Versatile Benchmarking for Concurrent Production System Architectures

José Nelson Amaral*
(*amaral@madona.pucrs.br*)
Departamento de Eletrônica
Pontifícia Universidade Católica do RGS
90619-900 - Porto Alegre, RS

Joydeep Ghosh
(*ghosh@pine.ece.utexas.edu*)
Dept. of Electr. and Comp. Engineering
The University of Texas at Austin
Austin, Texas 78712

Abstract

The shortage of adequate benchmarking facilities is a major problem in the evaluation of novel production system machine organizations. This paper presents a survey of benchmark programs used in published research for improvement of production systems. We offer a new benchmark problem that allows independent variation in the size of the database, the number of productions, the ratio between local and global data, and the variance in the size of local data clusters. This new benchmark, available via the Internet through anonymous ftp, is based on the traditional Traveling Salesperson Problem and has the advantage of being both simple and versatile. The advantages of our benchmark are explored through the evaluation of performance of a new production system architecture. Our experiments indicated that the performance of this new architecture scales with the size of the database, but is degraded by a database with high variance in the size of the local data clusters.

1 Introduction

In spite of a high volume of production systems developed for commercial, industrial, military defense, and educational applications in the past fifteen years, academic researchers experience a serious shortage of appropriate benchmarks to test new architectures and system organizations. Production System programs written in the corporate world usually contain the expertise of the business for which they are developed and are seldom made available as benchmarking for academic research. Furthermore, most of these systems have fixed number of productions and fixed database

*Supported by a fellowship from Conselho Nacional de Desenvolvimento Cient´ıfico e Tecnológico (CNPq) and by Pontif´ıcia Universidade Católica do Rio Grande do Sul (PUCRS) - Brazil.

size, which make them inappropriate for researchers, who need the ability to independently change the size of the production set and database.

Many promising architectures and innovative organizations are evaluated with a small number of "toy" problems, or with non-versatile benchmarks that only allow the researcher to modify the size of the database but not its locality properties, or to modify the number of productions without modifying the ratio between productions that access global or locally clusterized data.

In section 2 we present a list of benchmarks commonly used to evaluate new organizations for Production System machines. Section 3 presents a new versatile benchmark facility that allows the researcher to generate benchmark programs with varied data set and production set characteristics.

2 Existing Benchmarks

The benchmarks presented in this sections were used for performance measurement of new organizations and architectures for Production Systems. These benchmarks can be classified as "toy problems" and "real-life programs". In some of the "toy problems", it is possible to expand the size of the database by increasing the number of participants in a tournament, the number of guests in a dinner party or the number of rooms in a hotel. However, in general, these benchmarks have a fixed number of productions and fixed amount of locality in the database. Most of these programs are not sophisticated enough to actually evaluate characteristics of new organizations designed to speed up Production Systems. In the category of "real-life programs" we find sizable production systems that might be useful to evaluate new designs. The problem with these benchmarks is that they are not versatile enough to allow change in the characteristics of the production set or the database.

In tables 1 and 2, the number of productions and the average number of antecedents and consequents per production indicates the size of the benchmark program. The number of WME types indicates how decoupled the database might be. Some authors make available the initial number of WMEs in the working memory, some report the average working memory size throughout the program execution. Under "# WMEs" we report the number available in the publication. This number gives an idea for the size of the database manipulated by each benchmark.

The benchmarks included in Tables 1 and 2 are found in the works of Acharya *et al.* [2, 1], Amaral [3, 5], Bouaud [7], Brant *et al.* [9], Dixit an Moldovan [8], Gupta [10], Kuo *et al.* [11], Kuo and Moldovan [12], Miranker and Lofaso [13, 14], Neiman [15], Oflazer [16], Schmolze [17].

Bench.	# Prod	Ant./prod	Cons./prod	# WME types	# WMEs	Pub.
life	40	6.1	1.3	5	104	[3, 11]
hotel	80	4.1	2.0	62	484	[3]
patents	86	5.2	1.2	4	136	[3]
waltz2	10	2.7	8.0	7	60	[3]
R1	1932	5.6	2.9	31	—	[10]
XSEL	1443	3.8	2.4	36	62	[10]
PTRANS	1016	3.1	3.6	81	—	[10]
HAUNT	834	2.4	2.5	23	60	[10]
DAA	131	3.9	2.9	20	708	[10]
SOAR	103	5.8	1.8	12	353	[10]
mab	26	7.3	—	26	67	[7]
alexia	15	8.4	—	35	3734	[7]
chart	95	4.1	—	78	97	[7]
amd	196	8.2	—	197	1610	[7]
abacab	107	8.0	—	171	1028	[7]

Table 1: Static measures for benchmarks used.

Following we briefly state what each benchmark program does.

In tables 1 and 2, MAB, M&B, mab are different implementations of the classic "monkey and bananas" problem, chart is a syntax chart parser, amd is a semantic analyzer for natural language, abacab is a blackboard controller. R1 configures VAX computer systems, XSEL acts as a sales assistant for VAX computer systems, PTRANS is a program for factory management, HAUNT is an adventure game program, DAA is a program for VLSI design, and SOAR is an experimental problem solving architecture implemented as a production system. Tournament schedule bridge tournaments, waltz, waltz2, and Toru-Waltz are different implementations of the line labeling problem, Cafeteria sets up a cafeteria, hotel models a hotel operation, patents is a solution for the "Confusion of Patents Problem", and life is Conway's game of life. Mud analizes the casting from oil wells, Mapper assists a tourist navigate Manhattan's public transportation system, Mesgen takes Dow Jones figures and converts them into text describing the course of a trading day. Robot plans the movements for a robot arm, Jig25 is a simple jigsaw puzzle solver, Tourney schedules players for a bridge tournament, Weaver is a VLSI box router, and Rubik solves Rubik's cube.

Because of the limitation of the benchmarks encountered in the literature, the research community would greatly benefit from the development of versatile benchmarking facilities that allow not only for the expansion of the knowledge base by replication of data, but that also allows for change in the number of productions, and in the locality proprieties of the knowledge base. It is desirable

Bench.	# Prod	Ant./prod	# WMEs	Pub.
MAB	13	2.6	11	[13, 14]
Mud	884	2.4	241	[13]
Waltz	33	3.9	42	[13]
Mesgen	155	2.9	34	[13, 14]
Mapper	237	3.3	1153	[13, 14]
Jig25	6	—	50	[14]
Tourney	17	—	123	[14]
Robot	75	—	410	[14]
Rubik	70	—	287	[14]
weaver	637	—	152	[2, 9, 14]
waltz	33	—	42	[9, 14]
manners	8	—	—	[9]
ARP	118	—	—	[9]
Cafeteria	94	—	—	[12]
Tournam.	26	—	—	[12]
Toru-Waltz	48	—	—	[12, 11]
Hotel	723	—	—	[12]
Snap	574	—	—	[12]

Table 2: Static measures for benchmarks used.

that such benchmark facilities use a standard production system language to facilitate its use by many different research institutions.

In section 3 we present a new benchmarking facilities that is a modification of the well-known Traveling Salesperson Problem (TSP) that we call the *Contemporaneous TSP* (CTSP). This benchmarking allows the researcher to modify the size of the database, the number of productions, the database size, the amount of locality in the database, and the ratio between productions that access local and global data. All this modifications are implemented by simply modifying a "map" that specifies the location of the cities.

3 A Contemporaneous TSP

In this modified version of the TSP, cities are grouped into "countries". The tour has to be constructed such that the salesperson enters each country only once. The location and borders of the countries must allow the construction of a tour observing this restriction. The problem is formally stated as follows:

An instance of CTSP is represented by $(K, C, c, \mu_c, \sigma_c, O, d)$. $K = \{C_1, C_2, ..., C_n\}$ is a "continent" formed by "countries". Each country $C_i = \{c_{i,1}, c_{i,2}, ..., c_{i,m(i)}\}$ contains $m(i)$ "cities". The number of cities per country $m(i)$ is normally distributed with average μ_c and standard deviation σ_c. The ordering $O =< C_{\pi(1)}, C_{\pi(2)}, ..., C_{\pi(n)} >$ specifies the order in which the countries shall be visited. The function $d(c_i, c_j) \in Z^+$ specifies the distance between any two cities in the continent. The problem consists of finding an ordering of cities $< c_{i,\tau(1)}, c_{i,\tau(2)}, ..., c_{i,\tau(m(i))} >$ within each country C_i that minimizes the cost of the global tour:

$$\sum_{i=1}^{n} \sum_{j=1}^{m(i)-1} d(c_{i,\tau(j)}, c_{i,\tau(j+1)}) \quad + \quad \sum_{i=1}^{n-1} d(c_{i,\tau(m(i))}, c_{i+1,\tau(1)}) \; + \; d(c_{n,\tau(m(n))}, c_{1,\tau(1)}). \tag{1}$$

This formulation of TSP is called "contemporaneous" because it reflects some aspects of modern day life. In the current global economy, travelpersons are likely to have greater needs than the traditional salesperson driving from town to town. Consider a music star in a worldwide tour carrying along a huge crew and sophisticated equipment: the singer will visit many different locations in each continent; the cost of flying back and forth between continents is much higher than movements within a continent and depends on the locations of departure and arrival. Other situations involving sophisticated traveling requirements include the planning of airline routes and national political campaigns in large countries such as USA, Brazil and India. Applications in which data locality allows the creation of clusters include: insurance database management, banking industry, a national health care information network, and a national criminal offense information network[1].

3.1 A Production System Solution for CTSP

The formulation presented above for the CTSP is generic enough to allow its application in many fields: there is no restriction in what the words continent, country, city, and distance might represent. To facilitate the construction of a Production System solution that is useful for testing new PS architectures, we used a simpler version of CTSP with the following restrictions:

- The problem is symmetric, i.e., $d(c_{k,i}, c_{l,j}) = d(c_{l,j}, c_{k,i})$ for any i, j, k, and l.

[1] In the 1994 "Brady Bill", Congress mandated the construction of such a network for background verification for the purchase of fire weapons.

- A continent is a two-dimensional Euclidian space.
- A country is a contiguous, rectangular shape within this space.
- The number of cities in each country follows a normal distribution with average μ_c and standard deviation σ_c.
- The city locations are uniformly distributed within each country.
- There is a common boundary between two countries that are consecutive in the ordering O.

Our PS solution for CTSP has a set of productions for each country and a set of productions for each country boundary. The data set is constructed in such a way that the distances among cities located within each country are stored in WMEs with different types. Given a country C_i, the country that precedes C_i in the order O is denominated $P(C_i)$; the country that succeeds C_i in the order O is denominated $S(C_i)$. It is not necessary to store in the data base the distance between every two cities in the continent. For a city $c_{i,j}$ in a country C_i, the only relevant distances are the distance to the cities within C_i, to the cities in $P(C_i)$, and to the cities in $S(C_i)$. The following list illustrates WMEs typically used in our solution to CTSP:

```
(GERMANY_city ^name GERMANY_01 ^status not_in_trip)
(FRANCE_city ^name FRANCE_10 ^status in_trip)
(GERMANY_dist ^from GERMANY_04 ^to GERMANY_07 ^value 135)
(FRANCE_GERMANY_dist ^from FRANCE_14 ^to GERMANY_03 ^value 357)
(GERMANY_POLAND_dist ^from GERMANY_01 ^to POLAND_05 ^value 55)
```

Our solution has seventeen *local* productions per country and twelve productions per country boundary. This organization allows the researcher to vary the number of productions by creating continents with different number of countries. The size of the data base is determined by the number of countries and the average number of cities per country. The variance between the amount of data processed by each cluster of production is given by σ_c.

The heuristic used in the PS solution of the problem involves the computation of two extra locations for each country C_i: the geometric center of the borders with $P(C_i)$ and with $S(C_i)$. Because we impose the restriction that countries have rectangular shapes in a two-dimensional Euclidian space, the border between two subsequent countries in the tour is always a segment of a straight line. The *border center* $b(C_i, C_j)$ between countries C_i and C_j is the center of the line

segment that forms the boundary. The heuristic used to construct the internal tour in a country C_i is described below:

- The first city $c_{i,k}$ in the internal tour of a country C_i is the city with minimum distance $d(b(C_i, P(C_i)), c_{i,k})$.
- While the internal tour of country C_i is not complete, select a city $c_{i,l} l \in C_i$ such that $d(c_{i,k}, c_{i,l}) - d(c_{i,l}, b(C_i, S(C_i)))$ is minimum, where $c_{i,k}$ is the latest city inserted in the tour.
- Whenever the internal tours of two adjacent countries C_i and C_j are completed, the last city visited in C_i is connected to the first city visited in C_j.
- Whenever there is a segment of tour formed by four cities (c_i, c_j, c_k, c_l) such that $d(c_i, c_j) + d(c_k, c_l) > d(c_i, c_k) + d(c_j, c_l)$, change this segment of tour to (c_i, c_k, c_j, c_l)[2].

This rationale of the heuristic is to add to the internal tour the cities that are close to the latest city included in the tour and far from the border in which the internal tour shall end. There is a limited local optimization of the constructed tour. We developed a C program that allows researchers to specify continent maps and to experiment with different numbers of countries, μ_c, and σ_c.

Two production system solutions were constructed for CTSP. In the first one, identified as `tsp` in Table 3, a single set of productions performs the optimization in all country borders. In the second solution, identified as `tsp2` in Table 3, an specialized set of productions is used in the optimization of each country border. Table 3 presents static measures for instantiations of CTSP considering problems with C countries, with each country having an average of μ_c cities.

Measure	`tsp`	`tsp2`
# of productions	$20\,C + 1$	$30\,C + 1$
# WME types	$8\,C + 8$	$15\,C + 1$
# WMEs in initial database	$C\,(2\,\mu_c^2 + 2\,\mu_c + 3)$	$C\,(2\,\mu_c^2 + 2\,\mu_c + 3)$

Table 3: Static measures for the CTSP benchmark according to C and μ_c

This simplified CTSP offers many advantages for production system benchmarking: the number of productions in the program can be varied by changing the number of countries; the ratio of

[2]The first subscript in the notation $c_{i,j}$ is omited here because these local optimization might occur either within a country or across country's borders.

global to local data is controlled by the average number of cities in each country; the balance in the size of local data clusters is specified by σ_c; and the specification of the continent "map" is very simple making it easy for a researcher to generate new instantiations of the benchmark. The CTSP benchmarking facility is available through anonymous ftp to: `pine.ece.utexas.edu` in `/a/pine/home/pine/ftp/pub/parprosys`.

CTSP was used to measure the performance of a new parallel architecture described in [3, 4, 6]. This architecture is formed by a number of identical processors connected through a bus. At compile time each production is uniquely assigned to a processor according to a partitioning algorithm that takes into consideration inter-production dependencies and workload balance. Each processor stores locally all Working Memory Elements (WME) that are tested by its production antecedents. This architecture uses a *partially informed* selection to choose the next production to be fired, allowing the superposition of the mathing and acting phases of a production system. The results produced were proven to be correct under the serializability criterion. The architecture relies on the use of associtive memories as lookaside tables to guarantee correct operation without global synchronization. In the next section the benchmark presented in this section is used to study this novel architecture.

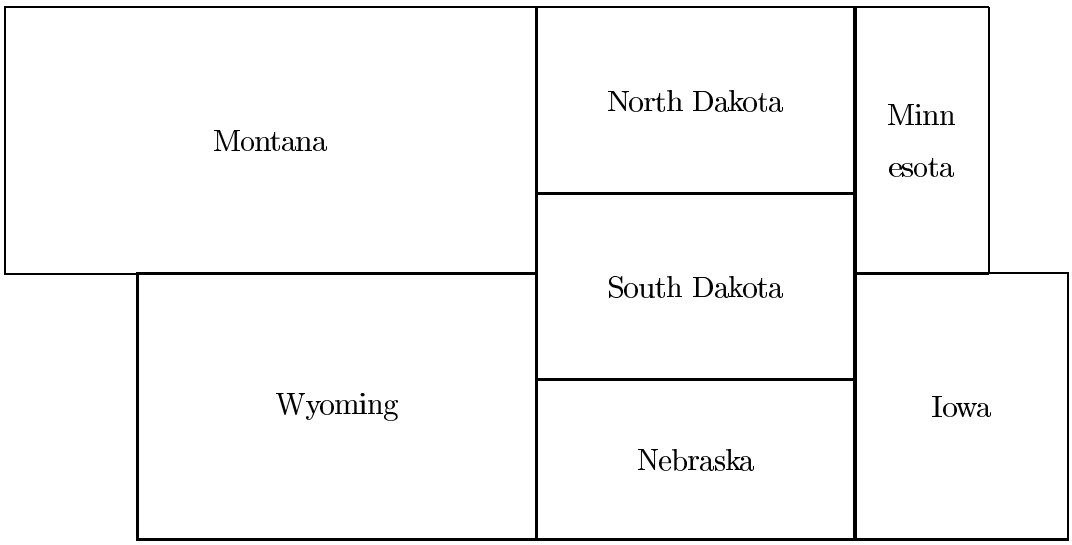

Figure 1: Country map.

4 Performance Evaluation Using CTSP

Benchmarks generated with CTSP are used to evaluate the performance of the architecture described in [4][3]. All experiments presented in this section use an instantiation of CTSP with seven "states"[4]. The mean and the variance for the number of cities per state is set based in the needs of each experiment. The seven-state map is shown in Figure 1. We performed two set of experiments. First, we maintain the standard deviation σ_c constant and change the average number of cities in each state in order to study the effect of problem size on performance. Then we maintain the average number of cities in each state constant and change the standard deviation. This allows us to study the effect of load unbalance among processors.

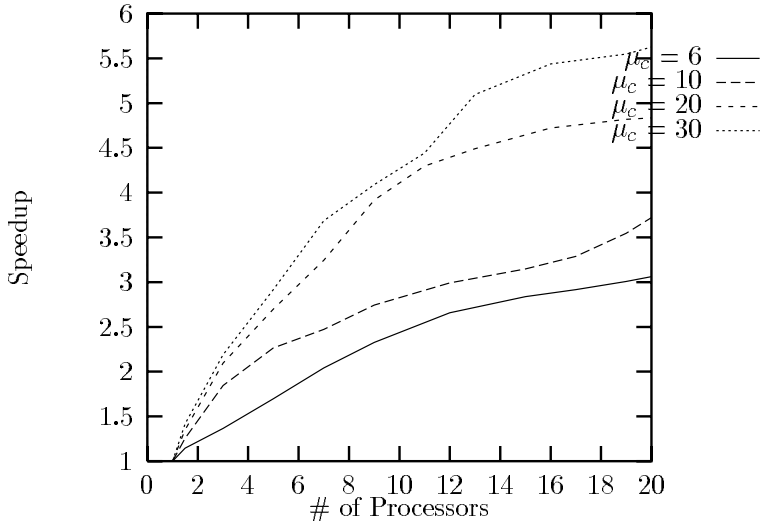

Figure 2: Speedup Curves for 10 β-unit architecture ($\sigma_c = 3$).

Figure 2 plots results for the first experiment in which the mean of the number of cities per state is changed while its variance remains constant. It shows the speed improvement as the number of processors is increased in a machine with a 10 β-unit Rete Network. The base of comparison for these curves is a machine with a single processor and 10 β-units in the Rete Network.

Figure 3 illustrates the effect of the standard deviation in the size of local data clusters. In this graph, the mean for the number of city per state remains constant while the standard deviation

[3]The page limitation for this paper prevent the exposition of the architecture, please refer to [3, 4].

[4]In the description of CTSP in section 3 a group of cities form a "country" and a group of countries form a "continent". In the implementation used for the experiments presented in this appendix, cities form states and states form countries.

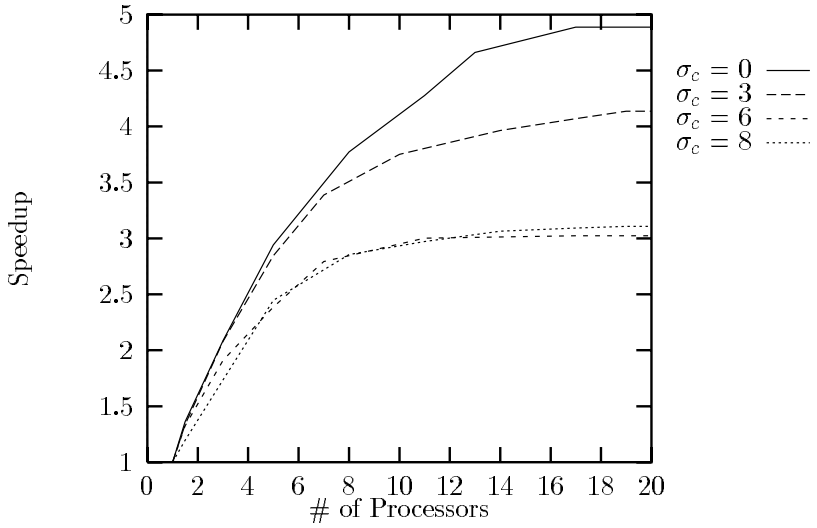

Figure 3: Speedup Curves for 10 β-unit architecture ($\mu_c = 15$).

is varied. Figure 3 plots the curves for a 10 β-unit architecture. The base of comparison for the curves in Figure 3 is a single processor architecture with 10 β-units.

Figure 4 plots the amount of speedup obtained for an instantiation of CTSP with seven states and $\mu_c = 15$ when the standard deviation for the number of cities σ_c is changed. The amount of speedup obtained decreases when there is a higher variance in the number of cities in each state. This is an expected effect because higher variance in the size of local clusters causes more unbalance in processor workload.

5 Conclusion

This paper addresses the problem of shortage of adequate benchmarks to evaluate production system architectures in the research community. We presented a new benchmark facility that allows independent modification in the size of the database, number of productions, ratio between the amount of local and global data, and size of local data clusters.

Benchmarks generated by this facility have the versatility of "toy" problems yet can be expanded to the data and production set size of "real world" problem. These benchmarks also produce correct results when using serializability as a criterion of correctness.

We illustrated the use of the new benchmark facility studying the performance of a novel archi-

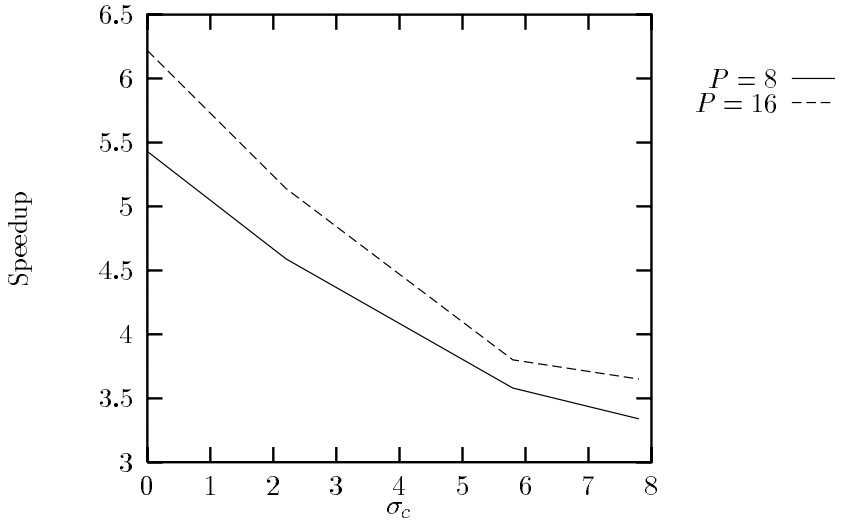

Figure 4: Speedup for single β-unit architecture versus the standard deviation in the number of cities σ_c ($\mu_c = 15$).

tecture for concurrent production systems. The results indicate that for a given standard deviation of the size of local data clusters, the amount of speedup obtained by the architecture scales with the size of the database. Also the architecture performs better in databases with low standard deviation among local data clusters.

References

[1] A. Acharya and M. Tambe. Collection-oriented match: Scaling up the data in production systems. Tech. Rep. CMU-CS-92-218, Carnegie-Mellon University, Pittsburgh, December 1992.

[2] A. Acharya, M. Tambe, and A. Gupta. Implementation of production systems on message-passing computers. In *IEEE Trans. on Parallel and Distributed Systems*, volume 3, pages 477–487, July 1992.

[3] J. N. Amaral. *A Parallel Architecture for Serializable Production Systems.* PhD thesis, The University of Texas at Austin, Austin, TX, December 1994. Electrical and Computer Engineering.

[4] J. N. Amaral and J. Ghosh. An associative memory architecture for concurrent production systems. In *Proc. 1994 IEEE International Conference on Systems, Man and Cybernetics*, pages 2219–2224, San Antonio, TX, October 1994.

[5] J. N. Amaral and J. Ghosh. Speeding up production systems: From concurrent matching to parallel rule firing. In L. N. Kanal, V. Kumar, H. Kitani, and C. Suttner, editors, *Parallel Processing for AI*, chapter 7, pages 139–160. Elsevier Science Publishers B.V., 1994.

[6] J. N. Amaral and J. Ghosh. Performance measurements of a concurrent production system architecture without global synchronization. In *Proc. 9th International Parallel Processing Symposium*, pages 790–797, Santa Barbara, CA, April 1995.

[7] J. Bouaud. Tree: the heuristic driven join strategy of a rete-like matcher. In *Proceedings of the 13th International Joint Conference on Artificial Intelligence*, pages 496–502, August 1993.

[8] V. V. Dixit and D. L. Moldovan. The allocation problem in parallel production systems. *Journal of Parallel and Distributed Computing*, 8:20–29, 1990.

[9] D. A. Brant et al. Effects of database size on rule system performance: Five case studies. In *17th Conference on Very Large Databases*, September 1991.

[10] A. Gupta. Implementing OPS5 production systems on DADO. In *Proceedings of International Conference on Parallel Processing*, pages 83–91, 1984.

[11] C.-M. Kuo, D. P. Miranker, and J. C. Browne. On the performance of the CREL system. *Journal of Parallel and Distributed Computing*, 13:424–441, December 1991.

[12] S. Kuo and D. Moldovan. Implementation of multiple rule firing production systems on hypercube. *Journal of Parallel and Distributed Computing*, 13:383–394, December 1991.

[13] D. P. Miranker. Treat: A better match algorithm for AI production systems. In *Proceedings of National Conference on Artificial Intelligence*, pages 42–47, July 1989.

[14] D.P. Miranker and B. J. Lofaso. The Organization and Performance of a TREAT Based Production System Compiler. *IEEE Trans. on Knowledge and Data Engineering*, pages 3–10, March 1991.

[15] D. E. Neiman. *Design and Control of Parallel Rule-Firing Production Systems*. PhD thesis, University of Massachusets, Amherst, MA, September 1992.

[16] K. Oflazer. Partitioning in parallel processing of production systems. In *Proceedings of International Conference on Parallel Processing*, pages 92–100, 1984.

[17] J. G. Schmolze. Guaranteeing serializable results in synchronous parallel production systems. *Journal of Parallel and Distributed Computing*, 13:348–365, December 1991.