Pre-publication draft of a paper published in the Journal of Experimental Theory in Artificial Intelligence, Taylor and Francis, Volume 2 (1990) 369-380

Improving the Efficiency of Rule-Based Expert Systems by Rule Activation

LOUIS SANZOGNI and FRANCIS SURAWEERA

School of Computing and Information Technology, Griffith University, Nathan, Brisbane, Queensland, Australia 4111

GEOFFREY I. WEBB

Department of Computer Science, Deakin University, Victoria, Australia 3217,

Abstract

In this paper we test a hypothesis that has shown promise in enhancing the efficiency (run-time) of rule-based systems. The results of our experiments suggest that the use of rule activation plays an active part in improving the performance of rule bases containing conflict sets.

Keywords

Activation levels, expert systems, hypothesis testing, production systems, rule activation, rule-based systems, run-time efficiency.

1 Introduction

Since the mid 60's expert systems have emerged as the most significant practical product of artificial intelligence (Al) research. An expert system is based on an extensive body of knowledge about a specific problem area. One of the most successful ways of representing this knowledge is as a production system. In the AI literature, production systems (or simply productions) are also known as rule- based systems or inference systems. The expert system uses the productions to infer new knowledge and conclusions from given data or premises.

Several studies in Cognitive Psychology have found evidence that mental activity causes related cognitive structures to become more highly activated (see for example, Anderson 1975, 1983). The more highly activated, the more readily it can be accessed and the more likely it is to be used. It is reasonable to assume that a piece of information will become active to the degree that it is related to the current sources of activation. Quite simply, the current focus of cognitive activity directs future foci. We presume that a similar mechanism might be of some use in an expert system in finding the best rule to select from a conflict set.

First we note that there are a number of methods for improving the inference process. Some of these methods range from simple to complex, in form and theory (see Forsyth, 1984). In a recent paper Lenat (1984) reported the following:

'The key to intelligent problem solving lies in reducing the random search for solutions . . .'. In line with Lenat's view, the research reported in this paper analyses a hypothesis which has shown promise in enhancing the efficiency of rule-based systems. It is our intention to show that we can improve the performance of an expert system by setting up the rule base according to associative memory concepts. Only the run time efficiency (how long it takes to reach a conclusion), is considered here. Evidently, the run time efficiency is a function of the number of rules that has been searched.

The hypothesis is that:

> *. . . the performance of the system will be improved by associating an activation level with each rule in the rule base.*

We now give the organisation of the rest of the paper. In Section 2 we define some of the terminology that will be used. In Section 3 the experiment to test the hypothesis will be described. The results from the experiment will be presented in Section 4. Lastly, Section 5 gives a discussion and conclusions.

2 Preliminaries

In this section we describe some of the terminology associated with expert systems. Since many of these terms are found in standard text books (for example: Waterman 1986, Hayes-Roth *et al* 1983, Harmon & King 1985, Jackson 1986, Silverman 1987) only those terms which might have an ambiguous meaning are defined here.

Post (1943) was the first to introduce production systems. Later, the same concept was used by Newell and Simon (1972). The knowledge embodied in a production system may be represented as a set of rules of the form:

```
Rule: <name>
    IF     condition 1,
           condition 2,
           . . .
           condition n
    THEN   action 1,
           action 2,
           . . .
           action m
```

where $n \geq 0$ and $m \geq 1$.

The condition-action pairs in the above rule are also referred to as the *antecedent-consequent* pairs or *situation-action* pairs. A *fact* is represented by a rule with no antecedent; that is, when $n = 0$, and $m = 1$.

In a production system the control structure allows only one rule to fire in a given cycle. However, more than one rule may be triggered (or activated) in which case a conflict resolution strategy must be applied to determine which of the triggered rules should fire. The set of activated rules is known as the *conflict set*.

The activation measures the likelihood that a particular piece of knowledge (rule) will be useful at a particular time. Although a great deal of information may be active at any given point of time activation does not necessarily result in behaviour (see Anderson 1983). The activation level will determine the speed of pattern matching and hence the performance. In our system each rule is assigned an integer called its *activation level.* At the beginning, the activation level of each rule is initialised to be zero. During an implementation of the algorithm a routine increments the activation level of those rules that are relevant (conditions satisfied) for attaining a success. A zero activation level indicates that either the rule has no usefulness or it is inactive. As the activation level increases it determines the degree of desirability of that particular rule during the inference process. Further elaboration of these ideas will be presented in Section 3.

3 The experiment

In order to have meaningful results it was imperative to test our hypothesis on a real world problem. In this connection, we examined a few expert systems to select one that matched our requirements. Specifically, we were looking for a tractable (in terms of the number of rules) expert system with a sufficient number of conflict sets. In the end, we selected the Fire Expert System

(FES) rule base which was supplied to us by Richard Davis of the CSIRO Institute of Biological Resources, Canberra. The FES rule base has 110 rules and 19 conflict sets. A total of 36 facts are needed by the knowledge base for inferencing though not all are required simultaneously.

Our proposed knowledge base consists of three components. These are, (i) rules, (ii) facts, and (iii) associations. The rules and facts are standard repertoire of any knowledge base but associations are not. The following examples represent (a) a fact, and (b) a rule, in general.

(a) fact2:	fuel type is (perennial annual sorghum)
(b) rule2:	**IF** season is (early storms) **AND** it has been up to 3 days since last rain
	THEN available biomass is negligible

It is worth noting that in the proposed knowledge base the rules are constructed so that the first proposition is the consequent and the rest, if more than one, are conjunctive antecedents.

The associations used in the proposed knowledge base, may be broadly classified into two types: (a) factual, and (b) predicate/object. Of the two types of associations, the factual associations are very easy to set up. This is done by looking at all possible facts available to the system, and associating rules and facts as shown below.

Fact 1 is needed by rule *r091*

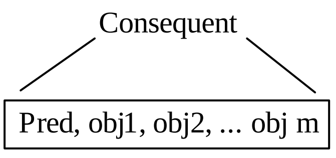
Fact 3 is needed by rule *r103*

Fact 5 is needed by rules *r078 r075 r072 r080 r068*.

where *rxxx* represents a rule.

The inclusion of fact/rule association in the knowledge base was deliberate. Such an organization of the knowledge base will make the information regarding rules and the corresponding facts readily available in a problem solving situation.

Associating predicates and/or objects with rules can be quite difficult. More research is currently being done in this area. However, a logical choice is to link predicates, objects, and facts with the rules most likely to attain a success in the given problem solving context. It is worth noting that each rule in the knowledge base has a consequent, and the consequent itself can be made up of a number of elements; the first element of the consequent is referred to as the predicate, and the rest as the objects as shown below.



The predicates and objects making up the consequent are scanned one at a time, and the rules most likely to have a bearing upon a solution are then associated with the respective predicates and objects. Table 1 shows an example of predicate/object and rules associations. In this study, these associations are done manually. However, it is reckoned that this activity could easily be automated in the future.

In such a knowledge base, a suitably modified inference engine will fire the rule most desired by the associations. The routine shown in table 2 has overall control of the inference process. Its main functions are as follows.

(a) It applies the activations to the rules according to the 'predicate' and 'objects' in the resolvent.

(b) It extracts the predicate from the resolvent and sets up the rules associated with that predicate in descending order of activation.

(c) It unifies the resolvent with the consequent of the next rule in the sorted sequence.

(d) If successful it applies the mgu (most general unifer) to the antecedents of the remainders of resolvents, and the goal. It then restarts the inference process with the next resolvent. If not successful, it moves on to the next rule.

This process is continued until either a resolvent is proved or all the rules are exhausted. In either case, the routine delivers a result eventually.

It is hypothesized that this mechanism improves the speed with which some expert systems reach a conclusion; particularly, in those expert systems with rule bases containing conflict sets.

Associate the following rules with the evaluation of predicate (*is-available-biomass*)

r064	*r069*	*r079*	*r070*	*r068*	*r080*	*r073*	*r071*
r074	*r072*	*r076*	*r077*	*r075*	*r078*	*r106*	*r065*
r111	*r055*	*r052*	*r056*	*r054*	*r053*	*r048*	*r057*
r050	*r047*	*r058*	*r046*	*r049*	*r051*	*r200*	*r045*
r043	*r042*	*r040*	*r039*	*r038*	*r037*	*r110*	

Associate rule *r055* with the evaluation of object *negligible*

Table 1. An example of predicate, object and facts associations

```
FUNCTION Backward_Chain (RS_stack, G)
Let RS_stack and G denote the Resolvent stack and the goal respectively.
IF RS_stack = empty THEN
        output goal G
ELSE    Increment the activation level of the rules associated with the contents of the
        StackTop(RSstack);
        R_stack ← set of rules (in descending activation level order) whose consequent's
        predicates unify with the RS_stack's top member consequent's predicates;
        FINISHED ← FALSE;
        WHILE NOT(R_stack = empty) OR NOT FINISHED DO
                R' ← Pop R_stack;
                RS ← Pop RS_stack; {current goal}
                IF R' unifies with RS
                THEN rename variables in R'
                        A' ← antecedents of R';
                        MGU ← mgu of RS and the consequent of R';
                        Push A' to the RS_stack;
                        Apply MGU to the RS_stack and G;
                        RESULT ← Backward_Chain(RS ← stack, G);
                        IF RESULT THEN FINISHED ← TRUE
        END WHILE
ENDFUNCTION
```

Table 2. The interface routine

A system using activation levels has to perform three extra tasks. These are:

(a) set up the knowledge base and activate rules pertaining to current available facts (once per run);

(b) maintain activation levels; and

(c) sort activation levels in descending order.

It is worth while to note that task (a) is performed on a one-off basis at the beginning of an execution. The other two tasks are performed as on-going processes throughout an execution. In what follows, tasks (a), (b), and (c) are explained further.

3.1 Task (a)

Consider an expert system whose rule base has a conflict set. Such an expert system will have an initial run-time overhead which is greater than that of a conventional system. This overhead is due to the initial setting up of the rule base (Sanzogni 1988). For the knowledge base considered in this paper, the setting up times for non activated versus activated rules were of the order of 1:3.37.

The routine in table 3 sets up the knowledge base in the correct format and is invoked by every rule in the rule base at setting up time. It sets up the rule name and associates with it: *(a)* an initial activation level of zero, *(b)* the rule itself.

```
FUNCTION Set_Rule (Name, Consequent, Antecedents)
    Let PR_stack denote the Predicate Rule stack. Let RN_AL be the activation level
    associated with Rule_Name (RN).

    Set up Rule_Name RN ← Name;
    RN_AL ← 0; {initialize}
    Associate with RN the Consequent C and Antecedents A;
    {i.e. RN_C_A ← C&A}
    Extract the Predicate P from C;
    IF PR_stack does not exist THEN set up PR_stack
    Associate RN to P and add to PR_stack
ENDFUNCTION
```

Table 3 Rule set-up routine

The routine also associates the rule's name with the predicate name of that rule's consequent.

The routine in table 4 increments the activation level of certain rules according to associations that exist between facts and rules that use those facts. It is invoked prior to the beginning of the inference process. It is also worth noting that this routine is not part of the main inference cycle.

3.2 Task (b)

In section 2 we gave some motivation behind the concept of activation level. In this part we illustrate the rule activation strategy using the following simple example drawn from the FES expert system. Consider the resolvent: 'is-available-biomass litter-mass'. The predicate and the object of this resolvent are: 'is available-biomass', and 'litter-mass' respectively. Suppose further that their lists of associations are rules r102 and r102 respectively. Consequently, the activation level of rule r102 is increased by 2. Table 5 presents the algorithm developed for increasing the activation level.

```
FUNCTION Fact_Activation (Facts_&_Associations)
Let F&A_stack and RN_stack denote the Facts_&_Associations stack and Rule Names stack
respectively.
WHILE NOT(F&A_stack = empty) DO
    F&A ← Pop F&A_stack;
    F ← extract fact F from F&A;
    IF F had been used in this run
    THEN set up a stack of rule names RN_stack from F&A associated with F
    WHILE NOT (RN_stack empty) DO
        RN ← Pop RN_stack;
        Increment activation level of RN;
    END WHILE
END WHILE
ENDFUNCTION
```

Table 4. Rule for fact activation routine

```
FUNCTION Activation (Current_Goal, Activation)
    Let A_stack and RN_stack denote the Activation stack and the rule names stack
    respectively. Let PO denotes Predicates and Objects.
    Set up the Predicates & Objects found in Current_Goal in a stack CG_stack.
    WHILE NOT(CG_stack = empty) DO
        PO ← Pop CG_stack;
        Extract from A_stack the rule names associated with PO and stack them in
        RN_stack;
        WHILE NOT(RN_stack = empty) DO
            RN ← pop RN_stack;
            Increment activation level of RN;
        END WHILE
    END WHILE
END FUNCTION
```

Table 5. Activation increment routine

Example

resolvent:	(is-available-biomass litter-mass)
associations:	(is-available-biomass r102)
	(litter-mass r102)
old activation level:	(r102 5)
new activation level:	(r102 7)

where (r102 5) denote that the rule r102 has an activation level equal to 5.

3.3 Task (c)

Recall that our rule base has been especially organized for the purpose of rule activation. As explained earlier, the knowledge base contains a list of predicates associated with all the rules that contain those predicates in their consequents. In the example given below suppose that the predicate ‘is-available-biomass’ is associated with a set of rules S (say). Let us also assume that the current resolvent is ‘is-available-biomass litter-mass’. Then ‘is-available-biomass’ becomes the predicate extracted from the resolvent. All the rules which have this predicate in their consequent (i.e. the rules in the set S) are then sorted and stacked up in descending order of activation level. The rule with the highest activation level is looked at first, since it is the most likely rule that will solve the resolvent.

4 Results

The measurements reported in this study were carried out using the Fire Expert System knowledge base on an Apple Macintosh II running XLISP 1.4. An expert system which employed rule activation was compared with a standard system. The expert system that did not employ the rule activation was designated as the standard one. It is worth noting that no conflict resolution strategy was used by the standard system during test runs.

To assess our hypothesis tests were designed to collect three different types of data, namely:

- run time
- rules visited
- rules fired

The data for each of the runs was chosen to give as much scatter as possible for the rule-path choice.

The data collected from each of the bench marked tests are given in tables 6, 7 and 8.

Figure 1 is a column graph depicting the difference between the number of rules visited by a system not using rule activation and a system using rule activation. All the executions demonstrated a marked decrease in the number of rules visited by the system using rule activation.

Figure 2 is a column graph which shows the difference between the number of rules fired by a system not using rule activation and a. system using rule activation. As shown in figure 2, rule activation has decreased the number of rules fired in 7 out of the 8 runs, and it came equal in run 3. On the basis of the above evidence, we can now say that rule activation plays a major part in guiding the system to reach a conclusion and hence it should show significant savings in execution times. At this point we may deduce that rule activation reduces the number of rules visited by an expert system in arriving at a conclusion.

Run	**Run Time (sec.) (non activated)**	**Run Time (sec.) (activated)**
1	449	136
2	1159	113
3	181	140
4	866	143
5	238	112
6	487	183
7	1648	112
8	1393	113

Table 6. Run time

Run	Rules Visited (non activated)	Rules Visited (activated)
1	545	30
2	1862	20
3	242	19
4	1332	27
5	366	18
6	718	20
7	2615	34
8	2224	25

Table 7. Rules visited

Run	Rules fired (non activated)	Rules fired (activated)
1	14	12
2	22	8
3	8	8
4	20	11
5	7	6
6	11	7
7	26	11
8	28	11

Table 8. Rules fired

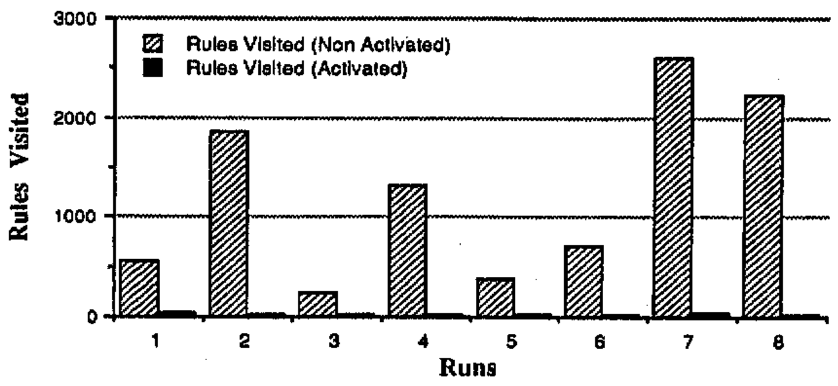


Figure 1. Non activated versus activated rules (Rules Visited)

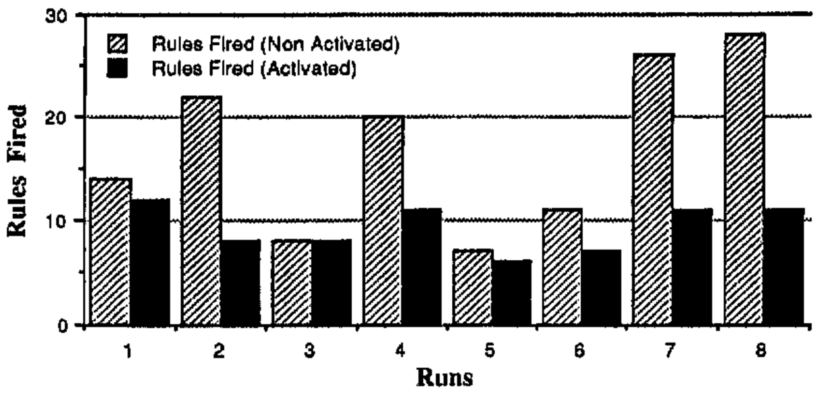


Figure 2. Non activated versus activated rules (Rules Fired)

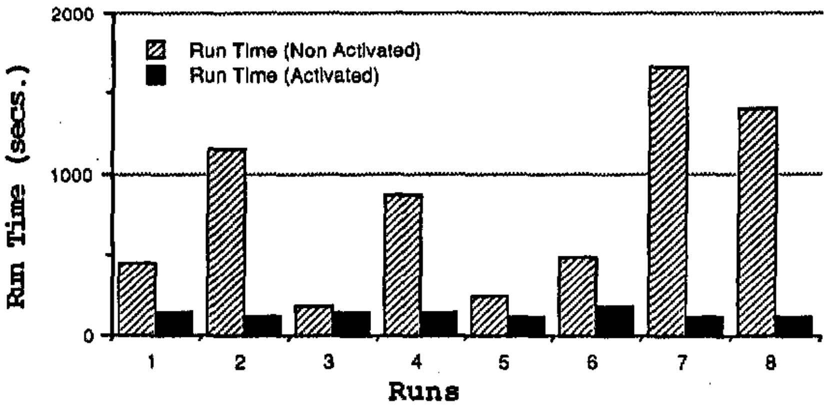


Figure 3. Non activated versus activated rules (Run Time)

Figure 3 shows a column graph which compares the run time (in seconds) between the runs done using rules without activation levels and rules with activation levels. The graph in figure 3 shows a marked improvement in majority of the runs and thus supporting the case for rule activation despite rather heavy overheads. These overheads arise from *(a)* maintenance of rule activation, and *(b)* sorting of rules. Intuitively it may seem that these two activities should slow down the overall time to reach a solution. However, computations using rules which employ activation levels took considerably less time to execute than those using rules with no activation levels. Run 3 had the least improvement, 29%, whilst run 7, at 1372%, had the greatest improvement.

5 Discussion and conclusions

In this paper we presented a method towards improving the efficiency (performance) of an inference system. The method employed was motivated by broad considerations of cognitive processes. The scheme involves increasing the activation level of rules and consequently reducing the costly searches for solutions.

In general, learning can be broadly classified as either (i) enhancing the skills of the learner, or (ii) in acquiring knowledge for the learner (see Firebaugh 1988). The first of these, skill enhancing, is the machine learning analog to performance improvement. In the present study it has been shown that changing activation levels leads to performance improvements. Thus, it is reasonable to state that changing activation levels is a form of *learning*. Of course, a learning system should be able to do the same task(s) more efficiently and effectively the next time by drawing upon past experience. However, it should be apparent by now that our rule-based system is not capable of self improvement through generalization of past experience. This kind of non-learning behaviour is common with deductive systems which employ high-level interpreters. It is known that these high-level interpreters can cause difficulties for learning (Booker *et al.* 1989). They commented:

‘High-level interpreters, by design, impose a complex relation between primitives of the language and the sentences (rules) that specify actions. Typically this complex relation makes it difficult to find simple combinations of primitives that provide plausible generalizations of experience.’

The success of this experimental study is not due to some sophisticated method of inference (see, for example, Holland’s bucket brigade algorithm, rete algorithm, etc.) but it stems from the special architecture of our rule base. More directly, it is related to the rule activation strategy that was employed. To summarise, on the basis of the evidence we have seen, it can be said that a considerable saving in run time was obtained by the expert system using rule activation as compared with the system not using rule activation. This saving in run-time is attributed to the activation levels of the rules. The activation levels force the attention of the inference engine towards those rules which are better suited for the solution of the problem at hand.

At this point it is logical to compare our technique (which basically employs activation-level incrementing algorithms) to an existing technique, viz; Holland’s bucket brigade algorithm. It appears that there are some similarities between activation level incrementing algorithms (ALIA) and Holland’s bucket brigade algorithm (HBBA) but on closer examination it turns out that they have their own distinctive characteristics. Both algorithms are motivated by broad considerations arising from cognitive processes. The HBBA is designed to solve the credit assignment problem for classifier systems (see Sutton 1987, Booker *et al.* 1989). Both methods employ the following:

- induction mechanisms with problem solving; and
- use of condition/action rules as a basic unit of representation.

A classifier system will hypothesize new rules by recombining building blocks (rules) to resolve a problem solving impasse. On the other hand in ALIA this is done by creating a subgoal. Once this subgoal is achieved it generates another subgoal and so on. Thus, the two methods implicitly use different learning mechanisms.

Another point of departure between the two methods is the way in which rules compete to be selected. In HBBA rules compete by bidding on the basis of the strengths and the highest bidding rules (classifiers) go on to the next time step. On the other hand in ALIA rules are selected (conditions satisfied) on the basis of activation levels and the rules are looked at starting from the highest activation level. If it does not unify with the resolvent then it looks at the one corresponding to the next highest activation level and so on. In HBBA control issues tend to come up because several rules are allowed to fire at the same time. Also probability ideas are used in the bidding process where as in ALIA we do not use probability ideas at all. It should be apparent from the above that the two methodologies are quite different despite their initial similarities.

A comparison with the rete algorithm (Forgy 1982) is not attempted at this point in time. A full comparison is being planned and we hope to present its findings at a later date. According to our computational experience we conjecture that only certain types of expert systems lend themselves to rule activation as first proposed, and that such systems can benefit greatly from rule activation. Our study provides an experimental demonstration that the use of rule activation for enhancing the performance of rule-based expert systems shows promise. The trials presented in this paper are by no means conclusive, as they were performed only on one expert system. Nevertheless, they are encouraging.

Acknowledgements

We would like to express our sincere thanks to the Editor and the anonymous referee whose comments have been very helpful in strengthening this paper.

References

[1] Anderson, B. F. (1975) *Cognitive Psychology: The Study of Knowing, Learning and Thinking* (New York: Academic Press).

[2] Anderson. J. R. (1983) *Cognitive Psychology and its Implications* (New York: W. H. Freeman and Company).

[3] Booker, L. B., Goldberg, D. E. and Holland. J. H. (1989) **Classifier systems and genetic algorithms**. *Artificial Intelligence*. 40: 235-282.

[4] Firebaugh. M. W. (1988) *Artificial Intelligence: A Knowledge-base Approach* (Boston: Boyd & Fraser).

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

[6] Forsyth, R. (1984) *Expert System - Principles and Case Studies* (New York: Chapman and Hall).

[7] Harmon. P. and King, D. (1985) *Expert Systems - Artificial Intelligence in Business* (New York: John Wiley & Sons, Inc.).

[8] Hayes-Roth, F., Waterman, D. A. and Lenat, D. B. (1983) *Building Expert Systems* (Reading: Addison-Wesley).

[9] Jackson. P. (1986) *Introduction to Expert Systems* (Reading: Addison-Wesley).

[10] Lenat, D. B. (1984) **Computer software for intelligent systems**, *Scientific American*. September Issue, 152-160.

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

[12] Post, F. (1943) **Formal deductions of the general combinatorial decision problem** *American Journal of Mathematics*. 65: 197-215.

[13] Sanzogni, L. (1988) **Improving the Efficiency of Expert Systems by Rule Activation**. *B. lnf. Honours Thesis*. Griffith University, Nathan, Brisbane, Australia.

[14] Silverman. B. G. (1987) *Expert Systems for Business* (Reading: Addison-Wesley).

[15] Sutton. R. S. (1987) **Learning to predict by the methods of temporal difference**. *Technical Report* TR87-509.l, GTE, Waltham, MA.

[16] Waterman. D. A. (1986) *A Guide to Expert Systems*. (Reading: Addison-Wesley).