarding this issue would be justifiable.

3.2.4 Testing Learning with Composition

Recall that connection strength is the basis for deciding when to compose. Thresholds for composition were set very low and very high, providing no selectivity and high selectivity for rule composition, respectively. Learning trials on randomly chosen problems from the set were performed for 20, 60, and 100 trials. The figures showing results of testing learning with composition are not included due to space limitations. However, the results are summarized below.

The total effort required to solve the problem set was measured and as expected, performance did improve with learning. It was observed that the number of rules *considered* increased monotonically over successive learning sessions for both thresholds. This is due to the fact that there were simply more rules to be sifted through by the matcher. Beyond 20 trials, the performance often degraded. This may have been due to the fact that learning in the 20 trial session worked well, but that in later sessions many rules were created through composition and the new rules were not as useful. This conclusion is supported by the monotonically increasing number of rules considered. Space and time complexity is affected by the level of the composition threshold. Being more selective lowers the memory cost and the matching time. Being less selective results in more composed rules and consequently shorter solution paths. The price to pay in the latter case is in terms of increased memory and matching time.

4 Discussion and Future Work

We feel that the initial test results warrant further investigation of this approach. The approach constitutes a weak method and as such it has many disadvantages. The major disadvantage is the lack of *goal-directedness*. The control information learned is not sensitive to the goal. Thus, changing the goal for a problem that the system was trained on would cause the control scheme to lead it astray until recovery through repeated practice takes place. Another problem is *interference* due to rules that are used in different problems. The connections do not code problem dependent information, thus rules common to different problems cause the system to "lose track" of which problem it is solving. This effect is emphasized further when the problems are similar thus eliminating any "filtering" by the matcher. Despite these problems, our experience with the system shows that improvements over conflict resolution schemes that select randomly from the conflict set are quite significant. We propose the following goals for further research on this method:

1. Provide extensions to this method that address the issues of interference and lack of goal-directedness discussed above. Emphasis on efficiency and low overhead should be carried

through in the development of these remedies.

2. Develop a formal model and general evaluation criteria for analytically evaluating the expected usefulness of the method for arbitrary domains.
3. Study the effect of varying the learning parameters used in increasing and decreasing connection strength and the connection strength threshold for rule composition. This includes the investigation of possible domain characteristics that may help in determining the proper values for these parameters.
4. Demonstrate the usefulness of this method in different application domains.

5 Acknowledgements

The first author gratefully acknowledges the Robotics Research Lab of The University of Michigan and the U.S. Airforce Office of Scientific Research for support under contract F49620-82-C0089.

The authors would like to extend their thanks to Prof. Terry Weymouth and to the Computer Vision Research Lab of the University of Michigan for granting us access to TI Explorer and Symbolics machines on which testing of the system took place.

Over several meetings, Prof. Keki B. Irani provided much help with some of the issues that arose during the design stage. He pointed out and clarified many of the problems with the approach. Conversations with Prof. John Laird regarding issues of production system implementaton and design decisions in SOAR were extremely helpful. We are also grateful for his careful reading of the draft. Prof. Brian Schunck gave encouraging remarks on an early draft of the paper.

Dr. U. Mukhopadyay and Dr. R. Uthurusamy at General Motors Research Labs gave helpful comments and insightful discussions on earlier drafts of this paper. Thanks to anonymous CAIA-88 reviewer R11 for a thorough commentary on the submitted draft.

References

[And83] Anderson J.R., *The Architecture of Cognition*, Harvard University Press, Cambridge, Ma. (1983).

[And83a] Anderson, J.R. Acquisition of proof skills in geometry, *Machine Learning: An Artificial Intelligence Approach, Volume I*, Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. (Eds.), Morgan Kaufmann, Los Altos, Chapter 7 (1983).

[Bar81] Barr, A., and Feigenbaum, E. (Eds.), *The Handbook of Artificial Intelligence*, Vol. I, p. 60, William Kaufmann, Los Altos, CA (1981).

[Dav77] Davis, R. and King, J., An overview of production systems, in *Machine Intelligence 8*, E. Elcock and D. Michie (Eds.), pp. 300-332, Ellis Horwood, Chichester, England (1977).

[For81] Forgy, C.L., Ops5 User's Manual, *Technical Report CMU-CS-81-135*, Department of Computer Science, Carnegie-Mellon University, Pittsburgh, PA (1981)

[Hay85] Hayes-Roth, B., A blackboard architecture for control, *Artificial Intelligence 26*, pp. 251-321, Elsevier Science Publishers, B.V., North-Holland (1985).

[Hol78] Holland J. H., and Reitman J.S., Cognitive systems based on adaptive algorithms, *Pattern-Directed Inference Systems*, Waterman, D.A. and Hayes-Roth, F. (Eds.), pp. 313-329, Academic Press, New York, (1978).

[Hol86] Holland J.H., Escaping Brittleness: The possibilities of general-purpose learning algorithms applied to parallel rule-based systems, *Machine Learning: An Artificial Intelligence Approach, Volume II*, Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. (Eds.), Chapter 22, Morgan Kaufmann, Los Altos, (1986).

[Iba86] Iba, G.A., Learning by composition, *Machine Learning, A Guide to Current Research* Mitchell, T.M., Carbonell, J.G., and Michalski R.S. (Eds.), pp. 115-117, Kluwer Academic Publishers, Hingham, MA (1986).

[Jur65] Jurgensen, Donnelly, and Dolciani, *Modern Geometry: Structure and Method*, pp. 198-209, Houghton Mifflin, Boston (1965).

[Kor85] Korf, R.E., Macro-operators: A weak method for learning, *Artificial Intelligence* 26:35-77, (1985).

[Lai84] Laird, J.E., Rosenbloom, P.S., and Newell A., Towards chunking as a general learning mechanism, *Proceedings of The National Conference on Artificial Intelligence*, pp. 188-192, Morgan Kaufmann, Austin, TX (1984).

[Lai86] Laird, J.E., Rosenbloom, P.S., and Newell, A., Chunking in Soar: the anatomy of a general learning mechanism, *Machine Learning* 1:11 (1986).

[Lan81] Langley, P. and Simon, H., The central role of learning in cognition, in *Cognitive Skills and Their Acquisition*, J.R. Anderson (Ed.), Lawrence Erlbaum, Hillsdale, NJ (1981).

[Lan87] Langley, P., A general theory of discrimination learning, in *Production Systems Models of Learning and Development*, Klohr, D., Langeley, P., and Neeches, R. (Eds.), MIT Press, Cambridge (1987).

[Lew78] Lewis, C., *Production System Models of Practice Effects*, Dissertation, University of Michigan, Ann Arbor (1978).

[McD78a] McDermott, J., Newell, A., and Moore, J., The efficiency of certain production system implementations, *Pattern-Directed Inference Systems*, Waterman, D.A. and Hayes-Roth, F. (Eds.), pp. 155-176, Academic Press, New York, (1978).

[McD78b] McDermott, J., and Forgy, C., Production system conflict resolution strategies, *Pattern-Directed Inference Systems*, Waterman, D.A. and Hayes-Roth, F. (Eds.), pp. 177-199, Academic Press, New York, (1978).

[Mit83] Mitchell, T.M., Utgoff, P.E., Banerji, R., Learning by experimentation: acquiring and refining problem-solving heuristics, *Machine Learning: An Artificial Intelligence Approach, Volume I*, R.S. Michalski, J.G. Carbonell, and T.M. Mitchell, (Eds.), Chapter 6, Morgan Kaufmann, Los Altos (1983).

[Mos78] Mostow, D.J., and Hayes-Roth, F., A production system for speech understanding. *Pattern-Directed Inference Systems*, Waterman, D.A. and Hayes-Roth, F. (Eds.), pp. 471-481, Academic Press, New York (1978).

[Nec81] Neches, R., *Models of Heuristic Procedure Modification*, Dissertation, Department of Psychology, Carnegie-Mellon University (1981).

[Nec87] Neches, R., Langley, P., and Klahr, D., Learning, development, and production systems, in *Production Systems Models of Learning and Development*, Klahr, Langley, and Neches (Eds.), The MIT Press, Cambridge (1987).

[New72] Newell, A., and Simon, H.A., *Human Problem Solving*, Prentice-Hall, Englewood Cliffs, NJ. (1972).

[Rus37] Russel, B., *A Critical Exposition of The Philosophy of Leibniz*, Allen and Unwin, London, 1937.

[Sam63] Samuel, A.L., Some studies in machine learning using the game of checkers, in *Computers and Thought*, E.A. Feigenbaum & J. Feldman (Eds.), McGraw-Hill, New York (1963)

[Sil86] Silver, B., Precondition analysis: learning control information, *Machine Learning: An Artificial Intelligence Approach, Volume II*, Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. (Eds.), Chapter 20, Morgan Kaufmann, Los Altos (1986).

[Sim83] Simon, H.A., Why should machines learn?, *Machine Learning: An Artificial Intelligence Approach, Volume I*, Michalski, R.S., Carbonell, J.G., and Mitchell, T.M. (Eds.), Chapter 2, Morgan Kaufmann, Los Altos (1983).

[Ver77] Vere, S.A., Relational production systems. *Artificial Intelligence* 8:47-68 (1977).

[Ver78] Vere S. A. 1978 Inductive learning of relational productions, *Pattern-Directed Inference Systems*, Waterman, D.A. and Hayes-Roth, F. (Eds.), pp. 281-295, Academic Press, New York, (1978).

[Wat70] Waterman, D. A., Generalization learning techniques for the learning of heuristics, *Artificial Intelligence*, 1:121-170 (1970).

[Wat78] Waterman, D.A. and Hayes-Roth, F., An overview of pattern-directed inference systems, *Pattern-Directed Inference Systems*, Waterman, D.A. and Hayes-Roth, F. (Eds.), pp. 3-22, Academic Press, New York, (1978).