The Match Cost of Adding a New Rule: A Clash of Views

Milind Tambe, Robert Doorenbos, Allen Newell

June 1992
CMU-CS-92-158

School of Computer Science
Carnegie Mellon University
Pittsburgh, PA 15213-3890

Abstract

What is the match cost of adding a new rule to a production system (rule-based system)? Two conflicting views have emerged. Research in EBL indicates that learned rules add to the match cost of a production system. Thus, as the production system size increases with learning, the match cost will also increase. There is much data in the literature to support this phenomenon. On the contrary, researchers in parallel production systems have concluded that the match effort in a production system is limited, independent of the size of the production system. Thus, an increase in the size of the production system will not lead to an increase in the match cost. There is much data to support this phenomenon as well.

In this paper, we point out these contradictory views of production match in the two research communities. A direct analysis of these conflicting views is difficult, since the two communities have worked with vastly different systems. Therefore, we have developed some large production systems in Soar, to analyze the situation within a common framework. This common framework narrows down the possible causes for this conflict, and raises important questions for future work.

This research was sponsored by the Avionics Laboratory, Wright Research and Development Center Aeronautical Systems Division (AFSC), U. S. Air Force, Wright-Patterson AFB, OH 45433-6543 under Contract F33615-90-C-1465, ARPA Order No. 7597. The second author was sponsored by the National Science Foundation under a graduate fellowship award.

Keywords: Artificial Intelligence, Machine learning, Explanation-based learning, Parallelism in rule-based systems, Production match

Table of Contents

List of Figures

1. Introduction

Production systems (rule-based systems) are used extensively in the field of AI [Laffey et al. 88, McDermott 82, Newell 90, Waterman and Hayes-Roth 78]. In production systems, computation proceeds by repeated execution of *recognize-act* cycles. In the recognize phase, productions (condition-action rules) in the system are matched with a set of data-items, called working memory. In the act phase, one or more of the matched productions are fired, possibly changing the working memory, and causing the system to execute the next recognize-act cycle.

This paper focuses on the computationally expensive recognize phase, specifically on production match. Does the match effort during a recognize-act cycle increase due to the addition of a new production? Two conflicting views have emerged.

Research in the field of explanation-based learning (EBL) indicates that learned rules add to the match cost of a production system. Thus, as the production system size increases with learning, the match cost continues to increase with it. This increasing match cost soon devours all the gains, and causes an overall slowdown — the *utility problem* [Minton 88a]. There is much data in the literature to support this phenomenon [Cohen 90, Etzioni 90, Minton 85, Minton 88a, Tambe, et al. 90].

On the contrary, researchers in the field of parallel production systems have concluded that the match effort in a production system is limited, independent of the number of productions in the system. Thus, an increase in the number of productions does not lead to an increase in the match effort during a recognize-act cycle. This limited match computation leads to limited match parallelism in production systems. There is much data to support this phenomenon as well [Gupta 86, Gupta, et al. 86, Miranker 87a, Miranker 91, Oflazer 87].

It is important to understand and analyze these conflicting views. In particular, each field has evolved around its particular view of match. In the EBL field, much research has focused on addressing the utility problem, and selectivity about learning new rules has emerged as a popular solution. In contrast, research in parallel production systems has progressed under the assumption of limited match parallelism, with the development of suitable parallel implementations. Thus, an investigation of the conflict could provide new research directions, and lead to a revision in the focus of either one or both fields.

Our analysis in this paper is an attempt to narrow down the scope of the conflict, as well as the factors responsible for the conflict. This analysis cannot be expected to result in a proclamation of one side to be absolutely right or wrong — there is much data to support both points of view. Instead, we attempt to provide a better understanding of the conflict and the factors responsible for it.

To narrow down the scope of the conflict, we focus on production systems with large numbers (thousands or tens of thousands) of productions. Much of the data on the conflict, particularly from the EBL field, is for smaller production systems with up to two-hundred productions. Thus, is the conflict of views confined to small production systems or does it extend to large systems? Understanding this issue is critical for large systems. These production systems could suffer from a catastrophic slowdown in the match; hence it is important to understand their processing requirements and address them adequately. For instance, if large production systems cause a severe slowdown, then would it be necessary to design new match algorithms to provide adequate speedups? Would it be necessary to design new parallel implementations? On the other extreme, if these systems do not cause any slowdown in production match, then could the utility problem be considered solved? and so on.

To narrow down the factors responsible for the conflict, we investigate the large production systems using a common framework — Soar [Laird, Newell, and Rosenbloom 87], an integrated problem-solving and learning system. Research in EBL and parallel production systems is based on vastly different systems. Some or all of the differences among these systems could potentially be responsible for the conflict in views. Choosing a common framework of analysis helps narrow down these differences.

The rest of this paper is organized as follows: Section 2 describes the conflict in more detail. Section 3 attempts to analyze the conflict using Soar. Finally, Section 4 discusses possible implications of our analysis.

2. The Clash of Views

We can obtain a better understanding of the clash of views by focusing on match algorithms, such as Rete [Forgy 82] and Treat [Miranker 87b], that are commonly used in production system implementations. These algorithms employ a two phase strategy. The first phase (called the α or *constant-test* phase) uses a discrimination tree to locate a small subset of productions that can potentially lead to a successful match. The second phase (called the β phase) then processes *only* this small subset of productions. It generates partial instantiations (called tokens) for this subset of productions. Each token indicates what conditions of a production have matched, and with what variable bindings. The β phase performs consistency-checks on the tokens, creates new tokens, generates instantiations for successfully matched productions and so on.

Researchers in parallel production systems [Gupta and Forgy 83, Gupta 86, Miranker 87a, Oflazer 87] have made detailed measurements on a variety of OPS5 [Brownston, et al. 85] production systems, and analyzed them in terms of the computations in the α and β phases. These OPS5 systems vary widely in the number of productions: from ~60 productions to ~2000 productions. The measurements reveal that as the number of productions in these systems increases, the computation in the α phase may show some increase, but the computation in the β phase does not change. The α phase consumes a very small fraction (approximately 5-10%) of the match time in a recognize-act cycle [Gupta, et al. 88]. Moreover, optimizations such as the use of hashing can further reduce the proportion of time spent in the α phase [Barachini and Theuretzbacher 88, Gupta 86]. Thus, an increase in the number of productions leads to almost no increase in the match time per recognize-act cycle. This phenomenon leads to the limited (10-20 fold) match parallelism available in production systems.[1]

Thus, the conclusion in the parallel production systems field is that the computation in the β phase is approximately constant, independent of the size of the production system. In Gupta's words [Gupta 86, page 43]: *...the number of productions that are affected is quite independent of the number of productions present in the system...*, where an affected production is one that causes some computation in the β phase. Miranker [Miranker 91, page 345], states this conclusion as: *...the proportion of time spent in match does not increase with the number of rules in a program.*

This conclusion has played a key role in the development of parallel production system implementations. For instance, Gupta [Gupta 84, page 25] points out that the limited match effort will lead to limited match parallelism and argues that *...since [match] parallelism is limited, we cannot*

[1] Some other minor factors, such as the large variance in the processing of individual productions, contribute in limiting match parallelism as well [Gupta 86]. Also, note that there are are other sources of parallelism besides match parallelism in production systems [Harvey, et al. 91].

effectively use a very large number of processors. Thus, parallel machines with a small number of powerful processors are preferred over specialized massively parallel architectures [Gupta 86, Gupta, et al. 88].

Data from existing EBL systems [Etzioni 90, Minton 88a, Tambe, et al. 90] points in the opposite direction. It predicts an increase in the processing of both the α and β phases with an increasing number of productions in the system. Figure 1 shows the EBL perspective on match cost for a hypothetical production system. The horizontal axis plots the number of productions in the production system. The figure shows that as the number of productions continues to increase, the match time per recognize-act cycle (on the vertical axis) increases as well. This increase in match time is called the *average growth effect*(AGE) [Tambe, et al. 90].[2]

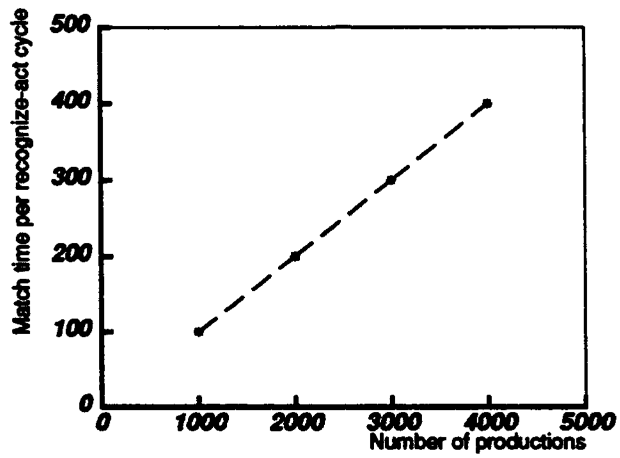

Figure 1: The average growth effect prediction.

In general, the AGE graph may not be perfectly linear. However, the impact of the addition of each new set of productions is non-negligible. Tambe [Tambe 91, page 63] states: *...accumulating a large number of productions may overwhelm the matcher after some time.* Minton [Minton 85, page 596] claims the following about systems that learn new rules from problem-solving experience: *As the system gains experience, it gradually becomes swamped by the knowledge it has acquired. In some cases, the performance can eventually degrade so dramatically that the system operates even more poorly than a non-learning system.*

This view of production match has led to much concern in the EBL field about acquiring new rules. Fisher et al. [Fisher et al. 92] make the following observation about this issue: *...the utility problem, or the exorbitant cost of using the learned knowledge, proved to be a significant obstacle to further progress.* Minton et al. [Minton et al. 87, page 122] claim: *A learning algorithm cannot simply add control knowledge to a problem solver haphazardly; it must be sensitive to the problem-solver's computational architecture and the potential costs of adding knowledge.* The utility issue has now emerged as a key research question in the EBL field [Fisher et al. 92].

[2]While [Tambe, et al. 90] reported on the AGE phenomenon, their main focus was on expensive learned rules, i.e., individual learned rules that required a combinatorial match effort. In contrast, our focus is on the impact of a large number individually inexpensive rules.

Thus, the two fields have completely contradictory views on production match: the EBL field has concluded that production systems will suffer from AGE, while the parallel production systems field has concluded that they will not. Furthermore, these are key conclusions in the two fields, which have influenced the course of their development.

3. Analyzing the Conflict

As discussed earlier, we have focused on production systems containing large numbers of productions to analyze the conflict, since it is particularly relevant in the context of such systems. Additionally, from an experimental perspective, large systems are expected to present a very clear case either for or against the presence of AGE — an AGE, if any, would be significant and clearly visible.

We have developed these large production systems in Soar, an integrated problem-solving and learning system. Soar has already been well-reported in the literature [Laird, Newell, and Rosenbloom 87, Rosenbloom, et al. 91]. For the purposes of this paper, two key points about Soar need to be noted. First, Soar uses a production system for its knowledge-base. Second, Soar uses chunking [Laird, Rosenbloom, and Newell 86], a form of explanation-based learning, to add new productions (called *chunks*) to its production system.

Developing these large systems in Soar helps to narrow down the factors responsible for the conflict. Specifically, much research in parallel production systems is based on OPS5, while EBL research is based on systems such as Prodigy [Minton 88b], Soar [Laird, Newell, and Rosenbloom 87] and others. There are a large number of differences among these systems, e.g., OPS5 syntax and execution semantics differs from the other systems, the match algorithm in OPS5 often differs from that in other systems, OPS5 rules are not generated via EBL, and so on. Some or all of these differences could potentially be responsible for the conflict in views. Choosing a common framework of analysis helps us to narrow down these differences. The following subsections describe the large production systems that we have developed and present the results observed in those systems.

3.1. Dispatcher-Soar

The task of our first system, Dispatcher-Soar [Doorenbos, et al. 92], is to dispatch messages for a large organization, which is represented in an external database; Dispatcher-Soar must make queries of the database to perform the task. The system began with 1,819 initial (unlearned) productions. Over the course of solving a sequence of 200 problems in more than 350,000 recognize-act cycles, it learned 10,112 new productions, bringing the total size to 11,931 productions. This represents one of the largest production systems in existence, and by far the largest number of rules ever learned by an AI system.[3] From the EBL perspective, Dispatcher-Soar would be expected to show a very significant AGE.

Figure 2 presents the result from Dispatcher-Soar[4]. It plots the total number of productions on the horizontal axis. This number increases from 1,819 to 11,931 productions. The figure plots the number of token changes per recognize-act cycle on a log scale on the vertical axis. (See Section 2 for a description

---

[3]The R1/XCON system at Digital is the only production system we know that exceeds 10,000 productions [Barker and O'Connor 89]. Systems such as BMT [Brainware 90], that have substantially more rules, are specialized to have only constants in their patterns, thus avoiding the main source of computational cost, matching of pattern variables.

[4]The results reported in this paper are for Soar version 5.2.2.

of tokens.) The number of tokens generated by the matcher during a recognize-act cycle is an implementation independent measure of the amount of match effort in that cycle [Minton 88b, Nayak, Gupta, and Rosenbloom 88, Tambe, et al. 90]. Since measuring the system's performance after each recognize-act cycle (for the 350,000 plus cycles) would be extremely inefficient, we made measurements at 50-cycle intervals. In Figure 2, each point indicates the average number of token changes per recognize-act cycle during such a 50-cycle interval.[5]

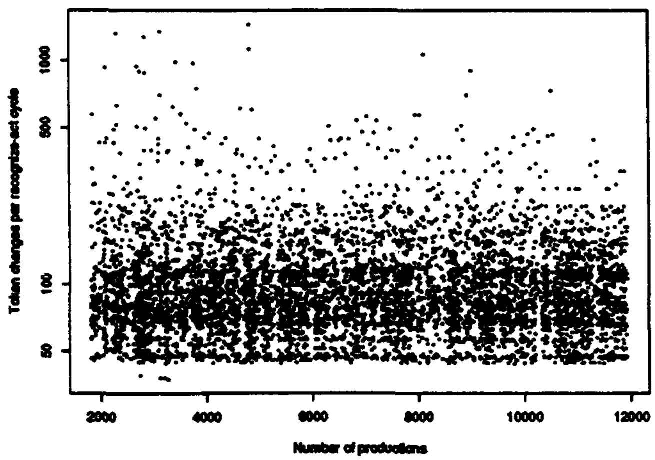

Figure 2: Token changes per recognize-act cycle for Dispatcher-Soar.

The figure shows the absence of any AGE — over the course of the addition of 10,112 new productions, there is no increase in the token changes per recognize-act cycle. In fact, a least squares fit indicates a slightly decreasing trend.

3.2. Path-planner-Soar

Our second system, Path-planner-Soar, plans paths for an agent on a two dimensional surface in the presence of obstacles. The primary concern of this path planning is safety, e.g., avoiding areas which could lead to an "ambush" from a hostile agent. The system is based on the concept of weighted region path-planning [Mitchell 88].

The system started out with 305 productions and learned 1,840 chunks over the course of solving 205 problems in more than 40,000 recognize-act cycles. Figure 3 presents the result from Path-planner-Soar. It plots the number of productions on the horizontal axis, and number of token changes per recognize-act cycle on a log scale on the vertical axis. The figure shows the presence of an average growth effect in Path-planner-Soar. A least squares fit indicates that the match cost is initially 156 tokens per cycle, and increases by 72 tokens per cycle per 1000 new productions. That is, with 1840 new productions (a six-fold increase in the number of productions), the matcher exhibits an approximately two-fold slowdown.

[5]The natural unit of measurement in Soar is actually a decision cycle, which is a collection of approximately five recognize-act cycles. These measurements were made at regular 10-decision cycle intervals.

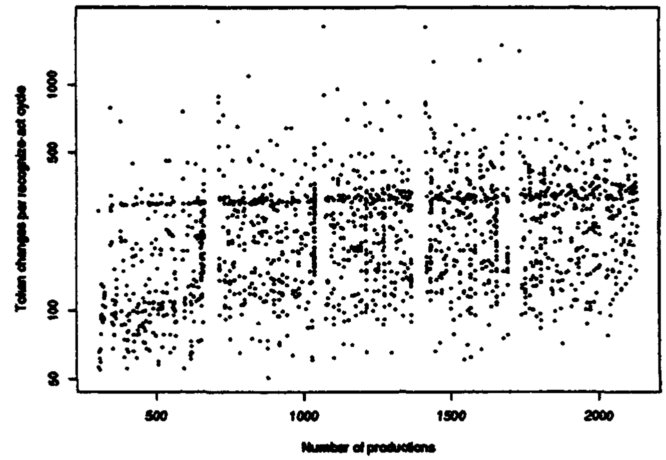

Figure 3: Token changes per recognize-act cycle for Path-planner-Soar.

3.3. Synthetic Systems

In addition to the above "natural" systems, we have also tried two deliberately simple "synthetic" systems involving the memorization technique introduced in [Rosenbloom and Aasman 90], which accounts for about 1/3 of the chunks in Dispatcher-Soar. In each system this technique was used repeatedly to learn a total of 10,200 chunks. (The two systems differed in the structure of the things being memorized.) Figure 4 shows the result for one of the systems; the result for the second system is very similar. Neither of these systems showed any increasing trend in token changes per recognize-act cycle.

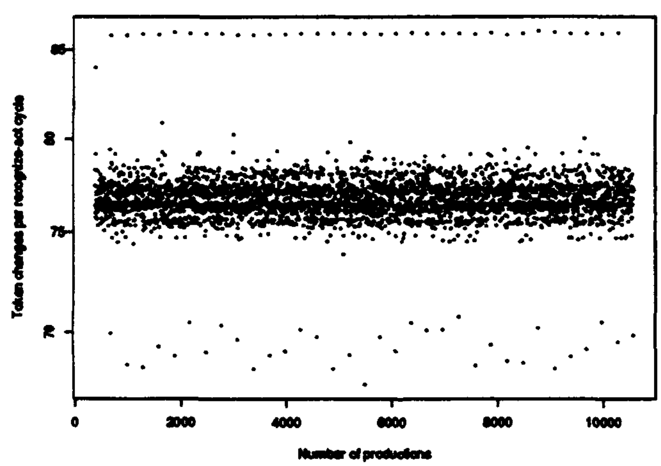

Figure 4: Token changes per recognize-act cycle for a synthetic system.

4. Discussion

In this paper, we focused on the completely contradictory views on production match in two of AI's subfields. Will production systems show an average growth effect? The EBL field has concluded that they will, while the parallel production systems field has concluded that they will not. Given that there is much data to support both points of view, our analysis was not expected to proclaim either side to be absolutely right or wrong. Instead, we attempted to narrow down the scope of the conflict as well as the factors responsible for it.

Our attempt to narrow down the scope of the conflict was based on production systems with large numbers of productions. However, results from the previous section indicate that this attempt was unsuccessful. Despite the addition of more than 10,000 new productions, there is no average growth effect in Dispatcher-Soar and the synthetic systems. But with the addition of 1,840 new productions, Path-planner-Soar shows an average growth effect. The results do not conclusively support either viewpoint, i.e., the conflict extends to large production systems.

However, our attempt to narrow down the factors responsible for the conflict is more successful. The presence and absence of AGE within a common framework — Soar — shows that the clash of views on match is not exclusively due to the differences in OPS5 and other systems. Rather, the presence or absence of AGE appears related to particular tasks (or domains) and encoding styles. This raises the possibility that some task encodings have the potential of causing an average growth effect, and some have the potential of avoiding it.

An obvious hypothesis about task encodings that avoid any AGE is that the new rules in such systems are extremely specific, i.e., they do not match at all. However, this does not seem true, especially in Dispatcher-Soar, where the new rules match often [Doorenbos, et al. 92]. This hypothesis also fails to explain the lack of AGE in OPS5 systems.

From the opposing viewpoint, an obvious hypothesis that explains the presence of AGE in Path-planner-Soar and other systems is the absence of a *smart* match algorithm. In our experiments with Soar, we employed the highly optimized Rete match algorithm [Forgy 82, Scales 86]. However, whether more optimized versions of Rete, or other smarter algorithms could alleviate or eliminate the AGE remains an interesting question for future work. Indeed, an acknowledgement that the existing match algorithms cannot alleviate the AGE, and that new match algorithms need to be explored, could be considered as a key outcome of our investigation.

Clearly, much more data and analysis are required. A lot of issues remain unresolved. It is unclear if a single common phenomenon underlies the lack of AGE in all of the OPS5 and Soar systems. Uncovering this mystery may reveal new solutions to the utility problem. Alternatively, if a large number of systems display a substantial AGE, then that will change the course of the field of parallel production systems — new parallel implementations and algorithms will need to be designed. We are currently in the process of collecting more data for large production systems.

References

[Barachini and Theuretzbacher 88]
Barachini, F. and Theuretzbacher N.
The challenge of real-time process control for production systems.
In *Proceedings of the National Conference on Artificial Intelligence*, pages 705-709. 1988.

[Barker and O'Connor 89]
Barker, V., O'Connor, D.
Expert systems for configuration at Digital: XCON and beyond.
Communications of the ACM 32(3):298-318, 1989.

[Brainware 90] Brainware.
The BMT expert system.
Pragmatica: Bulletin of the Inductive Programming Special Interest Group (IPISG). Spring, 1990

[Brownston, et al. 85]
Brownston, L., Farrell, R., Kant, E. and Martin, N.
Programming expert systems in OPS5: An introduction to rule-based programming.
Addison-Wesley, Reading, Massachusetts, 1985.

[Cohen 90] Cohen, W. W.
Learning Approximate Control Rules of High Utility.
In *Proceedings of the sixth international conference on machine learning*, pages 268-276. August, 1990.

[Doorenbos, et al. 92]
Doorenbos, R., Tambe, M., and Newell, A.
Learning 10,000 chunks: what's it like out there.
In *Proceedings of the National Conference on Artificial Intelligence*, pages (to appear). 1992.

[Etzioni 90] Etzioni, O.
A structural theory of search control.
PhD thesis, School of Computer Science, Carnegie Mellon University, December, 1990.

[Fisher et al. 92] Fisher, D., Subramanian, D., and Tadepalli, P.
An overview of current research on knowledge compilation and speedup learning.
Workshop on knowledge-compilation and speedup learning (held in conjunction with the International conference on machine learning: ML-92).
1992

[Forgy 82] Forgy, C. L.
Rete: A fast algorithm for the many pattern/many object pattern match problem.
Artificial Intelligence 19(1):17-37, 1982.

[Gupta 84] Gupta, A.
Implementing OPS5 production systems on DADO.
Technical Report CMU-CS-84-115, Computer Science Department, Carnegie Mellon University, March, 1984.

[Gupta 86] Gupta, A.
Parallelism in production systems.
PhD thesis, Computer Science Department, Carnegie Mellon University, 1986.
Also a book, Morgan Kaufmann, (1987).

[Gupta and Forgy 83]
Gupta, A. and Forgy, C.
Measurements on Production Systems.
Technical Report CMU-CS-83-167, Computer Science Department, Carnegie Mellon University, December, 1983.

[Gupta, et al. 86] Gupta, A., Forgy, C., Newell, A., and Wedig, R.
Parallel algorithms and architectures for production systems.
In *Proceedings of the Thirteenth International Symposium on Computer Architecture*, pages 28-35. June, 1986.

[Gupta, et al. 88] Gupta, A., Tambe, M., Kalp, D., Forgy, C. L., and Newell, A.
Parallel implementation of OPS5 on the Encore Multiprocessor: Results and analysis.
International Journal of Parallel Programming 17(2), 1988.

[Harvey, et al. 91] Harvey, W., Kalp, D., Tambe, M., McKeown, D., and Newell, A.
The Effectiveness of Task-Level Parallelism for Production Systems.
Journal of Parallel and Distributed Computing 13(4), December, 1991.

[Laffey et al. 88] Laffey, T.J., Cox, P. A., Schmidt, J. L., Kao, S. M., and Read, J. Y.
Real-time Knowledge-Based Systems.
AI magazine 9(1):27-45, 1988.

[Laird, Newell, and Rosenbloom 87]
Laird, J. E., Newell, A. and Rosenbloom, P. S.
Soar: An architecture for general intelligence.
Artificial Intelligence 33(1):1-64, 1987.

[Laird, Rosenbloom, and Newell 86]
Laird, J. E., Rosenbloom, P. S. and Newell, A.
Chunking in Soar: The anatomy of a general learning mechanism.
Machine Learning 1(1):11-46, 1986.

[McDermott 82] McDermott, J.
R1: A rule-based configurer of computer systems.
Artificial Intelligence 19(2):39-88, 1982.

[Minton 85] Minton, S.
Selectively generalizing plans for problem-solving.
In *Proceedings of the Ninth International Joint Conference on Artificial Intelligence*, pages 596-599. 1985.

[Minton 88a] Minton, S.
Quantitative results concerning the utility of explanation-based learning.
In *Proceedings of the National Conference on Artificial Intelligence*, pages 564-569. August, 1988.

[Minton 88b] Minton, S.
Learning Effective Search Control Knowledge: An explanation-based approach.
PhD thesis, Computer Science Department, Carnegie Mellon University, 1988.

[Minton et al. 87] Minton, S., Carbonell, J. G., Etzioni, O., Knoblock, C. A., Kuokka, D. R.,
Acquiring effective search control rules: explanation-based learning in the Prodigy System.
In *Proceedings of the fourth International workshop on Machine Learning*. 1987.

[Miranker 87a] Miranker, D. P.
Treat: A New and Efficient Match Algorithm for AI Production Systems.
PhD thesis, Computer Science Department, Columbia University, 1987.

[Miranker 87b] Miranker, D. P.
Treat: A better match algorithm for AI production systems.
In *Proceedings of the Sixth National Conference on Artificial Intelligence*, pages 42-47. 1987.

[Miranker 91] Miranker, D. P.
Guest Editor's Introduction.
Journal of Parallel and Distributed Computing 13, 1991.
Special issue on parallel execution of rule systems.

[Mitchell 88] Mitchell, J. S. B.
An algorithmic approach to some problems in terrain navigation.
Geometric Reasoning.
In Kapur, D., and Mundy, J. L.,
MIT press, 1988.

[Nayak, Gupta, and Rosenbloom 88]
Nayak, P., Gupta, A. and Rosenbloom, P.
Comparison of the Rete and Treat production matchers for Soar (A summary).
In *Proceedings of the Seventh National Conference on Artificial Intelligence*, pages 693-698. 1988.

[Newell 90] Newell, A.
Unified Theories of Cognition.
Harvard University Press, Cambridge, Massachusetts, 1990.

[Oflazer 87] Oflazer, K.
Partitioning in Parallel Processing of Production Systems.
PhD thesis, Computer Science Department, Carnegie Mellon University, 1987.

[Rosenbloom and Aasman 90]
Rosenbloom, P.S. and Aasman J.
Knowledge level and inductive uses of chunking (EBL).
In *Proceedings of the National Conference on Artificial Intelligence*, pages 821-827. August, 1990.

[Rosenbloom, et al. 91]
Rosenbloom, P. S., Laird, J. E., Newell, A., and McCarl, R.
A preliminary analysis of the Soar architecture as a basis for general intelligence.
Artificial Intelligence 47(1-3):289-325, 1991.

[Scales 86] Scales, D.J.
Efficient matching algorithms for the SOAR/OPS5 production system.
Technical Report KSL-86-47, Knowledge Systems Laboratory, Stanford University, June, 1986.

[Tambe 91] Tambe, M.
Eliminating combinatorics from production match.
PhD thesis, School of Computer Science, Carnegie Mellon University, May, 1991.

[Tambe, et al. 90] Tambe, M., Newell, A., and Rosenbloom, P. S.
The problem of expensive chunks and its solution by restricting expressiveness.
Machine Learning 5(3):299-348, 1990.

[Waterman and Hayes-Roth 78]
Waterman, D. A., Hayes-Roth, F.
Pattern-directed inference systems.
Academic press, 1978.