International Journal of Computer, Mathematical Sciences and Applications © Serials Publications
Vol. 4, Nos 1-2, Jan-June 2010, pp. 239–262 ISSN: 0973-6786

Rule Scheduling in Active Database Using Learning Automata

ABBAS RASOOLZADEGAN AND M.R. MEYBODI
Department of Computer Engineering and Information Technology, Amirkabir University of Technology (Tehran Polytechnic), Tehran, Iran, E-mail: rasoolzadegan@yahoo.com, mmeybodi@aut.ac.ir

Abstract: Active database systems (ADBS) can react to the occurrence of predefined events automatically by definition a collection of active rules. One of the most important modules of ADBS is the rule scheduler, which has considerable impact on performance and efficiency of ADBS. Rule scheduler selects a rule to execute (evaluate) its action (condition) section in each time through the rules, which are ready for execution (evaluation). We have already evaluated and compared the existing rule scheduling approaches in a laboratorial environment based on three-tier architecture. Five evaluation criteria were recognized and defined formally for evaluation and comparison of rule scheduling approaches including: Average Response Time, Response Time Variance, Throughput, Time Overhead per Transaction and CPU Utilization. At last, we introduced the most effective approach. In this paper, we first, design and implement the before mentioned laboratorial environment again to over and simulate the behavior of ADBS more exactly and completely, then propose a new approach to improve the rule scheduling process based on improvement of triggered rule scheduling using learning automaton. Then, we compare it with the most effective existing approach in the mentioned framework. Results of experiments show that the new method improves the rule scheduling.

Keywords: Estimation of Rule Execution Probability, Active Database Management Systems, Rule Scheduling, Learning Automaton

1. INTRODUCTION

Common (Traditional) database systems often have passive nature. It means operations such as: querying, updating, inserting, deleting, reporting, and, etc. are performed just provided that users request them. So database management system can't automatically react when special situations occur in the system. Many applications such as warehousing programs, automation of factories, systems with financial sophisticated calculations (e.g. stock market), and etc. need automatic supervision to react appropriately when predefined events occur. For supporting this reactive behavior, a new database system has been designed and called Active Database Systems (ADBS). Reactive behavior of ADBS is organized by creating Event-Condition-Action (ECA) rules (active rules). An ECA rule has three main sections: Event, Condition, and Action. When the event occurs, condition gets evaluated and if the condition is true, the action is executed. Below, you can see a simple example of ECA rule defined in an active database system for buying and selling stocks: (12).

DEFINE: *Low Risk*
ON: *Stock.Update Price*
IF: *(Stock.policy = Low_risk) and*
    *(Stock.price < Stock.initprice * e); (0 < e < 1)*
DO: *Stock.Buy*

First of all, in ADBS the application runs until an event occurs, then the rule processing unit is activated and triggers (activates) the appropriate rule(s). Triggered rule(s) is (are) queued in the "triggered rules" buffer. Then a triggered rule is selected according to some special criteria based on a scheduling mechanism for the evaluation of condition. If condition is true, the rule is added to "ready to execute" rules buffer. Then a "ready to execute" rule is selected using a rule scheduling approach and its action section will be executed. The executed rule may trigger some other rules subsequently; new triggered rules will be passed to the rule processing unit. When there aren't any triggered rules, the application continues running. The operations set mentioned above is called rules processing cycle. In summary, there are five different rule processing steps:

1. **Event Signaling:** When a primitive event occurs, the primitive event detector signals it. Additionally, the composite event detector considers these primitive events contributed to composite events (2).
2. **Rule Triggering:** After an event is signaled, ECA rules that correspond to the event signaled are selected, and for each of them rule instances are created. In each rule instance, there is some additional information based on scheduling approach, such as timestamp, deadline, execution time, etc. These rule instances are buffered to use in the next step.
3. **Condition Evaluation:** After buffering rule instances, their conditions are evaluated. Then, for each rule with a true condition, a transaction is generated according to its action section.
4. **Transaction Selection:** This step is also called transaction scheduling phase. In this phase, a selection algorithm (11) operates on execution buffer and selects one transaction which is generated based on triggered rules, and sends the transaction to the execution unit.
5. **Transaction Execution:** Transactions generated based on triggered rules are executed in this phase.

One of the most important features that affects the rule processing phases a lot and plays an important role in the specification of ECA rules are coupling modes. The phases of rule processing discussed so far are not necessarily executed contiguously, but depend on the so-called coupling modes which are pairs of values (x, y) associated with each rule. The value x couples event signaling and condition evaluation of a rule, whereas y couples condition evaluation and action execution. Possible coupling modes are immediate, deferred and independent (5):

- **Immediate mode:** In this mode, when an event occurs, corresponding rule is triggered, then current transaction is suspended and action section of the triggered rule is executed, if the condition holds.
- **Deferred mode:** In this mode, after the occurrence of an event condition evaluation and action execution of the triggered rule is deferred till the end of the current transaction. In deferred mode, the action of triggered rule should be executed before current transaction commits.
- **Independent mode:** When an event occurs in independent mode, there are no time-constraints and restrictions on condition evaluation and action execution of the triggered rule.

The approach used for rule scheduling has great and direct effect on some criteria such as Average Response Time, Throughput and generally on ADBS performance. One of the weak points of ADBS is the rule scheduling approaches which have already been presented. Some of existing approaches were designed for special situations and the rest of them don't have enough performance and efficiency. Rule scheduling approaches in ADBS is an important research topic.

This paper has five sections. In section two, we analyze existing rule scheduling approaches in ADBS. In section three, we introduce a framework to compare and evaluate existing rule scheduling approaches. In this framework, five evaluation criteria have been proposed: Average Response Time, Response Time Standard Deviation, Throughput, Time Overhead per Transaction and CPU Utilization. At the end of this section the approach which has the most positive impact on performance and efficiency of ADBS has been introduced by analyzing the weaknesses and strengths of existing approaches. Then, in section four we introduce a new algorithm for estimation of condition correctness probability of triggered rules using learning automaton and develop a new scheduling approach based on it. Then we show the positive impact of this algorithm on performance of ADBS. Finally, in section five, there is a conclusion of subjects presented in this paper.

2. EXISTING RULE SCHEDULING APPROACHES

In ADBS, the process of priority allocation to rules, ready for execution, is called rule scheduling. As we have also mentioned before, a rule gets ready for execution if and only if, firstly, get activated because of occurrence of the corresponding event in the system and secondly, its condition seems true in evaluation time. In this section, we briefly describe the approaches used for rules scheduling. In this paper, we use rule and transaction terms interchangeably. Figure 1, shows the formal specification of a scheduling method in general.

```
1. Rule_Base (Set of ECA Rules)
2. Active_Rule_Base (Sub Set of Rule-Base)
3. Ready To Execute_Rule_Base (Sub Set of Active_Rule_Base)
4. $n \equiv (i \in N \mid i =$ Number of ECA Rules in the ready To Execute_Rule_Base$)$
5. Rule_Scheduling (Ready To Execute_Rule_Base)
```

Fig. 1: The Formal Specification of Scheduling Method in General

2.1. Random Scheduling Approach

Random selection is one of the easiest approaches for rule scheduling in ADBS (8). This approach has been implemented in RPL and Ode active database systems (11). In the Random approach ADBMS selects randomly one of the activated rules. The most important characteristic of this approach is its simplicity, at the cost of efficiency. The formal specification of the random scheduling method in ADBMS is obtained by replacing line 5 in Fig. (1) with the following pseudo code.

```
5. Random_Generator $(n)$ $(i \in N \mid 1 \le i \le n$ and $i$ selected randomly$)$
   Rule_Scheduling (Ready To Execute_Rule_Base) $\equiv$
   $\{\forall R_i \in$ Ready To Execute_Rule_Base, $(R_1, R_n) \mid R =$
   Ready To Execute_Rule_Base $[R_{Random\_Generator\ (n)}]\}$
```

2.2. Static Priority Scheduling Approach

In this approach, the system assigns a numeric priority to each ECA rule but the priorities need not be unique. In the Ariel and POSTGRES systems, each rule is assigned a priority between – 1000 and + 1000. When an activated rule should be selected to run, the rule that has the minimum static priority is selected (8), (13), (17). The formal specification of static priority method in ADBMS is obtained by replacing line 5 in Fig. 1 with the following pseudo code.

```
5. Rule_Scheduling (Ready To Execute_Rule_Base) $\equiv$
   {$\forall$ $R_i \in$ Ready To Execute_Rule_Base, [($R_1$, $P_1$), ..., ($R_n$, $P_n$)], $R_i$
   Selected | [$R_i \in$ Ready To Execute_Rule_Base | $1 \le i \le n$]
   [$\forall$ $R_j \in$ ReadyToExecute_Rule_Base | $1 \le j \le n, P_i \le P_j$]}
```

2.3 FCFS Scheduling Approach

FCFS (First Come First Serve) scheduling approach is one of the classic approaches used for rule scheduling in ADBS (8), (14). When an event occurs and rules are triggered, an instance of each triggered rule is generated. This instance of triggered rule contains a timestamp which shows the time when the rule is triggered. When an activated rule should be selected to run, the activated rule that has the earliest timestamp is selected. This scheduling approach is used in SAMOS. The formal specification of FCFS method in ADBMS is obtained by replacing lines 5 in Fig. 1 with the following pseudo code.

```
5. Set_TimeStamp ($R$) (R' s Time Stamp = Current Time)
   Activate ($R$) [Create Instance of $R$ and Set_Time Stamp ($R$)]
   Rule_Scheduling (Ready To Execute_Rule_Base) $\equiv$
   {$\forall$ $R_i \in$ Ready To Execute_Rule_Base, [($R_1$, $T_1$) ($R_n$, $T_n$)], $R_i$
   Selected | [$R_i \in$ Ready To Execute_Rule_Base | $1 \le i \le n$
   [$\forall$ $R_j \in$ Ready To Execute_Rule_Base | $1 \le j \le n, T_i \le T_j$]}
```

2.4. Concurrent Execution Scheduling

Approach HiPAC active database system (12) supports this method. Rule processing in HiPAC is invoked whenever an event occurs and triggers one or more rules. This method differs from most other rule scheduling methods in its handling of multiple triggered rules. So we can not quantatively compare it with other rule scheduling methods which are serial. HiPAC executes all triggered rules concurrently. This means that if during rule execution, additional rules are triggered, they are executed concurrently. Another ADBMS called FAR (5) also uses concurrent rule execution.

2.5. EDF Based Scheduling Approach

Earliest Deadline First (EDF) is one of the classic algorithms for transaction scheduling in real-time systems (1), (5), (7). The EDF based approach is one of the best approaches introduced for rule scheduling till now. This approach has been presented for Real-time Active Database (RADB). In this approach, rules are scheduled based on their deadline. This approach has three different versions: 1. EDFPD, 2. EDFDIV, and 3. EDFSL. The EDFPD is a static baseline policy where the rules priorities do not change with time. EDFDIV and EDFSL are dynamic policies where rules priorities change depending on the amount of dynamic work they have generated (16). The formal specification of EDF based method in ADBMS is obtained by replacing line 5 in Fig. 1 with the following pseudo code.

```
5. Activate $(R) \equiv$ [Create Instance of $R$ and Set_Dead Line $(R)$]
   Rule_Scheduling (Ready To Execute_Rule_Base) $\equiv$
   {$\forall\ R_i \in$ Ready To Execute_Rule_Base, [$(R_1, D_1)$ $(R_n, D_n)$], $R_i$
   Selected |[$R_i \in$ Ready To Execute_Rule_Base | $1 \leq i \leq n$
   [$\forall\ R_j \in$ Ready To Execute_Rule_Base | $1 \leq j \leq n$, $D_i \leq D_j$]}
```

2.6. E_x-SJF Scheduling Approach

This method is based on Shortest Job First algorithm. The SJF algorithm is one of the most effective classic scheduling approaches (4), (10). This algorithm is not useful for rule scheduling in ADBS due to active work load nature (16) of it. So there is defined preprocesses for preparing rule base to use the SJF algorithm for rule scheduling in E_x-SJF (Extended SJF) approach (10). The difference between SJF and E_x-SJF is in manner of transactions (rules) execution time calculation. In E_x-SJF approach, the execution time of each rule (parent) is related to the number of its immediate and deferred child rules. According to the manner of interference of the execution time of immediate and deferred child rules in the execution time of their parent rules, there are four versions of E_x-SJF which are named E_x-SJF_{EXA}, E_x-SJF_{PRO}, E_x-SJF_{PRO}-V.1.8, and E_x-SJF_{PRO}-V.2.8 (10).

We explain the E_x-SJF approach more exactly than other approaches because the new approach introduced in Section 4 is an improvement of E_x-SJF. The formal specification of E_x-SJF method in ADBMS is obtained by replacing line 5 in Fig. 1 with the following pseudo code.

```
5. Rule_Scheduling (Ready To Execute_*Rule_Base)* $\equiv$
   {$\forall\ R_i \in$ Ready To Execute_Rule_Base, [$(R_1$, Exec $T_1)$ $(R_n$, Exec $T_n)$],
   $R_i$ Selected | [$R_i \in$ Ready To Execute_Rule_Base | $1 \leq i \leq n$
   [$\forall\ R_j \in$ Ready To Execute_Rule_Base | $1 \leq j \leq n$, Exec $T_i \in$ Exec $T_j$]}
```

Although calculation of rules execution time is possible in run time, it leads to an inefficient scheduling approach because of its too much time overhead. So all versions of E_x-SJF, calculate execution time of the rules before system's run time. As we mentioned before, executing of active rules depend on condition evaluation result of those rules in evaluation time. In other words, execution probability of the activated rules equals condition correctness probability of those rules. So we should estimate execution probability of the activated rules before system's run time. More precise estimation of rule execution probability leads to more precise calculation of rule execution time (10).

The condition section of the rules consists of conditional expressions, database query statements, recalling the procedures, functions, and logical composition of them. Suppose that condition section of rule *R* is defined like $[(A \cap B) \cup (C \cap D)$. *A*, *B*, *C*, and *D* are logical statements. So correctness probability of condition of *R* is calculated as below:

$$P[(A \cap B) \cup (C \cap D)] = P(A \cap B) + P(C \cap D) - P(A \cap B \cap C \cap D)$$

Supposing that *A*, *B*, *C*, and *D* are independent from each other, we will have:

$$P[(A \cap B) \cup (C \cap D)] = P(A) * P(B) + P(C) * P(D) - P(A) * P(B) * P(C) * P(D) \quad \text{... (1)}$$

If correctness probability of logical statements *A*, *B*, *C*, and *D* exist definitely, with putting their values in relation (1), we can calculate the correctness probability of condition of *R* precisely. But correctness probability of a conditional statement often doesn't exist before its execution. So we should estimate correctness probability of conditions before system's run time. All versions of E_x-SJF, i.e. E_x-SJF$_{EXA}$, E_x-SJF$_{PRO}$, E_x-SJF$_{PRO}$-V.1.8, and E_x-SJF$_{PRO}$-V.2.8 construct rule execution tree to anticipate the children of rules (4). In fact, the difference of these versions is in the manner of estimation of the execution probability of their children.

2.6.1. *E_x-SJFEXA Scheduling Approach*

In this approach condition correctness probability of each rule (rule execution probability) is considered 1 (16). So rules execution time is calculated by relation (2) supposing *P* (*R*) = 1 (4).

$$X(R) = L(R) + \sum_{i=1}^{n^{imm}(R)} P(R_i) * X^{imm}(R_i) + \sum_{j=1}^{n^{def}(R)} P(R_j) * X^{def}(R_j) \quad \text{... (2)}$$

$P(R_J)$	The correctness probability of condition section of deferred rule R_j activated by *R*
$P(R_I)$	The correctness probability of condition section of immediate rule R_i activated by *R*
$X^{imm}(R_i)$	The execution time of rule R_i triggered in immediate mode
$X^{def}(R_j)$	The execution time of rule R_j triggered in deferred mode
$L(R)$	The primary execution time of rule *R*
$n^{def}(R)$	Number of deferred rules triggered by *R* during its execution
$n^{imm}(R)$	Number of immediate rules triggered by *R* during its execution
$X(R)$	The execution time of rule *R* calculated by this approach

2.6.2. *E_x-SJF$_{PRO}$ Scheduling Approach*

In this approach, correctness probability of each conditional statement in the condition section of rules is equally considered 0.5. So correctness probability of the condition of *R* mentioned in last section according to relation (1) will become:

$$P(R) = P[(A \cap B) \cup (C \cap D)] = \frac{1}{2} * \frac{1}{2} + \frac{1}{2} * \frac{1}{2} - \frac{1}{2} * \frac{1}{2} * \frac{1}{2} * \frac{1}{2} = \frac{7}{16}$$

According to the above matters, execution time of each rule in this approach is also calculated by relation (2) (4).

2.6.3. E_x-SJF$_{PRO}$-V.1.8 Scheduling Approach

In this approach, at first, execution time of each rule is calculated like E_x-SJF$_{PRO}$ (3), (10). Then in every time which rule R_1 with the shortest execution time among activated rules is selected for condition evaluation, correctness probability of each logical statement of condition section of R_1, i.e. $P\ (R, LS_i)$ is repeatedly calculated and stored based on relation (3).

$$P\ (R, LS_i) = \frac{T1}{T2} \qquad \text{... (3)}$$

(T_1 shows the frequency that LS_i has been evaluated and has been true so far and T_2 shows the frequency that R has been activated so far).

This process is repeated for each logical statement of condition section of the corresponding rule until the changes rate of correctness probability of that logical statement, achieves to a desired value (e.g., 0.0000001 and in general mode ε). At this time, new value is replaced with primary default value (i.e. 0.5). It is evident that the smaller value ε, the more exact calculation of $P\ (R, LS_i)$. When correctness probabilities of all logical statements of a condition are updated, correctness probability of that condition is updated, too. And ultimately execution time of rule R is updated, when condition section correctness probability and all R's childes execution time are updated. So, after passing a short time from start of executing the system (which this time is very insignificant in compare with total executing time of system), execution time of all rules are calculated with a satisfied precision (which amount of this precision is depend on value ε). The formal specification of E_x-SJF$_{PRO}$-V.1.8 method is obtained by replacing pseudo code presented in Table 1 instead of line 5 in Fig. 1.

Table 1: Pseudo Code of Rule Execution Time Calculation and Method of Rule Selection in E_x-SJF$_{PRO}$-V.1.8

When Active Rule R is selected among Active Rules for condition evaluation by Scheduling Approach, LS_Probability0 is called for all conditional Statements of condition part of Rule R

```
LS_Probability (LS)
{Counter 1 = Counter 1 + 1
IF Check_Correctness (LS) Then
   Counter 2 = Counter 2 + 1
LS → New_Probability = counter 2/counter 1
IF –ε < = LS → New_Probability – LS → Old_Probability < = ε Then
   LS → fixed = True
   Condition_Probability (R → C)
LS → Old_Probability = LS → New_Probability}
```

Table Contd.

Table 1 Contd.

Condition_Probability (R $\rightarrow$ C)

```
{For Each $LS$ of $R \rightarrow C$ That $LS$ fixed = False $Do$
    $LS$_Probability ($R \rightarrow C \rightarrow LS$)
    IF $R \rightarrow C \rightarrow LS \rightarrow$ Fixed = False Then
    $R \rightarrow C \rightarrow$ Fixed = False
    Return ($R \rightarrow C \rightarrow$ Fixed) $R \rightarrow C \rightarrow$ Fixed = True
    Calc_Execution_Time ($R \rightarrow$ parents)
    Return [Calculate_Probability ($R \rightarrow C$)]}
```

"At first $R \rightarrow$ visited = False for all Rules"

Calc_Execution_Time (R)

```
{IF $R$ = NULL Then Return (0)
    For Each Child of $R$ $DO$
    IF $R \rightarrow$ Child $\rightarrow C \rightarrow$ Fixed = True
    Then $R \rightarrow$ Fixed = True
    IF ($R \rightarrow$ Fixed = True) or ($R \rightarrow$ visited = False) Then [$I = 0$
    For Each Child of $R$ $DO$
    $I = I$ + Calc_Execution_Time ($R \rightarrow$ Child)
    $R \rightarrow$ visited = True
    $R \rightarrow$ ExecutionTime =
    $R \rightarrow$ ExecutionTime * Condition_Probability ($R \rightarrow C$) + $I$ ]
    Return ($R \rightarrow$ ExecutionTime)}
```

Rule_Scheduling (Ready To Execute_Rule_Base) $\equiv$

```
{$\forall\ R_i \in$ Ready To Execute_Rule_Base,
    [($R_1$, Exec $T_1$) ($R_n$, Exec $T_n$)], $R_i$ Selected |[$R_i \in$
    Ready To Execute_Rule_Base|$1 \le i \le n$
    $\forall\ R_j \in$ Ready To Execute_Rule_Base|$1 \le j \le n$,
    $R_i \rightarrow$ Execution Time $\le R_j \rightarrow$
    Execution Time]}
```

2.6.4. E_x-SJF$_{PRO}$-V.2.8 Scheduling Approach

Both E_x-SJF$_{PRO}$-V.2.8 and E_x-SJF$_{PRO}$-V.1.8 are designed and implemented based on estimation of rule execution probability. The difference of these two approaches is in the manner of estimating (3). In this sub section, we are going to illustrate the operation of this technique. As we mentioned before, condition section of each rule is composed of some logical statements. Table 2, shows the condition section of some rules and Table 3 shows the domain of the conditional variables of condition section of the rules mentioned in Table 2. For example condition section of rule R_1 is true if data item A (D_{IA}) is greater than data item B (DI_B).

Table 2: Condition Section of Some Sample Rules

Rules	*Condition Section of Rules*
R_1	$DI_A > DI_B$
R_2	DI_C = (steel *or* copper *or* cement)
R_3	$DI_B = 20$ and $DI_D < = DI_A$
R_4	$DI_A < 20$ and DI_C = (brass)

Table 3: Domain of Conditional Variables of the Above Table's Rules

$0 < DI_A < 100$	$-110 < DI_B < 50$	$-2000 < DI_D < 50$
DIC = (shoe, steel, cement, copper, brass, carpet, orange, silver, gold)		

According to this fact that both condition section of all rules and the domain of conditional variables of all condition sections are specified before run time and on the assumption that probability distribution of conditional variables is equal and logical statements are independent, we can calculate correctness probability of each logical statement before system's run time mathematically. Assumption of equal probability distribution of conditional variables means that occurrence probability of all valid values in the domain of those variables is equal.

Here we are going to calculate correctness probability of the condition of rule R_1 according to assumptions and known information mentioned in Tables 2 and 3. As it is shown in Fig. 2, DI_A is greater than DI_B in each point inside the trapezoid $S1$ and DI_A is smaller than DI_B in each point inside the triangle $S2$. So correctness probability of $DI_A > DI_B$, on the assumption that area of the trapezoid S_1 is equal to S_{S1} and area of the triangle $S2$ is equal to S_{S2}, is calculated according to relation (4):

$$P(R_1) = \frac{S_{S1}}{S_{S1} + S_{S2}} \qquad \text{... (4)}$$

Then the appropriate values are assigned to the variables of relation (4), so we will have:

$$P(R_1) = \frac{(110*100) + [(100+50)*25]}{100*160} = \frac{14750}{16000} = 0.92$$

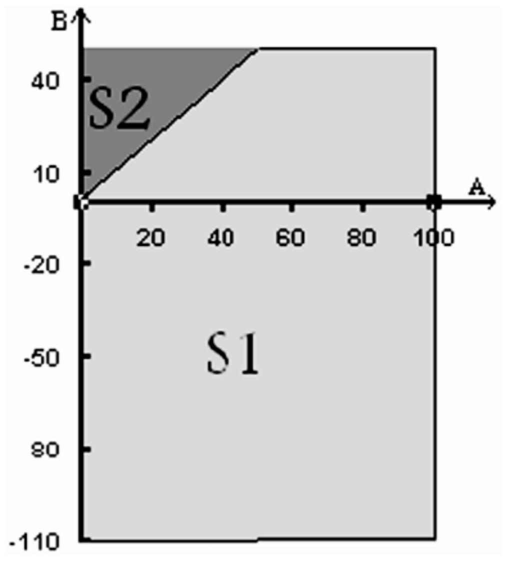


Fig. 2: Calculation the Correctness Probability

In this same manner, correctness probability of all rules (execution probability of the action section of all rules) is calculable before system's run time. After calculating the execution probability of action section of rules, we can calculate execution time of rules according to relation (2). Now system starts its work and scheduler, in each instant, select the rule which has the shortest execution time through activated rules with true condition to pass the operations of its action section to transaction execution unit for execution. But as we mentioned before, in this method, probability distribution of conditional variables is assumed equal in calculating of the rules execution time. But in real systems, probability distribution of conditional variables isn't often equal. So we act as below to correct this assumption to calculate execution time of rules more exact during executing the system.

After that the system starts to work, in specific intervals (δt_i), for every conditional variable of the condition section of the rules, all taken values with their taken intervals are recorded. Table 4 shows the above issue for DI_A.

Table 4: Manner of Recording the Taken Values with their Taken Intervals for DI_A by Passing the Time

Specific Intervals	δt_1 *(1000 min)*		δt_2 *(1000 min)*		...	δt_n *(1000 min)*	
Conditional variable	*Taken values*	*Taken intervals (by min)*	*Taken values*	*Taken intervals (by min)*		*Taken values*	*Taken intervals (by min)*
	2	10	23	8		20	102
	67	5	47	28		71	50
DI_A	⋮	⋮	⋮	⋮		⋮	⋮
	32	39	86	95		11	320
	14	15	4	5		1	55

At the end of every δt, occurrence probability of every value of the domain of every conditional variable is corrected based on taken interval by relevant variable during last δt (16). This process is repeated for each value of the domain of every conditional variable until the changes rate of occurrence probability of that value reach the agreeable extent (e.g., 0.001 and in general mode ε). At this time, new value is replaced with primary default value calculated based on the assumption of equal probability distribution. It is evident that the smaller value ε, the more exact calculation of distribution probability of the conditional variables that finally leads to more exact calculation of the rules execution time. When occurrence probability of all domain's values of all conditional variables existed in the condition part of a rule such as rule *R* are updated, correctness probability of the condition of rule *R* is mathematically calculated again (updated) based on the real probability distributions of its conditional variables and the new value is replaced with primary value. Ultimately execution time of rule *R* is updated based on relation (2), when correctness probability of its condition section and the execution time of all its children are updated. So, after passing a short time from start of executing the system (which this time is insignificant in compare with total executing time of system), execution time of all rules are updated with a satisfied precision (which amount of this precision is depend on value). We do all these calculations when the system is idle. In other words, we determine the length of *t* in such a manner that its termination and idle times of system get concurrent.

3. EVALUATIONAND COMPARISON OF EXISTING RULE SCHEDULINGAPPROACHES

In reference (16) a framework is introduced for comparison and evaluation of existing rule scheduling methods. This framework contains five evaluation criteria according to their priorities [W_i in relation (5)]: Average Response Time, Response Time Standard Deviation, Throughput, Time Overhead per Transaction and CPU Utilization. Table 5 defines these parameters formally.

Table 5: Formal Definition of Evaluation Parameters

N = Number of Executed Rules

ART = Average Response Time

$RTSV$ = Response Time Standard Variance

U_{CPU} = CPU Utilization

$TOPT$ = Time Overhead Per Transaction

T_1^i = Activation Time of ith Rule

T_2^i = Start of Execution Time of ith Rule

$T = (T_2^N +$ Execution Time of Nth Rule$) - T_1^1$

$T^* = \sum_{i=1}^{N}$Re al Execution Time of ith Rule

$$U_{CPU} = \frac{T^*}{T} * 100 \qquad RTSV = \sqrt{\frac{\sum_{i=1}^{N}\left[\left(T_2^i - T_1^i\right) - ART\right]^2}{N}}$$

$$ART = \frac{\sum_{i=1}^{N}\left(T_2^i - T_1^i\right)}{N} \qquad \text{Throughput} = \frac{N}{T}$$

$$TOPT = \frac{T - T^*}{N}$$

In this framework a laboratorial environment named Active Database System Simulator (ADSS) is designed and implemented to simulate the active database system behavior. So we can implement each rule scheduling approach and consider the performance of it in ADSS. Architecture, the manner of designing and implementation of ADSS and rule scheduling approaches are extensively described in reference (16). Figure 3 illustrates the architecture of the ADSS (we illustrate the position of learning automata and its relations with other parts in this architecture especially for E_X-SJF_{EsTLA}). An important characteristic of ADSS is flexibility. It means that we can implement each rule scheduling approach, only by replacing the scheduling algorithm in the ADSS.

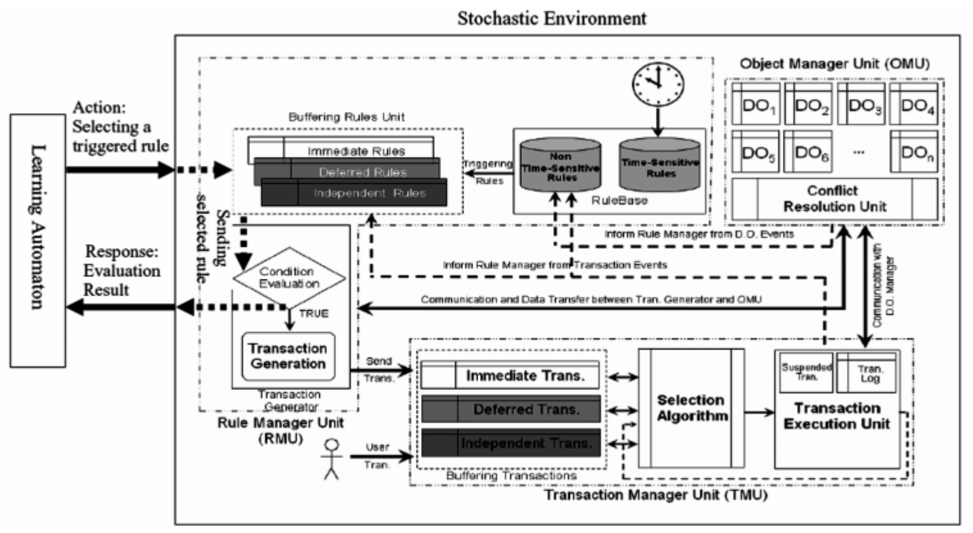
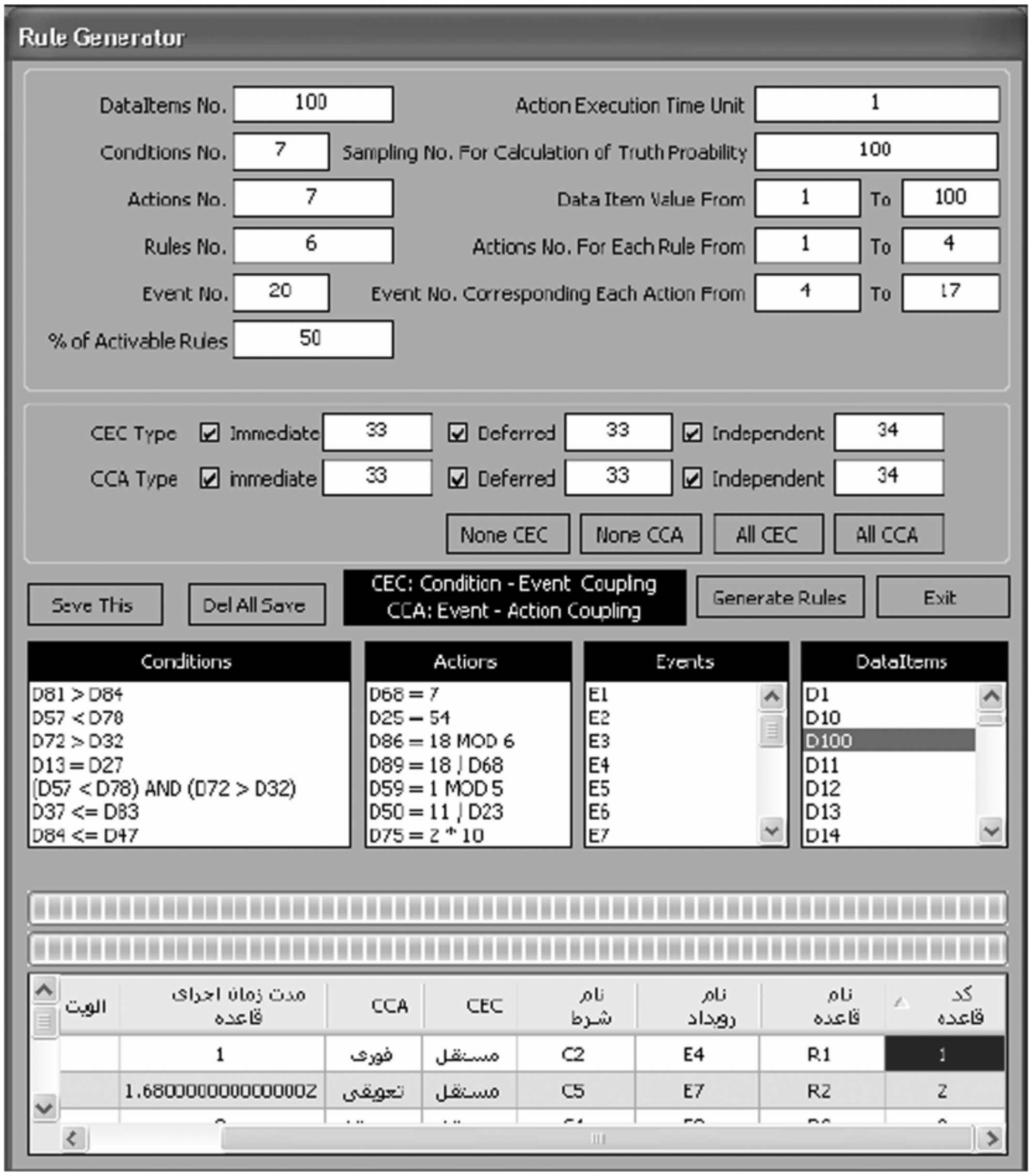


Fig. 3: The Active Database System Simulator Architecture

The ADSS has three-tier architecture: “object manager unit”, “rule manager unit” and transaction manager unit (16). Experiments are performed in three modes: “Deferred mode”, “Immediate mode”, and “Composite mode”. In the first mode system uses rules only in deferred mode. In the second mode, system uses rules only in immediate mode and ultimately in the third mode, system uses rules in all immediate, deferred, and independent modes (1).

In section of designing and implementation of ADSS, we added some features in order to simulate the behavior of active database management system, more completely and subsequently evaluate and compare rule scheduling approaches more exactly. Among them, we added two compilers to existing system in order to compile conditions and instructions. Therefore we can use real instructions and conditional statements in generating process of actions and conditions of active rules. So, we bring simulator and real environment close together as much as possible through compiling conditions and instructions, really, at run time.

In the previous version of ADSS, we had generated and used conditions and instructions virtually (16). But in the new version of ADSS, we considered and implemented the necessary parameters to cover all states and properties of active rules in the real systems. Figure 4 shows UI of active rule generator with mentioned parameters. As you watch in “CEC Type” and “CCA Type” sections of this UI, we can define what percentage of event-condition and conditionaction couplings of active rules are respectively, immediate, deferred, and independent. Moreover, the result of execution of program to generate some active rules has been shown.


Rule Generator

DataItems No.	100	Action Execution Time Unit	1		
Condtions No.	7	Sampling No. For Calculation of Truth Proability	100		
Actions No.	7	Data Item Value From	1	To	100
Rules No.	6	Actions No. For Each Rule From	1	To	4
Event No.	20	Event No. Corresponding Each Action From	4	To	17
% of Activable Rules	50				

CEC Type	✓ Immediate	33	✓ Deferred	33	✓ Independent	34
CCA Type	✓ immediate	33	✓ Deferred	33	✓ Independent	34

None CEC | None CCA | All CEC | All CCA

Seve This | Del All Save

CEC: Condition - Event Coupling
CCA: Event - Action Coupling

Generate Rules | Exit

Conditions	Actions	Events	DataItems
D81 > D84	D68 = 7	E1	D1
D57 < D78	D25 = 54	E2	D10
D72 > D32	D86 = 18 MOD 6	E3	D100
D13 = D27	D89 = 18 / D68	E4	D11
(D57 < D78) AND (D72 > D32)	D59 = 1 MOD 5	E5	D12
D37 <= D83	D50 = 11 / D23	E6	D13
D84 <= D47	D75 = 2 * 10	E7	D14

الویت	مدت زمان اجرای قاعده	CCA	CEC	نام شرط	نام رویداد	نام قاعده	کد قاعده
	1	فوری	مستقل	C2	E4	R1	1
	1.6800000000000002	تعویقی	مستقل	C5	E7	R2	2

Fig. 4: UI of Active Rules Generator

Results of experiments in deferred, immediate and composite modes are shown in Tables 6, 7, 8, respectively. The content of each cell shows the rank of corresponding scheduling method according to corresponding evaluation criteria. The final rank of approaches was calculated according to relation (5). Here is an inversed relation between final rank of an approach and the corresponding value of $S(k)$.

Table 6: Results of Simulation of Available Rule Scheduling Approaches in Deferred Mode

Evaluation Criteria Methods	*Average Response Time*	*Response Time Standard Deviation*	*Throughput*	*Time Overhead Per Transaction*	*CPU Utilization*
Random	4	5	2	1	3
Static Priority	4	5	2	1	3
FCFS	4	4	2	1	3
EDF_{PD}	4	4	2	1	3
EDF_{DIV}	4	7	3	1	3
EDF_{SL}	4	6	4	1	1
E_x-SJF_{EXA}	3	6	3	1	2
E_x-SJF_{PRO}	3	3	3	1	2
E_x-SJF_{PRO}-V.1.8	2	2	2	1	2
E_x-SJF_{PRO}-V.2.8	1	1	1	1	2

Table 7: Results of Simulation of Available Rule Scheduling Approaches in Immediate Mode

Evaluation Criteria Methods	*Average Response Time*	*Response Time Standard Deviation*	*Throughput*	*Time Overhead Per Transaction*	*CPU Utilization*
Random	4	5	2	3	3
Static Priority	4	5	2	3	3
FCFS	4	5	2	1	1
EDF_{PD}	3	4	1	1	2
EDF_{DIV}	3	4	1	1	2
EDF_{SL}	2	3	4	2	2
E_x-SJF_{EXA}	3	4	5	3	2
E_x-SJF_{PRO}	2	3	4	2	2
E_x-SJF_{PRO}-V.1.8	1	2	3	3	2
E_x-SJF_{PRO}-V.2.8	1	1	3	2	2

Table 8: Results of Simulation of Available Rule Scheduling Approaches in Composite Mode

Evaluation Criteria Methods	*Average Response Time*	*Response Time Standard Deviation*	*Throughput*	*Time Overhead Per Transaction*	*CPU Utilization*
Random	6	5	1	1	1
Static Priority	5	4	4	2	1
FCFS	5	4	4	2	1
EDF_{PD}	4	3	5	2	1
EDF_{DIV}	4	3	5	2	1
EDF_{SL}	4	6	6	2	1
E_x-SJF_{EXA}	3	3	3	2	1
E_x-SJF_{PRO}	3	3	3	2	1
E_x-SJF_{PRO}-V.1.8	2	2	2	2	1
E_x-SJF_{PRO}-V.2.8	1	1	1	2	1

$$S(k) = \sum_{j=1}^{3}\sum_{i=1}^{5} W_i * Rnk_{i,j}(k) \qquad \dots(5)$$

$k \in$ Rule Scheduling Approaches Set

W_i is the weight of ith evaluation metric

j = 1, 2 and 3 indicates deferred, immediate and composite state, respectively

$Rnk_{i,j}(k)$ is the rank of approach k in jth state based on ith evaluation metric

$S(k)$ is the total score of approach k among all approaches

Results of experiments show that E_x-SJF$_{PRO}$-V.2.8 has generally the most positive impact on performance (Average Response Time, Response Time Standard Deviation, and Throughput) and efficiency (Time Overhead per Transaction and CPU Utilization) of ADBS. In other words, rule execution time is calculated in E_x-SJF$_{PRO}$-V.2.8 approach (by adding an estimation module) more precise than last versions of E_x-SJF approach and this subject finally leads to improvement of rule scheduling process. Process of estimation of rule execution probability doesn’t impose computational overhead on system so doesn’t have negative impact on Time Overhead per Transaction and CPU Utilization (10).

According to the obtained results, we are going to present a new version of E_x-SJF approach in the next section by applying a more precise technique of estimation of condition correctness probability of triggered rules to improve scheduling process more than this.

4. E_X-SJF$_{EsTLA}$1 AS NEW PROPOSED RULE SCHEDULING APPROACH

This approach is designed based on estimation of rule execution probability like E_X-SJF$_{PRO}$-V.1.8 and E_X-SJF$_{PRO}$-V.2.8 approaches. But in new approach we have attempted to improve the rule scheduling based on improvement of estimation of condition correctness probability of triggered rules using learning automaton. As we have already mentioned, processing cycle of active rules consists two scheduling phases including: 1. triggered rules scheduling to evaluate their condition parts, and 2. ready-to-execute rules scheduling to execute their action parts. All approaches which have already introduced, have the same mechanism for both mentioned scheduling phases. For example, in both scheduling phases of E_X-SJF$_{PRO}$-V.2.8 approach, active rules are scheduled based on their execution time. But in the new approach, scheduling mechanisms of triggered rules and ready-to-execute rules are different from each other. In other words, the new approach schedules ready-to-execute rules like E_X-SJF$_{PRO}$-V.2.8 approach. But it has different mechanism for scheduling triggered rules based on learning automaton. E_X-SJF$_{EsTLA}$ schedules triggered rules based on correctness probability of their conditions using Learning automaton. Learning automaton estimates the correctness probability of condition parts of triggered rules during the execution of active database system. But now, we are going to define learning automaton and related concepts, before description of its position and function in triggered rules scheduling.

1 E_xSJF (based on)-Estimation-(using)-Learning Automata

4.1. Learning Automaton

Learning automata are adaptive decision-making devices operating on unknown random environments. A learning automaton has a finite set of actions and each action has a certain probability (unknown to the automaton) of getting rewarded by the environment of the automaton. The aim is to learn to choose the optimal (i.e. the action with the highest probability of being rewarded) through repeated interaction on the system. If the learning algorithm is chosen properly, then the iterative process of interacting on the environment can be made to result in selection of the optimal actions. Figure 5 shows interactions of the automaton and the environment (9) · (15).

An environment can be described by triple $E \equiv (V, \beta, c)$, where $V = (\alpha_1, ..., \alpha_r)$, shows a set of inputs, $\beta = (\beta_1, ..., \beta_m)$, shows a set of outputs, and $c = (c_1, ..., c_r)$, is set of probability distributions over β. If the response of the environment takes binary values, learning automata model is P-model where $\beta_1 = 1$ and $\beta_2 = 0$ are considered as penalty and reward, respectively and if it takes finite output set with more than two elements that take values in the interval (0, 1), such a model is referred to as Q-model, and when the output of the environment is a continuous variable in the interval (0, 1), it is refer to as S-model. Learning automata can be classified into two main families: fixed structure learning automata and variable structure learning automata (VSLA).

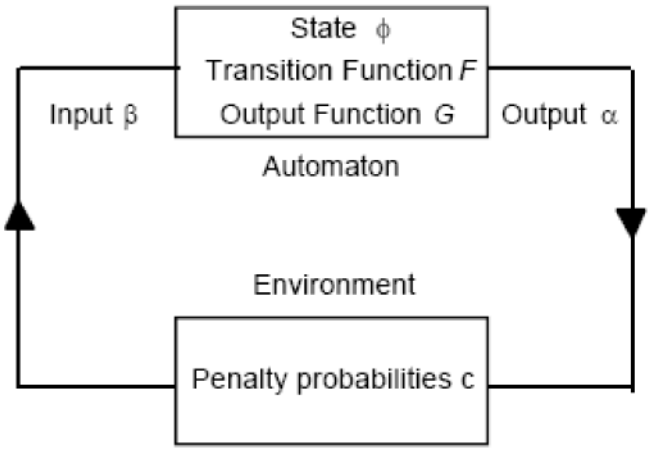


Fig. 5: The Automaton and the Environment

A fixed structure learning automata can be represented by a tuple (V, β, F, G, ϕ) where $V = (\alpha_1, ..., \alpha_r)$, is an action set of automaton, $\beta = (\beta_1, ..., \beta_m)$, is an environment response set, $\phi \equiv (\phi_1, \phi_2, ..., \phi_s)$, is set of internal states, $F: \phi \times \beta \rightarrow \phi$, is state transition function that determines the state of automaton at instant $n + 1$ from its state and inputs at instant n, and $G: \phi \rightarrow \alpha$, is output function, which determines the output of automaton at instant n from its current state and input.

A VLSA is a quintuple (V, β, p, T), where $V = (\alpha_1, ..., \alpha_r)$, is an action set of automaton, $\beta = (\beta_1, ..., \beta_m)$, is an environment response set, and the probability set $P = (P_1, ..., P_r)$ contains r probabilities, each being the probability of performing every action in the current internal automaton state. The function of T is the reinforcement algorithm, which modifies the action probability vector

P with respect to the performed action and received response. Selection probability vector of actions at instant *n* is as follow [relation (6)]:

$$P(n) \equiv [p_1(n), p_2(n), \ldots, p_r(n)]$$

$$\sum_{i=1}^{r} p_i(n) = 1, \ \forall n, p_i(n) = \text{Prob}\,[\alpha(n) = \alpha_i] \qquad \ldots (6)$$

Consider a variable structure automaton with *r* actions in a stationary environment (where the penalty probabilities c_i are constant) with $\beta = (0, 1)$. The general scheme for updating action probabilities is as follows:

If $\alpha(n) = \alpha_i$

(*a*) When $\beta = 0$; favorable response:

$$p_i(n+1) = p_i(n) + \sum_{\substack{j=1 \\ j \neq i}}^{r} f_j[p_j(n)]$$

$$p_j(n+1) = p_j(n) - f_j[p_j(n)] \qquad \forall j, \quad j \neq i$$

(*b*) When $\beta = 1$; unfavorable response:

$$p_i(n+1) = p_i(n) - \sum_{\substack{j=1 \\ j \neq i}}^{r} g_j[p_j(n)]$$

$$p_j(n+1) = p_j(n) + g_j[p_j(n)] \qquad \forall j, \quad j \neq i$$

Where f_j and g_j are nonnegative functions named reward and penalty functions respectively. These functions are defined in a linear enforcement learning algorithm as follows:

$$f_j[p_j(n)] = ap_j(n) \qquad 0 < a < 1$$

$$g_j[p_j(n)] = \frac{b}{r-1} - bp_j(n) \qquad 0 \leq b < 1$$

Where *r*, is the number of automaton's actions, *a*, is the reward parameter, and *b*, is the penalty parameter. So the general scheme for learning algorithm is as follows:

If $\alpha(n) = \alpha_i$

(*a*) When $\beta = 0$; favorable response:

$$p_i(n+1) = p_i(n) + \alpha[1 - p_i(n)]$$

$$p_j(n+1) = (1-a)\,p_j(n) \qquad \forall j, \quad j \neq i$$

(*b*) When $\beta = 1$; unfavorable response:

$$p_i(n+1) = (1-b)\,p_i(n)$$

$$p_j(n+1) = \frac{b}{r-1} + (1-b)\,p_j(n) \qquad \forall j, \quad j \neq i$$

If the learning parameters a and b are equal, learning automaton is called L_{RP} (Linear Reward-Penalty), if $b = 0$, learning automaton is called L_{RI} (Linear Reward Inaction), and if $b << a$ learning automaton is called $L_{R\varepsilon P}$ (Linear Reward Epsilon Penalty).

4.2. How does LA Help us to Improve Rules Scheduling Process?

In the new approach, we schedule triggered rules based on correctness probability of their conditions using a L_{RI} with variable structure. This LA should update the selecting probability of its actions (triggered rules) according to responses received from the environment (correctness or incorrectness of conditions) that eventually leads to improve the performance of active rule scheduling process based on before mentioned evaluation criteria. The position of this automaton in the architecture of ADSS has been shown in Fig. 3.

Moreover, the model of this LA is P in which the input from the environment can take only one of two values, 0 or 1. The response value of 1 corresponds to an "unfavorable" (failure, penalty) response, and occurs when the condition of selected triggered rule is false, while output of 0 means the action is favorable and occurs when the condition of selected triggered rule is true. LA should, in each instance, select one of triggered rules for condition evaluation. The number of triggered rules in each instance is variable. Consequently the number of automaton's actions are variable too. So it is necessary to do some processes on selection probabilities of automaton's actions, whenever the number of triggered rules changes (6). Set of actions at instant k is shown by $V(k)$ which is a sub set of V. It is mentionable that selection probability of each action at instant k, is proportionate to correctness probability of condition section of corresponding rule at instant k {$p_i(k) = \text{prob}\,[\alpha(k) = \alpha_i] \approx$ Correctness probability of rule R_i's condition at instant k}.

As we have also mentioned before, in the new approach, at first, condition correctness probability of rules are calculated based on the technique, presented in E_x-SJF_PRO-V.2.8 approach. So on the assumption that $k(0)$, shows the sum of condition correctness probability of rules at instant 0, according to the relation (6), selection probability of each action at first in scale 1 is calculated as follow:

$$p_i(0) = (\text{Correctness probability of rule } R_i\text{'s condition at instant } 0)/k(0)$$

At instant k, on the assumption that the actions belonging to $V(k)$ are active, selection probabilities of actions are updated as follow [relation (7), (8), (9)]:

$$K(k) = \sum_{\substack{q \\ \alpha_q \in v(k)}} p_q(k) \quad \text{... (7)}$$

(a) Favorable response from environment in response of

$$\alpha_q \in V(k) \quad \text{... (8)}$$

$$p_q(k+1) = p_q(k) - ap_q(k) + ak(k)\ \alpha_q \in V(k)$$

$$p_j(k+1) = p_j(k) - ap_j(k)\ \ \forall j, j \neq q, \alpha_j \in v(k)$$

(b) Unfavorable response from environment in response of

$$\alpha_q \in V(k): \quad \text{... (9)}$$

$$p_q(k+1) = p_q(k)$$

$$p_j(k+1) = p_j(k) \qquad \forall j, \quad j \neq q$$

(*c*) For each $\alpha_i \notin V(k)$:

$$p_i(k+1) = p_i(k)$$

Figure 6 shows updating cycle of the condition correctness probability of triggered rules using LA. It is mentionable that this cycle do its activities, if there are at least two triggered rules. But if there is only one triggered rule, that rule is selected for condition evaluation, without changing of rules selection probability.

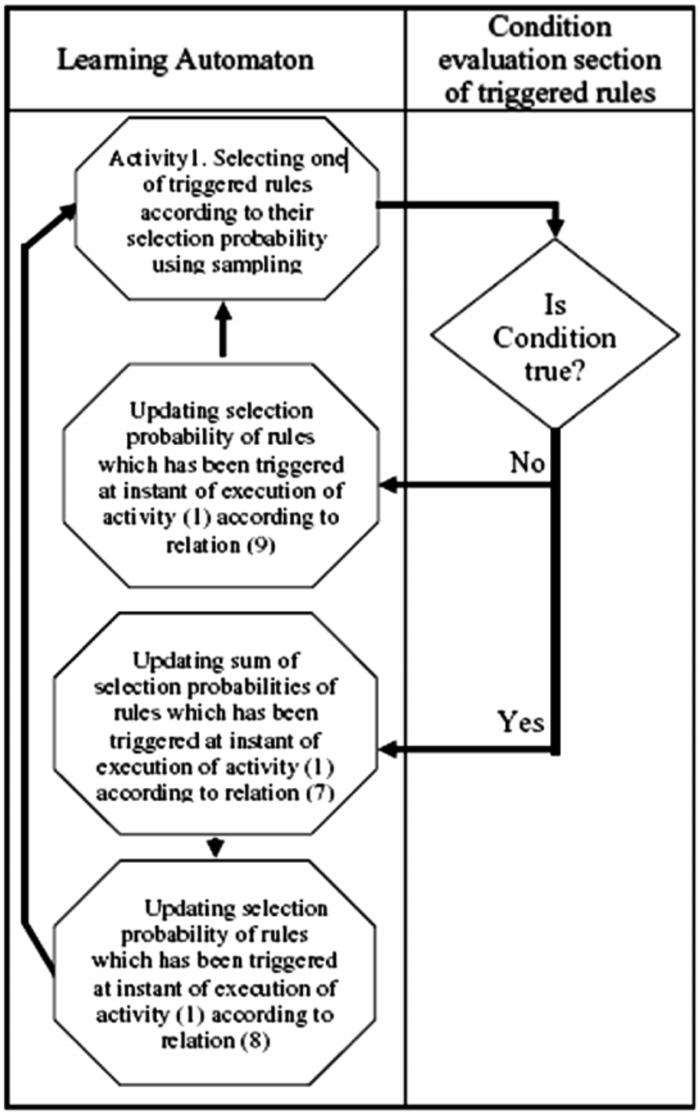

Learning Automaton
Condition evaluation section of triggered rules
Activity1. Selecting one of triggered rules according to their selection probability using sampling
Is Condition true?
No
Yes
Updating selection probability of rules which has been triggered at instant of execution of activity (1) according to relation (9)
Updating sum of selection probabilities of rules which has been triggered at instant of execution of activity (1) according to relation (7)
Updating selection probability of rules which has been triggered at instant of execution of activity (1) according to relation (8)

Fig. 6: Updating Cycle of the Condition Correctness Probability of Triggered Rules Using LA

After that the system starts to work, in specific intervals (δt_i), condition correctness probability (execution probability) of active rules are updated according to the same technique mentioned in Section 2.6.4. Moreover, at the first of each execution interval (σt_{i+1}) of system, actions selection probabilities of learning automaton are updated proportionate to conditions correctness probabilities of active rules and actions selection probabilities of learning automaton at the end of last interval (σt_i) according to relation (10): selection probability of ith action (rule R_i) calculated by the technique embedded in E_X-SJF_{PRO}-V.2.8 approach at the end of σt_k in scale 1: $p_{R_i,1}(\sigma t_k)$ selection probability of ith action (rule R_i) calculated by learning automaton at the end of σt_k: $p_{R_i,2}(\sigma t_k)$

$$\phi(t) = \text{Min}\left({}^{k}/_{k'}, 1\right) \quad \ni \quad k' \geq 25$$

selection probability of ith action (rule R_i) by learning automaton in the beginning of σt_{k+1} t :

$$p_{R_i}\left(\sigma t_{k+1}\right) = \frac{\phi(t) * p_{R_i},2\left(\sigma t_k\right) + p_{R_i,1}\left(\sigma t_k\right)}{\phi(t)+1} \qquad \text{... (10)}$$

After implementation of new approach in the framework mentioned in Section 3, we evaluate its operation. Table 9 shows the percentage of optimizing the rule scheduling in E_X-SJF_{EsTLA} in compare with E_x-SJF_{PRO}-V.2.8 (the most effective existing rule scheduling approach), based on three evaluated parameters: Average Response Time, Response Time Standard Deviation, and Throughput.

Table 9: Percentage of Rules Scheduling Improvement in E_X-SJF_{EsTLA} in Compare with E_x-SJF_{PRO}-V.2.8

Evaluation Criteria Mode	*Average Response Time*	*Response Time Standard Deviation*	*Throughput*
Immediate	8.6%	6.4%	7%
Deferred	9.8%	11.9%	6.7%
Composite	7.8%	6.9%	12.3%

Results of experiments show that by adding a learning automaton, rule scheduling process, is improved. In other words triggered rules scheduling based on their condition correctness probability (not based on their execution time) leads to improving the Average Response Time, Response Time Variance and Throughput of ADBS. The new technique dose not impose any overhead on the ADBS. So the Time Overhead per Transaction and CPU Utilization of E_X-SJF_{EsTLA} and E_x-SJF_{PRO}-V.2.8 are equal.

Just as we mentioned before, in new approach, i.e. E_X-SJF_{EsTLA}, rule scheduling process is improved through improving triggered rule scheduling. We expect logically that such improvement occurs. Because just those triggered rules are executed that their conditions are true at the instant of evaluation, and others are sent out of processing cycle. But the time spent for selection and evaluation of this type of triggered rules is counted as wasted time of system's function that effect directly on average response time, throughput, and indirectly on response time standard deviation. It is evident that the less priority of triggered rules which sent out of processing cycle, the more performance of system from before mentioned criteria viewpoint. We reach this goal by scheduling triggered rules based on their condition correctness probability. A mechanism that estimate condition correctness probability of rules more exactly at run time, help us more to reach this goal. We consider the above points in the new approach. Computational overhead in this approach regarding E_X-SJF_{PRO}-V.2.8 is just in updating of condition correctness probability of some triggered rules according to relation (8) which is insignificant

5. CONCLUSIONS

In this paper, we first defined Active Database System and rules processing cycle. Then we expressed the position and importance of rules scheduling process. Afterwards existing rule scheduling approaches were introduced. Then we showed the results of comparison and evaluation of these approaches which had already been obtained by using a defined framework based on five evaluation criteria. Then, to improve the most effective existing rule scheduling approach, we developed a new algorithm for estimation of rule execution probability using learning automata and developed a new rule scheduling approach based on this algorithm. The new approach was called E_X-SJF_{EsTLA}. Then we compare and evaluate E_X-SJF_{EsTLA} and the most effective existing rule scheduling approach, i.e. E_X-SJFPRO-V.2.8 in the mentioned framework with each other. Results of experiments show that E_X-SJF_{EsTLA} has the positive impact on Average Response Time, Response Time Standard Deviation, and Throughput of ADBS and doesn't have any negative impact on Time Overhead per Transaction and CPU Utilization Figs 7 to 10 show some evaluation diagrams of rule scheduling approaches in various states based on mentioned criteria.

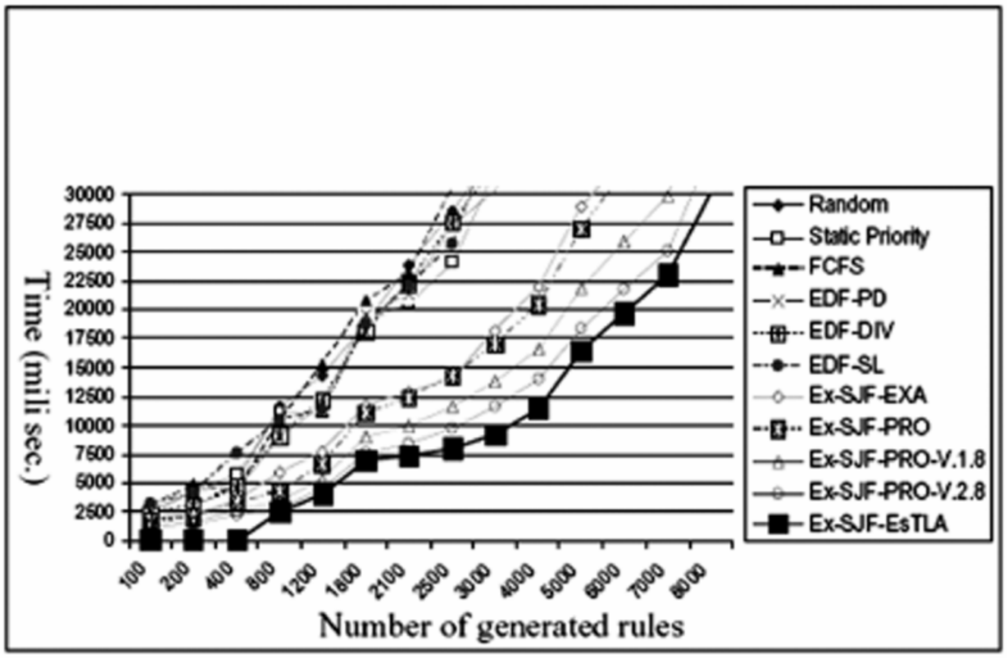


Fig. 7: Average Response Time Diagram of Rule Scheduling Approaches in Deferred Mode

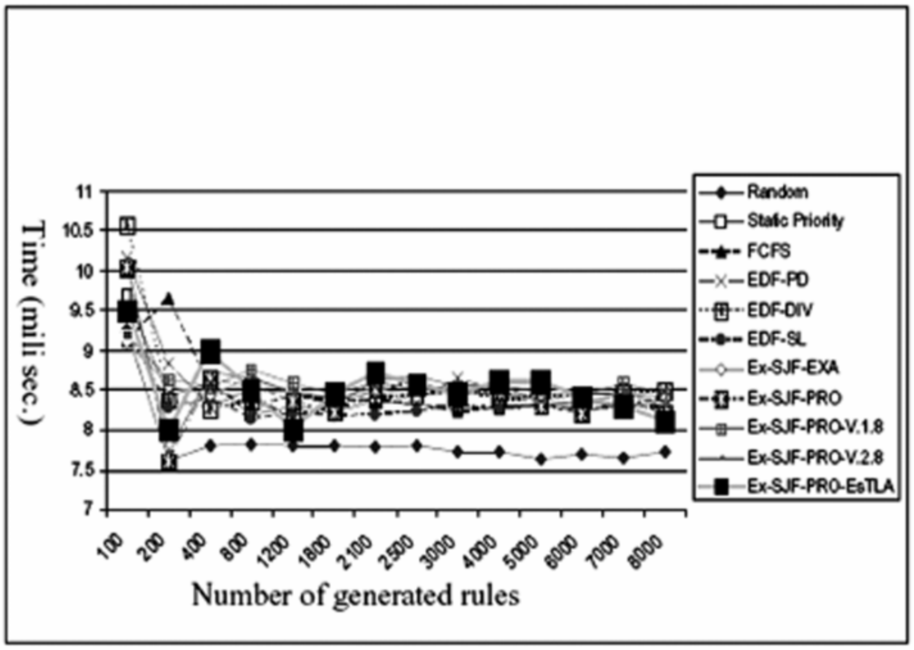


Fig. 8: Throughput Diagram of Rule Scheduling Approaches in Immediate Mode

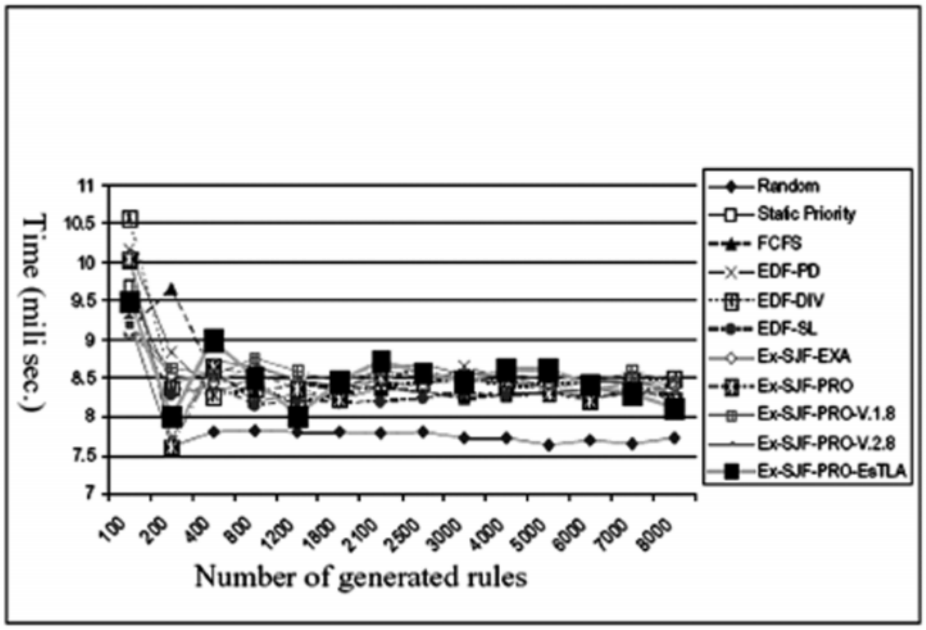


Fig. 9: Time Overhead per Transaction Diagram of Rule Scheduling Approaches in Composite Mode

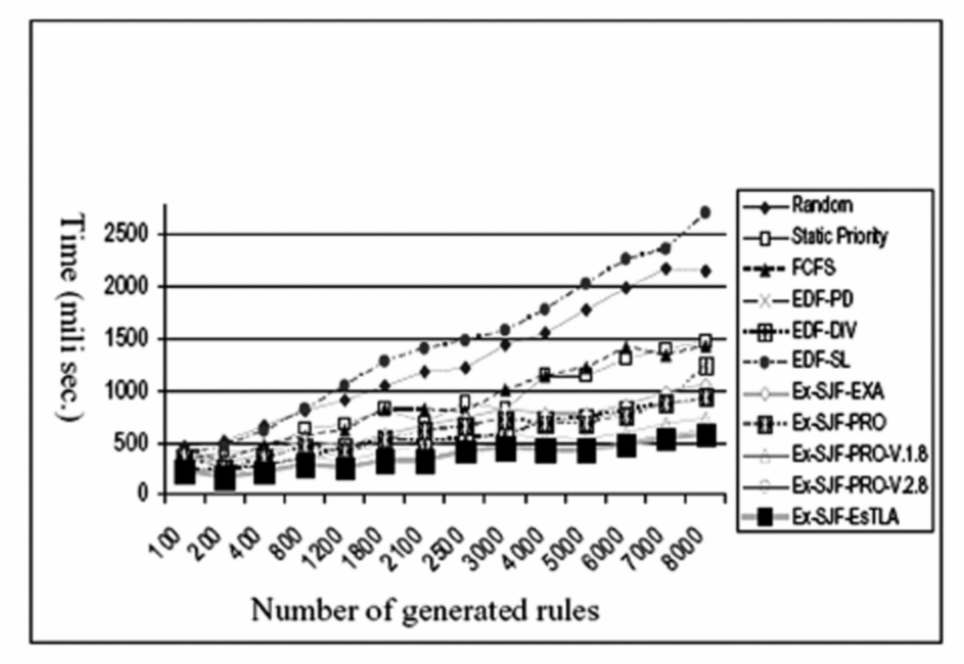


Fig. 10: Response Time Standard Deviation Diagram of Rule Scheduling Approaches in Composite Mode

REFERENCES

[1] Vadua, A.; "Rule Development for Active Database", *PhD Thesis, CS Department*, University of Zurich, 1999.

[2] Gatziu, S.; "Events in an Active Object-Oriented Database System", *PhD Thesis*, University of Zurich, 1994.

[3] Rasoolzadegan, A.; "A New Rule Scheduling Approach based on Estimation of Rule Execution Probability in Active Database System", *MSc Thesis*, Amirkabir University of Technology (Tehran PolyTechnic), 2007.

[4] Alesheykh, R.; "An Effective Rule Selection Approach in Active Database Systems", *MSc Thesis*, Amirkabir University of Technology (Tehran PolyTechnic), 2005.

[5] Ceri, S.; Gennaro, C.; Paraboschi, S.; Serazzi, G.; "Effective Scheduling of Detached Rules in Active Database", *IEEE Transaction on Knowledge and Data Engineering*, **15**, No. 1, 2003.

[6] Thathachar, M.A.L.; Sastry, P.S.; "Varieties of Learning Automata: An Overview", IEEE Transaction on Systems, Man and Cybernetics-Part B: Cybernetics, **32**, No. 6, pp. 711–722, 2002.

[7] Sivasankaran, R.M.; Stankovic, J.A.; Towsley, D.; Purimetla, B.; Ramamritham, K.; "Priority Assignment in Real-Time Active Databases", *The International Journal on Very Large Data Bases*, **5**, No. 1, January 1996.

[8] Hanson, E.N.; Widom, J.; "An Overview of Production Rules in Database Systems". In the Knowledge Engineering Review, **8**, No. 2, pp. 121–143, 1993.

[9] Thathachar, M.A.L.; Harita, B.R.; "Learning Automata with Changing Number of Actions", IEEE Transaction on Systems, Man and Cybernetics, **SMC-17**, No. 6, pp. 1095–1100, 1987.

[10] Rasoolzadegan, A.; Alesheykh, R.; Abdollahzadeh, A.; "A New Approach for Event Triggering Probability Estimation in Active Database Systems to Rule Scheduling Improvement", *2nd IEEE International Conference on Information and Communication Technologies: From Theory To Applications*, Damascus, Syria , April 24 – 28, 2006.

[11] Rasoolzadegan, A.; Alesheykh, R.; Abdollahzadeh, A.; "Measuring Evaluation Parameters in Benchmarking Rule Scheduling Methods in Active Database Systems", *The IEEE International Conference on Computer and Communication Engineering*, Kuala Lumpur, Malaysia, 2006.

[12] Vaduva, S. Gatziu; Dittrich, K.R.; "Investigating Termination in Active Database Systems with Expressive Rule Languages", *In Proceedings of the 3rd International Workshop on Rules In Database Systems*, pp. 149 – 164, Skovde (Sweden), 1997.

[13] Potaminsto, S.; Stonebraker' M.; "The POSTGRES Rule System", *in Active Database Systems: Triggers and Rules for Advanced Systems*, Morgann Kaufmann Publishers, Sanfrancisco, CA, 1996.

[14] Stankovic, J.; Natale, S.M.D.; Buttazo, G.C.; "Implications of Classical Scheduling Results for Real-Time Systems", IEEE Computer Society Press, Los Alamitos, CA, 1995.

[15] Narendra, K.S.; Thathachar, L.; *Learning Automata: An Introduction*, Prentice Hall, 1989.

[16] Rasoolzadegan, A.; Abdollahzadeh; "ADSS: Active Database System Simulator to Compare and Evaluate Rule Scheduling Methods", *Technical Report-CE/TR.RP/85/01, Intelligent Systems Laboratory*, IT and Computer Engineering Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, 2006.

[17] Geppert, A.; Gatziu, S.; Dittrich, K.R.; Fritschi, H.; Vaduva, A.; "Architecture and Implementation of the Active Object-oriented Database Management System SAMOS", *Technical Report 95.29, CS Department*, University of Zurich, 1995.