A New Approach to Active Rule Scheduling

Abbas Rasoolzadegan[1]*; Rohollah Alesheykh*[2]*; M.R. Meybodi*[3]

[1]*Faculty of Engineering, Ferdowsi University of Mashhad, Mashhad, Iran, rasoolzadegan@um.ac.ir*

[2]*Faculty of Engineering, Payame Noor University (PNU), Iran, alesheykh@pnu.ac.ir*

[3]*Department of Computer Engineering and IT, Amirkabir University of Technology, Iran, meybodi@aut.ac.ir*

Abstract

Active database systems (ADSs) react automatically to the occurrence of predefined events by defining a set of active rules. One of the main modules of an ADS is the rule scheduler, which has a significant impact on the effectiveness and efficiency of ADSs. During the rule scheduling process, the rule scheduler is responsible for choosing one of the activated or ready-to-be-executed rules to evaluate its condition section or execute its action section, respectively. This process continues until there is no rule to be evaluated or executed. In this research, we evaluate and compare existing rule scheduling approaches in a laboratory environment based on a three-tier architecture. There are criteria used for the evaluation and comparison of rule scheduling approaches: Average Response Time, Throughput, Response Time Standard Deviation, Time Overhead per Transaction, and CPU Utilization. The three first criteria are used to evaluate the effectiveness, and the latter two criteria are used to evaluate the efficiency of rule scheduling approaches. In this paper, a new approach, referred to as E_X-SJF_{EsTLA}, is proposed to improve the rule scheduling process, using a learning automaton. In our laboratory environment, E_X-SJF_{EsTLA} is compared with those rule scheduling approaches that are unconstrained as E_X-SJF_{EsTLA} is. Unconstrained scheduling approaches serially schedule the rules that do not have any priorities or deadlines. The results of experiments revealed that the proposed approach improved the rule scheduling process according to the evaluation criteria.

Keywords: *Active database management systems, probability estimation, active rule scheduling, learning automata.*

1. Introduction

Common (Traditional) database systems are often of a passive nature. This means that operations such as querying, updating, inserting, deleting, and reporting are performed only in the event that users request them. Database Management Systems, abbreviated to DBMSs, cannot automatically react when various events occur. Many applications such as real-time expert systems [32-33], warehousing programs, the automation of processes, and complex financial calculations in stock markets need automatic control for handling events that have occurred. Active Database Systems [19], abbreviated to ADSs, meet the requirements of such applications by defining ECA (Event-Condition-Action) rules, referred to as active rules. An ECA rule has three main sections: Event, Condition, and Action. An ECA rule for the context of buying and selling stocks [12] is defined as follows:

DEFINE *LowRisk*	*This active rule is named "LowRisk"*
ON *Stock.UpdatePrice*	*Event section*
IF *(Stock.policy = Low_risk) and (Stock.price< Stock.initprice * e) ;(0<e<1)*	*Condition section*
DO *Stock.Buy*	*Action section*

In addition to event, condition, and action components, each active rule has two other features: event-condition coupling mode and condition-action coupling mode. When an event E_1 occurs, the event-condition coupling mode of each rule triggered by E_1 determines the time when the condition section of the rule should be evaluated. There are three choices for the event-condition coupling mode of each active rule: *immediate*, *deferred*, and *independent*. If the event-condition coupling mode of an activated rule is immediate, the condition of the rule should be evaluated immediately. If the event-condition coupling mode of an activated rule is deferred, the evaluation of the condition of the rule should be deferred until the execution of the action section of the rule that is being executed is terminated. If the event-condition coupling mode of an activated rule is independent, the condition section of the rule is evaluated only after the condition sections of all activated rules with the immediate and deferred modes have been evaluated. For example, suppose there are some activated rules, waiting to be evaluated. The condition sections of the rules with the independent event-condition coupling mode are not evaluated until the condition sections of the other rules have been evaluated. If the condition section of an activated rule is evaluated to true, the condition-action coupling mode of the rule determines the time when its action section should be executed.

Similarly, there are three choices for the condition-action coupling mode of each rule: immediate, deferred, and independent. If the condition-action coupling mode of a ready-to-be-executed rule (a rule whose condition-section has been evaluated to true) is immediate, its action section must be executed immediately. If the condition-action coupling mode of a ready-to-be-executed rule is deferred, the execution of its action section should be deferred until the execution of the action section of the rule that is being executed is terminated. If the condition-action coupling mode of a ready-to-be-executed rule is independent, the action section of the rule is executed only after the action sections of ready-to-be-executed rules with immediate and deferred modes have been executed.

Events can be classified as primitive or composite events. Primitive events refer to elementary occurrences which are predefined in the system. Primitive events are typically further categorized as database events, temporal events, transaction events, etc. Database events are related to database operations and are further classified into Insert, Delete, and Update. A temporal event can be an absolute point in time, defined by the system clock (e.g., 9:00:00 a.m., April 10, 1988), relative (30 seconds after event *A* occurred), or periodic (every day at midnight). As the name implies, transaction events are kinds of events related to transactions; for example, the beginning and end of a transaction are events signaled at the beginning and end of the transaction. A composite event is a set of primitive events which are combined using event operators (such as "and", "or," and "not") to form a new event specification.

An ADS processes its active rules to automatically control various events. The rules processing cycle, elaborated on in section 2, consists of the following steps [27]:

1) **Event Signaling:** When a primitive event occurs, the primitive event detector signals it. In addition, the composite event detector considers the occurring primitive events to investigate the occurrence of composite events.
2) **Rule Triggering:** After an event is signaled, those ECA rules that correspond to the signaled event are activated. One instance of each activated rule is created, which includes some additional information (such as a timestamp, a deadline, and an execution time) depending on the scheduling mechanism used by the rule scheduler. These instances are buffered to be used in the next step.
3) **Condition Evaluation:** In this step, the rule scheduler sequentially selects the instances created in the previous step, and their conditions are evaluated. If the condition section of an instance is evaluated to true, the instance is added to the ready-to-be-executed buffer.
4) **Transaction Selection:** In this step, the rule scheduler selects the ready-to-be-executed rules in order of priority. A transaction is generated for each ready-to-be-executed rule based on its action section. The transaction is then sent to the execution unit. This step is also called the transaction scheduling phase.
5) **Transaction Execution:** The transactions generated in the previous phase are executed in this step.

ADSs have many applications in controlling and automatic handling of processes in different areas such as the stock exchange organization, portfolio management systems, automatic traffic control systems, all applications with continuous monitoring, and real-time systems as real-time active database systems [19], [20-23]. An ADS plays the role of an automatic controller by defining and executing several active rules. During the running of the system, several activated rules with different event-condition and condition-action coupling modes are waiting to be evaluated or executed.

There is an important question here: why is it important to have an effective and efficient rule scheduling algorithm in ADSs? To answer this question, consider a stock exchange system that uses an ADS. This ADS has tens of thousands of users, and the number of active rules in their rule-bases reaches hundreds of thousands of rules. For example, suppose there are n users in the electronic management system of the stock exchange. Each user has defined a different number ($m_1, \dots, m_n$) of active rules. The definition of these rules is illustrated in Table (16) (see

Appendix). After each update operation (i.e. edit, insert, or delete) in the electronic management system of the stock exchange, thousands of rules are activated. Suppose that in a stock exchange system with tens of thousands of users, n' number of users ($n' < n$ and n' is rather big; for example, n' is about two or three thousands), define some rules triggered by each update on the price of Mercedes Benz. When the price of this stock is updated, in a moment, the corresponding rules are activated. At this moment, the stock price of the Volvo Company may be updated as well. And this may cause thousands of other rules to be activated. Such scenarios may continue. There are several similar scenarios that may happen during the running of the electronic management system of the stock exchange. It is obvious that during the running of these kinds of systems in the real-world, many activated rules wait to be evaluated and executed. And in these situations, even milliseconds matter. If a rule, defined by user X, is not executed at the desired moment, the goal of a user X is not satisfied. This goal may be "buying a stock A" or "selling a stock B," so user X does not obtain his/her expected benefit. Thus, the importance of having an effective and efficient rule scheduling mechanism is quite evident.

So far many rule scheduling mechanisms have been introduced such as Random [8], FCFS (First Come, First Served) [14], and E_x-SJF (Extended Shortest Job First) [4], presented in section 3. There are also some criteria for evaluating the effectiveness (performance) and efficiency of rule scheduling approaches such as Average Response Time, Throughput, and CPU Utilization, presented in section 2.

The scheduling of active rules is one of the main research topics in ADSs [3], [5], [24-29]. There have been several attempts in this area. The focus is mainly on the unconstrained rule scheduling approaches - such as the various versions of E_x-SJF - rather than on those introduced for constrained rule scheduling approaches - such as the "Static Priority" approach. Constrained rule scheduling approaches are used when rules have some restrictions (or constraints) such as deadlines or priorities. For example, in safety-critical and high-integrity systems - such as the automatic control system of an airplane - priorities are assigned to rules according to their importance. Among all activated or ready-to-be-executed rules, the rule that has the highest priority is selected to be evaluated or executed first. The priority of rules is determined by rule developers before the running of the system. In real-time systems, each rule has its own deadline. Therefore, initially, it is necessary to define the deadline of each rule accurately. The deadline of a rule is the greatest amount of time that the execution of the action section of the rule can be postponed after its condition is evaluated to true. In real-time systems, the rule scheduler should always select the rule that has the nearest deadline among all activated or ready-to-be-executed rules. In such systems, the greater the number of missed deadlines, the less effective they become in critical situations. However, in systems where rules do not have any constraints (all kinds of systems and applications except safety-critical, high-integrity, and real-time systems) such as a stock exchange system, unconstrained rule scheduling algorithms serially schedule the rules based on some criteria such as activation time and execution time. Each rule scheduling mechanism suffers from some disadvantages in the scheduling process of active rules. The performance and efficiency of a rule scheduling approach, used in an ADS, directly impact the overall performance and efficiency of the ADS.

This study investigates the advantages and limitations of various rule scheduling approaches in ADSs, according to the literature review. In order to improve the scheduling process of active rules, this work equips the most effective existing unconstrained approach (i.e. E_X-SJF_{PRO}-V.2.8 [3]) with a learning module. The learning module helps to make better estimates of the probability of truth of the condition section of each rule - which, in turn, improves the performance of the rule scheduling process. This work also proposes a new framework for comparing and evaluating existing rule scheduling approaches. This framework consists of an active database system simulator, named ADSS[1] as well as some standard criteria for evaluating rule scheduling approaches. Our new approach, referred to as E_X-SJF_{EsTLA}, is compared with the previously developed unconstrained scheduling approaches according to the mentioned criteria in ADSS. The results of these experiments could show that E_X-SJF_{EsTLA} improves the rule scheduling process.

The rest of this paper is organized as follows: section two presents a set of key terms that are necessary to be able to communicate the scope and contributions of this work. This section also defines the problem to be solved by the proposed approach. In section three, existing rule scheduling approaches are reviewed. Section four introduces the new rule scheduling approach. This section defines learning automata and elaborates on how a learning automaton helps us improve the rule scheduling process. Then in section five, ADSS is explained. This section describes the design of the experiments. Two cases of experiments are designed: the rules used in one of the two

[1] Active Database System Simulator

cases are more highly correlated and more dependent than those of the other. The results of experiments are illustrated, using several tables and charts. The results could show that the proposed approach improves the effectiveness of the rule scheduling process. Finally, section six presents the conclusions and discusses the suggestions for future work.

2. Key Concepts and Definitions

As previously mentioned, the action section of each active rule is executed as a transaction. One of the main characteristics of ADSs, called active work load, is the dynamic activation of rules and subsequently the dynamic generation of transactions during the running of the system. Each transaction causes additional events to occur in the system during its execution, for example, through data manipulation (inserting, updating, deleting, or restoring data) in the database. The new events that have occurred may activate other rules, and subsequently, some new transactions may be added to the rules processing cycle. The execution of each new transaction may trigger other events. The newly triggered events may, in turn, activate further rules. Such an iterative cycle may continue to some levels. Those transactions that lead to generation of additional transactions during their execution are called "parent transactions," and the generated transactions are called "child transactions." Similarly, the rules that can potentially activate other rules are called "parent rules," and the activated rules are named "child rules." Moreover, we formally define four time-related idioms for each transaction: 1) Expected Execution Time (EET), 2) Actual Execution Time (AET), 3) Expected Termination Time (ETT), and 4) Actual Termination Time (ATT). The formal definition of these idioms are depicted in Table (1).

Table1. Formal definition of four time-related idioms

$EET = Expected\ Execution\ Time$ $ETT = Expected\ Termination\ Time$	$AET = Actual\ Execution\ Time$ $ATT = Actual\ Termination\ Time$	$ET(T) = Execution\ Time\ of\ T$
$k = the\ number\ of\ immediate\ child\ transaction\ of\ T$	$n = the\ number\ of\ immediate\ child\ transaction\ of\ T$	
$EET(T) = ET(T) + \sum_{i=1}^{k} EET(T_i^{imm})$ $ETT(T) = S(T) + EET(T)$	$AET(T) = ET(T) + \sum_{i=1}^{k} AET(T_i^{imm}) + \sum_{i=1}^{n} AET(T_i^{def})$ $ATT(T) = S(T) + AET(T)$	

Considering the above-mentioned definitions, we can say that a transaction is expectedly terminated if the execution time of the transaction is terminated, and all of its immediate child transactions are expectedly terminated. A transaction is actually terminated if the execution time of the transaction is terminated, and all of its immediate and deferred child transactions are actually terminated.

2.1. An Example for Active Work Load

Fig. 1 illustrates the dynamic generation of transactions. The mechanism used for scheduling the transactions of Fig. 1 is FCFS, which is discussed in section 3.3. This figure indicates the dynamic generation of transactions up to four levels. In this example, a transaction, called T, has already been generated and is ready to be executed at t_0 $(a(T))$. The execution of T starts at t_2 $(s(T))$. This transaction generates two deferred transactions T_1^{def} and T_2^{def} at t_4 and t_5 and one immediate transaction T_1^{imm} at t_9 during its execution. The execution of T_1^{imm} starts at t_9. During the execution of T_1^{imm}, two deferred transactions T_3^{def} and T_4^{def} are generated. The execution of T_1^{imm} is expectedly terminated at t_{13}. Now the execution of T continues until another immediate transaction T_2^{imm} is generated by T at t_{14}. After the execution of T_2^{imm} is actually terminated at t_{15} (expected termination time and actual termination time of T_2^{imm} are equal because T_2^{imm} dose not generate any other transaction during its life time), the execution of T continues, and one independent transaction T_1^{ind} is generated at t_{16}. Although it was expected that the execution of T be terminated at t_{17} $(ETT(T))$, its execution is actually terminated after the execution of T_1^{def}, T_2^{def}, T_1^{imm} and T_2^{imm} are actually terminated (as mentioned T_2^{imm} is actually terminated at t_{15}, so as soon as T_1^{def}, T_2^{def}, and T_1^{imm} be actually terminated, T is actually terminated too). The execution of T_1^{def} starts at t_{17}, and one immediate transaction T_3^{imm} is generated during its execution. Two deferred and independent transactions T_5^{def} and T_2^{ind} are generated by T_1^{def} at t_{21} and t_{22}, respectively. The execution of T_1^{def} is terminated expectedly at t_{23}. The execution of T_2^{def} starts at t_{23}. This transaction causes two immediate transactions T_4^{imm} and T_5^{imm} to be generated. The execution of T_2^{def} is actually terminated at t_{28}. The execution of T_3^{def} starts at t_{28} and causes two immediate transactions T_6^{imm} and T_7^{imm} to be executed. The execution of T_3^{def} is actually terminated at t_{33}. Now it is time for T_4^{def} and T_5^{def} to be executed. The execution of T_4^{def} starts at t_{33}. One independent transaction T_3^{ind} and one immediate transaction T_8^{imm} are generated during the execution of T_4^{def}. The execution of T_4^{def} and T_8^{imm} are actually terminated at t_{37}. As mentioned T_3^{def} and T_4^{def} are actually terminated at t_{33} and t_{37}, respectively, so their parent transaction, T_1^{imm}, is actually terminated at t_{37}. It is now time for the execution of T_5^{def}. The execution of this transaction and T_1^{def} are actually terminated at t_{39}. Consequently the execution of T is actually terminated at t_{39} ($ATT(T)$).

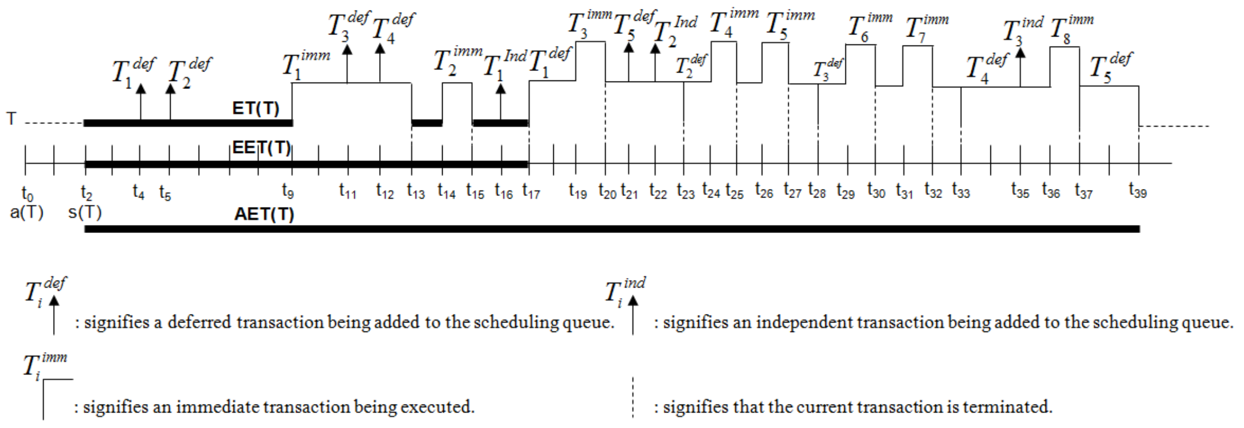

T
ET(T)
EET(T)
AET(T)
a(T)
s(T)
t0 t2 t4 t5 t9 t11 t12 t13 t14 t15 t16 t17 t19 t20 t21 t22 t23 t24 t25 t26 t27 t28 t29 t30 t31 t32 t33 t35 t36 t37 t39
T1 def T2 def T3 def T4 def T1 imm T2 imm T1 Ind T1 def T3 imm T5 def T2 Ind T2 def T4 imm T5 imm T3 def T6 imm T7 imm T4 def T3 ind T8 imm T5 def
: signifies a deferred transaction being added to the scheduling queue.
: signifies an independent transaction being added to the scheduling queue.
: signifies an immediate transaction being executed.
: signifies that the current transaction is terminated.

Fig. 1 The life-time of an active rule

Table (2) presents some information regarding the transactions executed during the life-time of transaction T. The information is about 1) the time when each transaction is activated, starts to be executed, expectedly terminated, and actually terminated 2) the expected and actual execution time of each transaction, and 3) the number of the immediate, deferred, and independent transactions generated during the execution of the parent transactions. As is observed in Table (2), the actual execution time of each parent transaction is directly related to the number and the actual execution time of its deferred and immediate children.

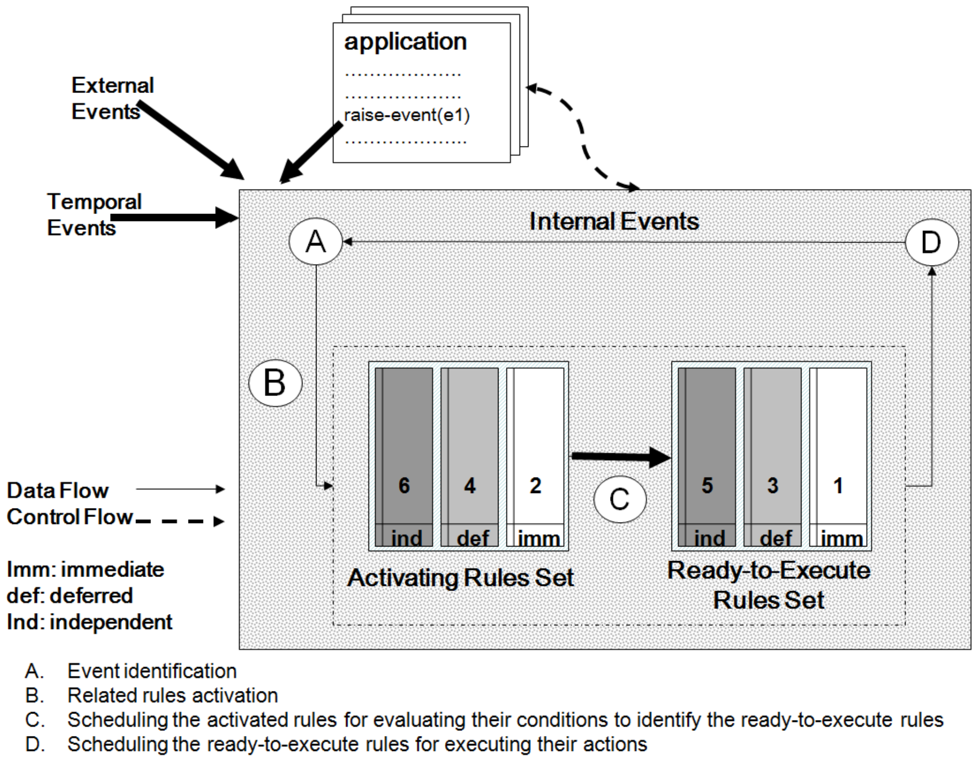
Table2. The number of generated transactions and other information of Fig. 1

	Time (Per Unit Time)						**The number of generated transactions (children)**			
	Activation Time	**Start Time**	**ETT**	**ATT**	**EET**	**AET**	**Independent**	**Deferred**	**Immediate**	**Overall**
T	t_0	t_2	t_{17}	t_{39}	15	37	1	2	2	5
T_1^{imm}	t_9	t_9	t_{13}	t_{37}	4	28	0	2	0	2
T_2^{imm}	t_{14}	t_{14}	t_{15}	t_{15}	1	1	0	0	0	0
T_3^{imm}	t_{19}	t_{19}	t_{20}	t_{20}	1	1	0	0	0	0
T_4^{imm}	t_{24}	t_{24}	t_{25}	t_{25}	1	1	0	0	0	0
T_5^{imm}	t_{26}	t_{26}	t_{27}	t_{27}	1	1	0	0	0	0
T_6^{imm}	t_{29}	t_{29}	t_{30}	t_{30}	1	1	0	0	0	0
T_7^{imm}	t_{31}	t_{31}	t_{32}	t_{32}	1	1	0	0	0	0
T_8^{imm}	t_{36}	t_{36}	t_{37}	t_{37}	1	1	0	0	0	0
T_1^{def}	t_4	t_{17}	t_{23}	t_{39}	6	22	1	1	1	3
T_2^{def}	t_5	t_{23}	t_{28}	t_{28}	5	5	0	0	2	2
T_3^{def}	t_{11}	t_{28}	t_{33}	t_{33}	5	5	0	0	2	2
T_4^{def}	t_{12}	t_{33}	t_{37}	t_{37}	4	4	1	0	1	2
T_5^{def}	t_{21}	t_{37}	t_{39}	t_{39}	2	2	0	0	0	0

2.2. Active Rule Processing Cycle

Fig. 2 illustrates how the rules processing cycle works in an ADS. The application runs until an internal event – such as inserting, deleting, or updating of data - or an external one whose origin is out of the system's scope –such as an increase or decrease of temperature reported by the sensors - occurs at the runtime of the system; then the rule processing unit takes the control of the system (stage A). The rule processing unit activates those rules of the rule-base triggered by the events that have occurred. The activated rules are then placed in one of the lists numbered 2, 4, or 6 according to their event-condition coupling modes (stage B). The condition section of each of the activated rules that are in the immediate, deferred, or independent lists has the first, second, or third priority to be evaluated, respectively. For example, the condition section of none of the rules that are in the deferred or independent lists is evaluated as long as there is an activated rule in the immediate list. In each list (i.e. immediate, deferred, or independent), the priority of a rule for condition evaluation depends on the scheduling mechanism used by the rule scheduler (stage C). If the condition section of a rule holds true at the evaluation time, the rule is placed into one of the lists numbered 1, 3, or 5 according to its condition-action coupling mode. Similarly, the action section of each of the ready-to-be-executed rules that are in the immediate, deferred, or independent lists has the first, second, or third priority to be executed, respectively. The rule scheduler chooses the ready-to-be-executed rule that has the highest priority to execute its action section as a transaction (stage D). New events may occur during the execution of the action section of the rule being executed. The rules triggered by the new events that have occurred are activated and entered into the rules processing cycle. This cycle continues until there are no rules remaining in any of the six lists. Then control is returned to the application.

In general, the rules that are in the lists numbered 1, 2, 3, 4, 5, and 6 have the first to sixth priorities, respectively. For example, the condition section of none of the rules in the "immediate-activated-rules" list (list number 2) will be evaluated as long as there is a rule in the "immediate-ready-to-be-executed-rules" list (list number 1). However, the selection criterion for the rules placed in each list is determined by the scheduling mechanism used. Active rules are scheduled for two purposes: 1) to evaluate the condition section of each rule that is in the "activated-rules" lists (lists number 2, 4, or 6) and 2) to execute the action section of each rule that is in the "ready-to-be-executed-rules" lists (lists number 1, 3, or 5).



application
raise-event(e1)
External Events
Temporal Events
Internal Events
A
B
C
D
6
4
2
ind
def
imm
5
3
1
ind
def
imm
Activating Rules Set
Ready-to-Execute Rules Set
Data Flow
Control Flow
Imm: immediate
def: deferred
Ind: independent
A. Event identification
B. Related rules activation
C. Scheduling the activated rules for evaluating their conditions to identify the ready-to-execute rules
D. Scheduling the ready-to-execute rules for executing their actions

Fig. 2 Rules Processing Cycle

2.3. Rule Scheduling Evaluation Criteria

There are several approaches to the scheduling of active rules. These approaches are discussed in section 3. There are five criteria used to evaluate the effectiveness and efficiency of the rule scheduling approaches. These criteria are Average Response Time, Response Time Standard Deviation, Throughput, CPU Utilization, and Time Overhead per Transaction [30-31]:

1. **Average Response Time:** The response time of a given rule is the time that an ADS takes to execute the action section of the rule completely after the rule is activated. Average response time is a standard metric for evaluating the effectiveness of rule scheduling approaches. The average response time of a given set of rules is equal to the sum of their response time divided by their number. The lower the average response time of each rule set scheduled, the higher the effectiveness of the rule scheduling mechanism used.
2. **Response Time Standard Deviation:** The response time standard deviation of a given set of rules shows how much variation exists in the response time of each rule from the average response time of the rules. A low response time standard deviation indicates that the response time of each rule scheduled tends to be very close to their average response time. In other words, the lower the response time standard deviation, the closer the response time of each rule to the average response time. A low response time standard deviation facilitates better prediction of the behavior of the system. If two different rule scheduling approaches have equal average response time, the one that has the lower response time standard deviation is more effective than the other approach.
3. **Throughput:** Throughput, which is another standard metric for evaluating the effectiveness of rule scheduling approaches, is the number of executed rules per unit of time. Throughput and Average Response Time may have a direct or an inverse relationship with each other under the certain circumstances.
4. **CPU utilization:** CPU utilization is defined as the amount of time spent purely on executing the activated rules.
5. **Time Overhead per Transaction:** Time Overhead per Transaction is another metric for evaluating the efficiency of rule scheduling approaches. This metric shows the amount of time overhead of a scheduling mechanism. The time overhead of a rule scheduling mechanism includes every extra or indirect computation time that is required to execute active rules.

2.4. Estimation of Actual Execution Time

The accurate estimation of the actual execution time of a given rule is needed for scheduling rules effectively. Although it is possible to calculate the actual execution time of each active rule at run time, such a calculation imposes too much time overhead on the system. This time overhead significantly decreases the effectiveness and efficiency of the scheduling process. As an alternative, the behavior of each active rule could be estimated before the run time. To do so, the child rules of each parent rule should be estimated before run time. In order to make such an estimation, some preprocessing should be applied to the rule-base. For example, it needs to discover the dependencies of all active rules on each other in the rule-base as a hierarchical model, called rule-base tree [10], [11]. Fig. 3 depicts the hierarchical model of the execution of transaction *T* and its children whose life time is illustrated in Fig. 1.

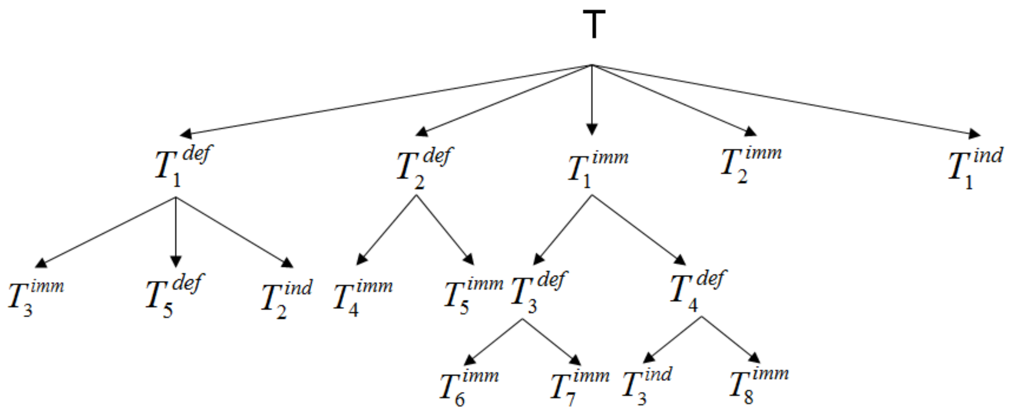


Fig. 3 The hierarchical model of the execution of transaction *T* and its children

As previously mentioned, the execution of the action section of an active rule depends on the evaluation result of its condition section at evaluation time. In other words, the probability of the execution of an active rule equals the probability of truth of its condition section. If the condition section of a rule is evaluated to false, its action section is not executed. The probability of truth of the condition section of an active rule is the extent to which it is possible for the condition section of the rule to be true at the evaluation time during the system's running process. The truth (or falsehood) of the condition section of each child rule, activated during the execution of a parent rule, affects the actual execution time of the parent rule. The more accurately the execution probabilities of active rules are estimated, the more precisely the actual execution times of the rule are calculated [10] - which, in turn leads to more effective and efficient scheduling of active rules. Estimation of the actual execution times of active rules has the main role for introducing an effective rule scheduling process [3],[4],[10]. One key issue is being able to accurately estimate how long an active rule takes to be executed.

E_X-SJF_{EsTLA}, using learning machine capabilities, improves the active rule scheduling process proposed by E_x-SJF_{PRO}-V.2.8. The E_X-SJF_{EsTLA} approach uses learning to make better estimates of the probability of truth of the condition section of each rule - which, in turn, improves the performance of the rule scheduling process of ADSs. This work is a step towards the accurate estimation of the actual execution time of active rules.

3. Related Works

As shown in Fig. 4, existing rule scheduling approaches are divided into two major categories: 1) concurrent scheduling approaches and 2) serial scheduling approaches. The first category is fundamentally different from the second, and so they are not comparable with each other. Concurrent scheduling approaches require special hardware and software platforms. In these approaches, an ADS evaluates or executes triggered rules concurrently. Serial scheduling methods are divided into two groups: 1) constrained approaches and 2) unconstrained approaches. The former includes EDF-based approaches and the static priority approach. The latter includes the random scheduling mechanism, FCFS, and the SJF-based scheduling approaches. Constrained approaches are used when rules have some restrictions (or constraints) such as deadlines or priorities. Unconstrained approaches can be used for all purposes except when the rules have some restrictions. In other words, unconstrained scheduling algorithms serially schedule the rules that do not have any constraints such as priorities or deadlines.

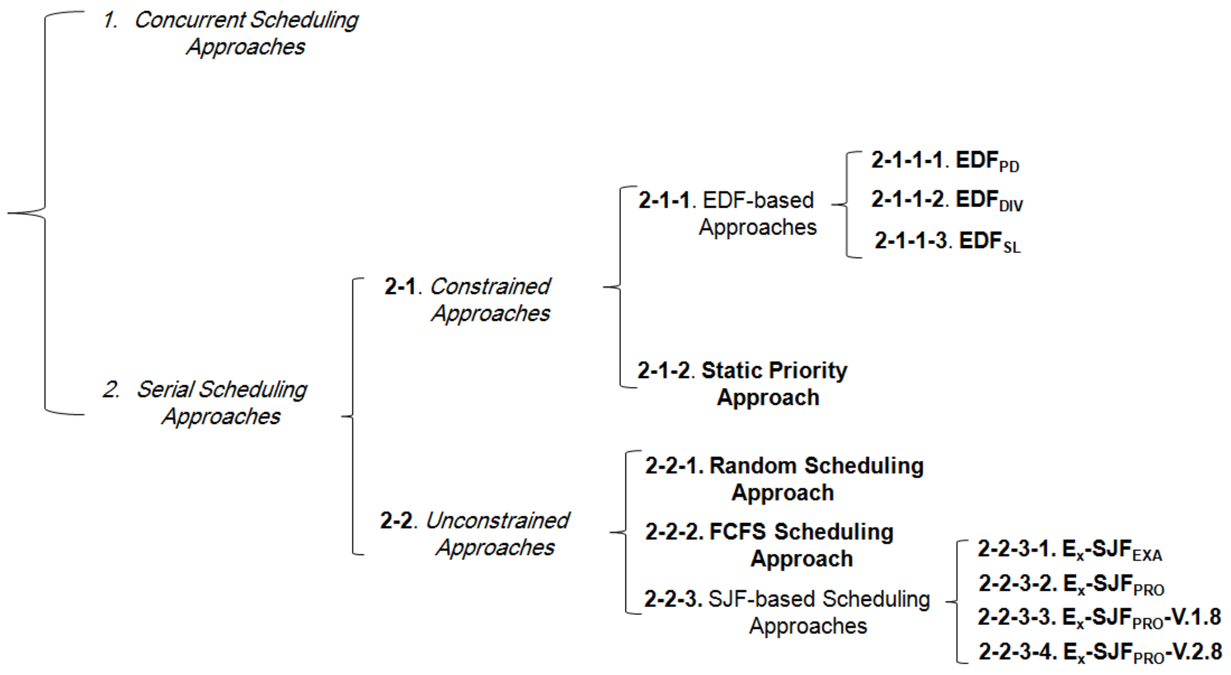

1. Concurrent Scheduling Approaches
2. Serial Scheduling Approaches
2-1. Constrained Approaches
2-2. Unconstrained Approaches
2-1-1. EDF-based Approaches
2-1-2. Static Priority Approach
2-1-1-1. EDF_PD
2-1-1-2. EDF_DIV
2-1-1-3. EDF_SL
2-2-1. Random Scheduling Approach
2-2-2. FCFS Scheduling Approach
2-2-3. SJF-based Scheduling Approaches
2-2-3-1. Ex-SJF_EXA
2-2-3-2. Ex-SJF_PRO
2-2-3-3. Ex-SJF_PRO-V.1.8
2-2-3-4. Ex-SJF_PRO-V.2.8

Fig. 4 The categories of existing rule scheduling approaches

As mentioned in Fig. 2, the processing cycle of active rules consists of two scheduling phases: 1) scheduling activated rules to evaluate their condition sections (Stage C), and 2) scheduling ready-to-be-executed rules to execute their action sections (Stage D). Although scheduling phases 1 and 2 are completely separated, each existing scheduling approach, elaborated on in the following subsections, uses the same mechanism for both scheduling phases mentioned. In other words, although rule scheduling in phases 1 and 2 are totally independent to each other, according to the rule scheduling algorithm used in each existing rule scheduling approaches, each approach uses a special method of scheduling for both phases. The rule scheduling approaches categorized in Fig. 4 are briefly introduced in the following subsections in order of presence in this figure.

3.1. Concurrent Scheduling Approach

HiPAC [12] and FAR [5] active database systems support this approach. In HiPAC, the rules processing cycle is invoked whenever an event occurs and triggers some rules. This method differs from other approaches to scheduling multiple triggered rules. Therefore, it cannot be compared with serial scheduling approaches. HiPAC evaluates or executes triggered rules concurrently. This means that if some rules are triggered during the execution of a rule, they are evaluated or executed concurrently. IRS and CIRS are two other concurrent rule scheduling approaches which are presented in [25] and [24], respectively. The performance of parallel scheduling methods is better than sequential ones, but they are not able to be executed in all hardware and software platforms. As mentioned in [25], one of the open problems of ADSs, which prevents the widespread use of ADSs, is lack of an effective rule scheduling approach that can be used in various hardware and software circumstances.

3.2. Serial Rule Scheduling Approaches

As previously mentioned, serial scheduling methods are divided into two groups: 1) constrained approaches and 2) unconstrained approaches.

3.2.1. Constrained Approaches

This group of scheduling approaches includes EDF-based approaches and the static priority approach. Constrained approaches are used when rules have some restrictions (or constraints) such as deadlines or priorities. There are no other kinds of constraints or restrictions for active rules except deadline or priority.

3.2.1.1. EDF based Scheduling Approach

EDF (Earliest Deadline First) is one of the best classic algorithms that have so far been introduced for rule scheduling in real-time systems [2], [5], [7]. The EDF based approach, which is constrained approach, schedules active rules based on their deadlines in RADBs (Real-time Active Databases). When an active rule should be selected among a set of rules to be evaluated or executed, the rule, which has the nearest deadline, is selected. The EDF based approach has three different versions: EDF_{PD}, EDF_{DIV}, and EDF_{SL}. In EDF_{PD}, the deadline assigned to each rule is not altered during its life-time. However, in EDF_{DIV} and EDF_{SL}, the deadline assigned to each rule is dynamically altered during its life-time according to the number of the immediate and deferred rules activated due to its execution. EDF_{SL} uses the LSF (Least Slack First) technique. In this approach, the deadline of a rule is estimated according to 1) the estimation of the execution time of the rule, 2) the number of the rules that have been activated since the start of the execution of the rule, and 3) the estimation of the number of the rules that will be activated during the remainder time of the execution of the rule. In order to use EDF_{SL}, the information mentioned above (in particular, the third one) should be available. If the required information is available for all rules, EDF_{SL} is the most effective approach among the three versions of the EDF based approach [16].

3.2.1.2. Static Priority Scheduling Approach

In this constrained approach, the system assigns a numerical priority to each ECA rule [1]. The priorities need not be unique. Ariel and Postgres systems assign a static priority, between -1000 and +1000, to each rule. When an active rule should be selected among a set of rules to be evaluated or executed, the rule that has the minimum static priority is selected [8], [13], [17].

3.2.2. Unconstrained Approaches

This group of scheduling approaches includes the random scheduling mechanism, FCFS, and the SJF-based scheduling approaches. Unconstrained approaches can be used for all purposes except when the rules have some restrictions.

3.2.2.1. Random Scheduling Approach

Random scheduling is one of the easiest approaches to scheduling active rules in ADSs [8]. This approach has been implemented in RPL and Ode active database systems [11]. The random scheduling approach selects one of the activated or ready-to-be-executed rules randomly. The main characteristic of this unconstrained approach is its simplicity, though at the cost of efficiency.

3.2.2.2. FCFS Scheduling Approach

FCFS is one of the classic approaches used for rule scheduling in ADSs. When an event occurs, and the corresponding rule(s) is (are) triggered, an instance of each triggered rule is generated. Each instance contains a timestamp, which shows the time when the corresponding rule is triggered. When an active rule should be selected among a set of rules to be evaluated or executed, the rule that has the earliest timestamp is selected. This scheduling approach, used in SAMOS [8], [14], is a unconstrained approach.

3.2.2.3. SJF-based Scheduling Approaches (E_x-SJF)

E_x-SJF is based on the SJF algorithm. SJF is one of the most effective classic scheduling algorithms [4], [10]. It is useful for scheduling the rules (as jobs) when their execution duration is fixed and deterministic. Due to the active work load nature of ADSs, the SJF algorithm needs to go through some changes to be appropriate for scheduling active rules [16]. It needs to be able to estimate the execution time of rules, and that is what E_x-SJF does. E_x-SJF uses the rule tree construct, previously introduced in Fig. 3.

In order to utilize SJF algorithm effectively in ADSs, rules should be scheduled based on the actual execution time of generated transactions. In a higher level view, each transaction along with all its immediate and deferred children can be considered as a virtual transaction. SJF based approaches should schedule virtual transactions generated during the run time of the system. As previously mentioned, E_x-SJF constructs hierarchical models (rule trees) to prepare a rule-base for scheduling active rules according to the actual duration of their execution [10].

The execution time of immediate and deferred child rules has different influences on the calculation process of the execution time of parent rules in different versions of E_x-SJF. There are four versions of E_x-SJF; namely, E_x-SJF$_{EXA}$, E_x-SJF$_{PRO}$, E_x-SJF$_{PRO}$-V.1.8, and E_x-SJF$_{PRO}$-V.2.8 [10]. All versions of E_x-SJF construct the hierarchical models of the execution process of active rules to predict their execution time [4]. The difference between these versions is the manner in which the execution of active rules is estimated. In section 4, we introduce a fifth version of E_x-SJF. All versions of E_x-SJF, which are unconstrained, try to predict the actual execution time of active rules before run time.

The condition section of an active rule contains some conditional expressions. Suppose the condition section of a rule R is defined as $[(A \wedge B) \vee (C \wedge D)]$. A, B, C, and D are logical statements. The probability of truth of the condition section of R is calculated as follows:

$$P[(A \wedge B) \vee (C \wedge D)] = P(A \wedge B) + P(C \wedge D) - P(A \wedge B \wedge C \wedge D)$$

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

$$P[(A \wedge B) \vee (C \wedge D)] = P(A) * P(B) + P(C) * P(D) - P(A) * P(B) * P(C) * P(D) \qquad (1)$$

If the probability of truth of each of A, B, C, and D is available, the probability of truth of the condition section of R can be calculated precisely by putting the probabilities in relation (1). As mentioned in section 2, the probability of truth of the condition section of each active rule should be estimated before run time. Subsequently, the actual execution time of each active rule is approximated by relation (2) before run time.

$$X(R) = L(R) + \sum_{i=1}^{n(R)} P(R_i) * X(R_i) \qquad (2)$$

$P(R_i)$	The probability of truth of the condition section of an immediate or deferred rule R_i activated by R
$X(R_i)$	The actual execution time of an immediate or deferred rule R_i activated by R
$n(R)$	Number of those deferred and immediate rules activated due to the execution of R
$L(R)$	The expected execution time of R
$X(R)$	The actual execution time of R

3.2.2.3.1. E_x-SJF$_{EXA}$ Scheduling Approach

In this approach, the probability of truth of the condition section (the probability of the execution) of each rule is considered as 1 ($P(R_i) = 1$) [4], [16].

3.2.2.3.2. E_x-SJF$_{PRO}$ Scheduling Approach

In this approach, the probability of truth of each conditional statement in the condition section of each rule is uniformly considered as 0.5. Based on relation (1), the probability of truth of the condition section of *R*, defined in section 3.6, will be:

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

The actual execution time of each rule in this approach is estimated by relation (2) before run time [4]. This rule scheduling mechanism hypothesizes that the logical statements of the condition section of each rule are independent of each other. Such an assumption decreases the accuracy of the estimation process. However, this approach estimates the actual execution time of each active rule more accurately than E_x-SJF$_{EXA}$.

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

Before run time, E_x-SJF$_{PRO}$-V.1.8 calculates the actual execution time of each rule as E_x-SJF$_{PRO}$ does [3], [10]. At run time, when rule *R* that has the shortest actual execution time among all activated rules is selected to be evaluated, the probability of truth of each logical statement (LS_i) of the condition section of *R*; i.e. $P(R, LS_i)$ is calculated and stored by relation (3).

$$P(R, LS_i) = \frac{T_1}{T_2} \tag{3}$$

T_1 shows the number of times LS_i has been evaluated to true so far, and T_2 shows the number of times *R* has been activated so far. This process is repeated for each logical statement of the condition section of each activated rule during the running of the system until the change rate of the probability of truth of the logical statement achieves a desired value (e.g., 0.0000001 referred to as ε in general mode). At this time, the initial value (i.e. 0.5) is replaced with the new one. If after a specific period of time, the change rate of the probability of truth of a logical statement does not converge into ε, the average of the results calculated up to that time is considered as the final value of the probability, and the updating process of the probability is stopped. It is evident that the smaller the value ε, the more exact the calculation of $P(R, LS_i)$. Whenever the probability of truth of each logical statement of a condition is updated, the probability of truth of the condition is also updated. Finally, the actual execution time of *R* is updated according to relation (2), whenever the probability of the execution (the probability of truth of the condition section) of each of its deferred and immediate children is updated. After a short time passes from the beginning of the running of the system, which is trivial in comparison with the total running time of the system, the actual execution times of all rules are calculated with sufficient precision, depending on the value ε. This approach is a step towards the accurate calculation of the actual execution time of each active rule.

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

E_X-SJF$_{PRO}$-V.2.8 and E_X-SJF$_{PRO}$-V.1.8 estimate the execution probability of each active rule with two different mechanisms [3]. As mentioned earlier, the condition section of each rule is composed of some logical statements. Table (3) shows four condition sections belonging to four given rules, and Table (4) shows the domain of each conditional variable used in the condition section of each rule mentioned in Table (3). For instance, the condition section of R_1 is true if the data item *A* (DI_A) is greater than the data item *B* (DI_B).

The probability of truth of each logical statement can be calculated mathematically before the system run time, according to the following facts and assumptions: 1) the condition section of each rule and the domains of the conditional variables used in the condition sections are defined before run time, 2) logical statements are independent, and 3) the probability distribution of a conditional variable is uniform; i.e., the values belonging to the domain of the variable have the same probability of occurrence.

Here, the probability of truth of the condition section of rule R_1 is calculated based on the assumptions and information mentioned in Tables (3) and (4).

Table3. The condition sections of four given rules

Rules	**Condition Sections**
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}

Table4. The domain of each conditional variable of the condition section of each rule introduced in Table (3)

$0<DI_A<100$	$-110<DI_B<50$	$-2000<DI_D<50$
DI_C= {shoe, steel, cement, copper, brass, carpet, orange, silver, gold}		

As illustrated in Fig. 5, 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*. Therefore, the probability of truth of $DI_A > DI_B$, on the assumption that the area of the trapezoid *S1* is equal to S_{S1}, and the area of the triangle *S2* is equal to S_{S2}, is calculated with relation (4):

$$P(R_1) = \frac{S_{S1}}{S_{S1} + S_{S2}} \tag{4}$$

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

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

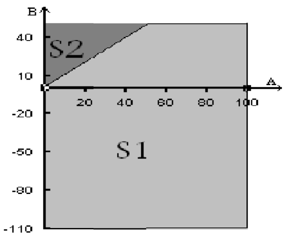


Fig. 5 Calculating the probability of truth of the condition section of R_1

Similarly, the probability of truth of the condition section (the execution probability of the action section) of each rule is calculable before run time. After calculating these probabilities, the actual execution time of each rule can be calculated with relation (2) before run time. At run time, the scheduler continuously selects the rule that has the shortest actual execution time among those activated rules whose condition sections are evaluated to true. Then the scheduler passes the operations of the action section of the selected rule to the transaction execution unit to be executed. As previously mentioned, in this method, the probability distribution of conditional variables is assumed uniform in the calculation process of the actual execution time of each rule. However, in real systems, the probability distribution of conditional variables is not usually uniform. In order to calculate the actual execution time of each rule more precisely, this approach follows a certain procedure during the running of the system:

At run time, during the definite periods of time (δt_i), all assigned values and the duration of their assignment are recorded for every conditional variable of the condition section of each activated rule. The duration of the assignment of a value to a conditional variable is a period of time during which the value is assigned to the conditional variable. Table (5), using an example, reveals how the assigned values of a conditional variable DI_A as well as the duration of their assignment are stored during the successive periods of time (δt_i). In this example, each 1000 minutes is considered as a period of time (δt). Table (5) shows that value '2' has been assigned to DI_A during δt_1 for 10 minutes, and value '11' has been assigned to this variable during δt_n for 320 minutes.

Table5. The manner of recording the assigned values and their duration for DI_A during the running of the system

Periods of time / Conditional Variables	δt_1 (1000 min)		δt_2 (1000 min)		…	δt_n (1000 min)	
	assigned values	the duration of assignment (by min)	assigned values	the duration of assignment (by min)		assigned values	the duration of assignment (by min)
DI_A	2	10	23	8		20	102
	67	5	47	28		71	50
	⋮	⋮	⋮	⋮		⋮	⋮
	32	39	86	95		11	320
	14	15	4	5		1	55

At the end of each δt, the occurrence probability of a given value within the domain of each conditional variable is calculated according to the duration of the assignment of the value to the conditional variable during δt [16]. This process is repeated for each value of the domain of each conditional variable until the change rate of the occurrence probability of that value reaches a specific quantity (e.g., 0.001 referred to as ε in general mode). If after a specific period of time, the change rate of the occurrence probability of a value does not converge into ε, the average of the results calculated up to that time is considered as the final value of the probability, and the updating process of the probability is stopped. At this time, the initial value is replaced with the new one. It is evident that the smaller the value ε is, the more accurate the calculation of the distribution probability of each conditional variable will be. When the occurrence probability of each value within the domain of each conditional variable that exists in the condition section of a rule R is updated, the probability of truth of the condition section of R is mathematically updated. Such an updating is based on the newly updated probability distribution of all its conditional variables, and the initial value is replaced with the new one. Finally, the actual execution time of R is updated based on relation (2), whenever the probability of truth of the condition section of each of its child rules is updated. The actual execution time of each rule is updated with the desired precision (the amount of this precision depends on the value ε) after the passage of a short time from the beginning of the system's running process, and consequently, the system reaches the stable state. Indeed, the system reaches the stable state, as soon as the process of calculating and updating the actual execution time of all rules is completed.

The system on which the scheduler is run is not only responsible for scheduling the defined rules, but also for performing other tasks and functions such as daily computational and operational activities. Based on the assumption that idle times mean the times when the system is doing none of its daily computational and operational tasks, if the termination time of each δt and the idle times of the system can be synchronized, the above-mentioned calculations related to the scheduling process can be done during the idle times of the system, and less computational overhead is imposed on the system. Such synchronization is possible if and only if the idle times of the system are regular. If the system does not have a heavy workload at run time, or it can perform the rule scheduling process along with the other computational tasks without delay, it is not necessary to synchronize the termination time of each δt with the idle or low work times of the system. It is evident that the longer the period between the idle times of the system gets, the longer each δt becomes. The length of a δt does not have any influence on the effectiveness of the system in the stable state but may affect the effectiveness of the system before it reaches the stable state. The smaller the length of each δt, the more frequently the probability of truth of the condition section of each rule is updated – which, in turn, may cause the system to reach the stable state sooner. This also helps the system become more effective before it reaches the stable state, but at the cost of imposing some computational overhead on the system. The administrator of a system should adjust ε according to the required effectiveness and efficiency of the system. The appropriate threshold of effectiveness and efficiency differs from one system to another; therefore, the value ε varies in different situations and environments.

Before introducing the new rule scheduling approach in the next section, the advantages and drawbacks of the existing rule scheduling approaches are qualitatively summarized in Table (6). It is worth mentioning that the advantages and drawbacks of the unconstrained rule scheduling approaches are quantitatively evaluated and discussed in terms of Effectiveness (Average Response Time, Response Time Standard Deviation, and Throughput) and Efficiency (Time Overhead per Transaction and CPU Utilization) in section 5. The advantages and drawbacks of the constrained rule scheduling approaches (the Static Priority and EDF based approaches) have also been discussed in [4], [10], [11].

Table6. The advantages and drawbacks of the existing rule scheduling approaches

	Advantages	**Drawbacks**
Concurrent Scheduling Approach	High performance	Requires special hardware and software platforms.
EDF-based Scheduling Approaches	These approaches are used when rules have deadlines. EDF$_{SL}$ is the most effective between EDF-based with the lowest missed deadlines.	Complex computations and overheads. Special-purpose EDF$_{SL}$ Requires some special information that always may not be available.
Static Priority	Simple and has less computations and overheads. This approach is used when rules have priorities.	Special-purpose Priorities assigned to the rules are static and not suitable for systems that needs to change the priorities at run time

Random	Simple and multi-purpose The least computations and overheads among all the unconstrained rule scheduling approaches.	The worst Average response time among all the unconstrained rule scheduling approaches.
FCFS	Simple and multi-purpose The second least computations and overheads among all the unconstrained rule scheduling approaches.	The second worst Average response time among all the unconstrained rule scheduling approaches.
E_x-SJF$_{EXA}$	Multi-purpose Has better average response time than FCFS and Random scheduling approaches	The actual execution time of each rule is calculated imprecisely because the probability of truth of the condition section (the probability of the execution) of each rule is considered as 1. Therefore, it has the least performance among all the SJF-based scheduling approaches.
E_x-SJF$_{PRO}$	Multi-purpose Has better effectiveness than E_x-SJF$_{EXA}$ because it estimates the actual execution time of each rule more precisely than E_x-SJF$_{EXA.}$	Supposes that the logical statements of the condition section of each rule are independent of each other. Such an assumption decreases the accuracy of the estimation process.
E_x-SJF$_{PRO}$-V.1.8	Multi-purpose Has better effectiveness than E_x-SJF$_{PRO}$ because it estimates the actual execution time of each rule more precisely than E_x-SJF$_{PRO.}$	The mechanism used for estimating the actual execution time of rules is somewhat inefficient and imprecise.
E_x-SJF$_{PRO}$-V.2.8	Multi-purpose Has better effectiveness than E_x-SJF$_{PRO}$-V.1.8 because it is equipped with a more efficient mechanism than E_x-SJF$_{PRO}$-V.1.8 for estimating the actual execution time of rules.	The mechanism used for scheduling activated rules to evaluate their condition sections is not as efficient as possible. This mechanism schedules ready-to-be-evaluated rules regardless of their chance to be evaluated to true. The evaluation process of the condition section of each activated rule that is evaluated to false is useless for the system, which, in turn, decreases the effectiveness of the scheduling process.

4. The New Rule Scheduling Approach

A new rule scheduling approach, called E_X-SJF$_{EsTLA}$ (E_x-SJF [based on]-Estimation-[using]-Learning Automata), is proposed by equipping E_X-SJF$_{PRO}$-V.2.8 with a learning module. This learning module, used during the scheduling process of activated rules, is based on learning automata. It helps the rule scheduler estimate the probability of truth of the condition section of each activated rule more accurately. In other words, the learning module improves the estimation process of the execution probability of each rule. The role of the learning module in the scheduling process will be elaborated further in the following section.

All ready-to-be-executed rules are sequentially executed in the order determined by the rule scheduling mechanism used. In other words, the selection of each ready-to-be-executed rule definitely leads to a useful action; i.e., the execution of the action section of the selected rule. However, the selection process of an activated rule for evaluation of its condition section is not always a useful action; the rule scheduler throws the selected rule out of the rules processing cycle if its condition section is evaluated to false. Therefore, the evaluation process of the condition section of each activated rule that is evaluated to false is useless for the system. Indeed, here the term “useful” means that the scheduling process will ultimately lead to the execution of the action section of the selected rule. In contrast, the term “useless” means that the scheduling process will not ultimately lead to the action section of the selected rule being executed.

Now a question comes to mind: is it possible to reduce the side effects of these useless actions during the scheduling process of the activated rules? The response to this question is the idea behind E_X-SJF$_{EsTLA}$. The higher the priority of those activated rules with condition sections that are more likely to be true at evaluation time, the more effective the rule scheduler will be. To do so, we have chosen the probability of truth of the condition section of each rule as a metric for scheduling activated rules. Therefore, it is necessary to estimate the probability of truth of the condition section of each rule as accurately as possible before run time. The estimation process is divided into two phases: 1) before system run time, E_X-SJF$_{EsTLA}$ calculates the initial probability of truth of the condition section of each rule as E_X-SJF$_{PRO}$-V.2.8 does, and 2) during the running of the system, the real values of the initial probabilities are approximated by the learning automaton embedded in E_X-SJF_{EsTLA}. The learning automaton uses the previous results of the condition evaluation of each rule for estimating the probability of truth of the condition section of the rule in the future. Therefore, the system learns to gradually estimate the real value of each initial probability. The probabilities are estimated more accurately as time passes. The more accurately the probabilities are estimated, the more likely the activated rules whose conditions are evaluated to true are selected. This increases the number of useful actions - which, in turn, improves the effectiveness and efficiency of the scheduling process.

In summary, E_X-SJF_{EsTLA} is designed based on the estimation of the execution probability of rules as E_X-SJF_{PRO}-V.1.8 and E_X-SJF_{PRO}-V.2.8 are. However, E_X-SJF_{EsTLA} attempts to improve the rule scheduling process by increasing the precision of estimation of the execution probability of rules, using learning automata. As previously mentioned, the rules processing cycle consists of two scheduling phases: 1) scheduling activated rules to evaluate their condition sections, and 2) scheduling ready-to-be-executed rules to execute their action sections. All scheduling approaches, which have already been introduced, have the same mechanism for both scheduling phases. For example, in both scheduling phases of the E_X-SJF_{PRO}-V.2.8 approach, active rules are scheduled based on the actual execution time. In E_X-SJF_{EsTLA}, the scheduling mechanisms of activated rules and ready-to-be-executed rules are different from each other: E_X-SJF_{EsTLA} schedules ready-to-be-executed rules as E_X-SJF_{PRO}-V.2.8 does, but it schedules activated rules according to the probability of truth of their condition sections, using a learning automaton.

4.1. Learning Automaton

Reinforcement learning is the study of how artificial systems can learn to optimize their behavior in the face of rewards and punishments. Learning automata (LA), as one of the earliest models of reinforcement learning, are adaptive decision-making devices operating on unknown random environments [6], [9], [15]. 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 how the optimal action (i.e., the action with the highest probability of being rewarded) is chosen through repeated interaction with the environment, using a proper learning algorithm.

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

A fixed structure learning automaton can be represented by a tuple $\{V, \beta, F, G, \phi\}$, where $V = \{\alpha_1, \dots, \alpha_r\}$ is an action set of the automaton, $\beta = \{\beta_1, \dots, \beta_m\}$ is an environment response set, $\phi = \{\phi_1, \phi_2, \dots \phi_s\}$ is a set of internal states, F: $\phi \times \beta \longrightarrow \phi$ is a state transition function, which determines the state of the automaton at an instant $n+1$ with respect to its state and inputs at the instant n, and $G: \phi \longrightarrow \alpha$ is an output function, which determines the output of the automaton at an instant n according to its state at the same instant.

A variable structure learning automaton is a quadruple $\{V, \beta, P, T\}$, where $V = \{\alpha_1, \dots, \alpha_r\}$ is an action set of the automaton, $\beta = \{\beta_1, \dots, \beta_m\}$ is an environment response set, and the probability set $P = \{P_1, \dots, P_r\}$ contains r probabilities, each being the probability of performing a given action in the current internal state of the automaton. The function T is a reinforcement algorithm, which modifies the action probability vector P with respect to the performed action and received response. The selection probability vector of actions at an instant n is as follows:

$$P(n) = \{p_1(n), p_2(n), \dots, p_r(n)\}$$

$$\sum_{i=1}^{r} p_i(n) = 1 \qquad \forall i, p_i(n) = Prob[\alpha(n) = \alpha_i] \tag{5}$$

Consider a variable structure automaton with r actions in a stationary environment with $\beta = \{0,1\}$. The reinforcement algorithm used in the function T for updating actions probabilities is generally as follows:

If $\alpha(n) = \alpha_i$

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

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

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

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

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

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

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:

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

$$g_i[p_j(n)] = {}^{b}\!/_{(r-1)} - bp_j(n) \qquad 0 \leq b < 1$$

where r is the number of actions of the automaton, a is the reward parameter, and b is the penalty parameter, so the general scheme for the learning algorithm is as follows:

If $\alpha(n) = \alpha_i$

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

$$p_i(n+1) = p_i(n) + a[1 - p_i(n)]$$

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

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

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

$$p_j(n+1) = {}^{b}\!/_{(r-1)} + (1-b)p_j(n) \quad \forall j, \quad j \neq i$$

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

4.2. How Does a Learning Automaton Help Us Improve the Rule Scheduling Process?

In E_X-SJF_EsTLA, activated rules are scheduled based on the probability of truth of their condition sections, using a learning automaton. The learning automaton should update the selection probability of each of its actions (i.e., activated rules) according to the responses received from the environment (i.e., the truth or falsehood of the condition section of each activated rule). As mentioned in subsection 4.1, the current action of a fixed structure learning automaton depends on its current state, and all the time, in such a state, this action is its sole option. This means that if a FSLA is in a state φ_j, the corresponding action α_j is always selected to be executed. A variable structure learning automaton, in each selection time, unlike a fixed structure automaton, selects the action that has the greatest success probability among all existing actions. In a VLSA, it is not possible to make a deterministic assignment of an action to a state due to the continual changes of the structure. A learning automaton with a variable structure should be used in the proposed approach because: 1) the learning automaton should select an action (an activated rule) among all existing actions based on the success probability of each action (the probability of truth of the condition section of each activated rule), 2) the number of activated rules changes continuously, and 3) the probability of truth of the condition section of each activated rule is updated continually.

After selection of an activated rule using the learning automaton, the selection probability of the rule should be updated based on one of the three existing mechanisms: L_{RI}, L_{RP}, and $L_{R\varepsilon P}$. We have done some experiments to select one of these updating mechanisms. It is possible to implement these mechanisms in a learning automaton, using some low cost and partial changes into the base automaton. The results of all experiments reveal that with some minor difference, among these three mechanisms, L_{RI} has the best performance for updating the selection probability of each active rule. Moreover, the model of the learning automaton, used in the proposed approach, should be P, in which the input from the environment can take only one of the two values 0 or 1. The response value 1 corresponds to an 'unfavorable' response and occurs when the condition section of the selected rule is evaluated to false, while the output value 0 means the action is 'favorable' and occurs when the condition section of the selected rule is evaluated to true. At each moment, the learning automaton should select one of the activated rules to evaluate its condition section. The number of activated rules at each moment is variable. Consequently, the number of the actions of the automaton is variable too. The set of actions at an instant k is shown by $V(k)$, where $V(k)$ is a subset of V. It should be mentioned that the selection probability of each action at the instant k is proportionate to the probability of the truth of the condition section of the corresponding rule at the same instant:

$$p_i(k) = prob[\alpha(k) = \alpha_i] \ \approx The\ truth\ probability\ of\ the\ condition\ section\ of\ R_i\ at\ instant\ k$$

E$_X$-SJF$_{EsTLA}$ initially calculates the probability of truth of the condition section of each active rule as E$_x$-SJF$_{PRO}$-V.2.8 does. If $S(0)$, shows the sum of the truth probabilities of the condition sections of all rules at instant 0, according to relation (5), the initial selection probability of each action is calculated on scale 1 as follows:

$$p_i(0) = (The\ truth\ probability\ of\ the\ condition\ section\ of\ R_i\ at\ instant\ 0)\Big/ S(0)$$

At instant k, based on the assumption that the actions belonging to $V(k)$ are active, the selection probabilities of the actions are updated according to relations (6), (7), and (8):

$$S(k) = \sum_{\alpha_q \in V(k)}^{q} p_q\,(k) \tag{6}$$

a) The favorable response from the environment in the response of $\alpha_q \in V(k)$: (7)

$$p_q(k+1) = p_q(k) - ap_q(k) + aS(k) \qquad \alpha_q \in V(k)$$
$$p_j(k+1) = p_j(k) - ap_j(k) \qquad \forall j,\ j \neq q,\ \alpha_j \in V(k)$$

b) The unfavorable response from the environment in the response of $\alpha_q \in V(k)$: (8)

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

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

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

Fig. 6 shows how the probability of truth of the condition section of each activated rule is updated, using LA. The cycle of updating the mentioned probabilities is iterated as long as there is more than one activated rule. If there is only one activated rule, it is selected to be evaluated, without changing the selection probabilities of the rules.

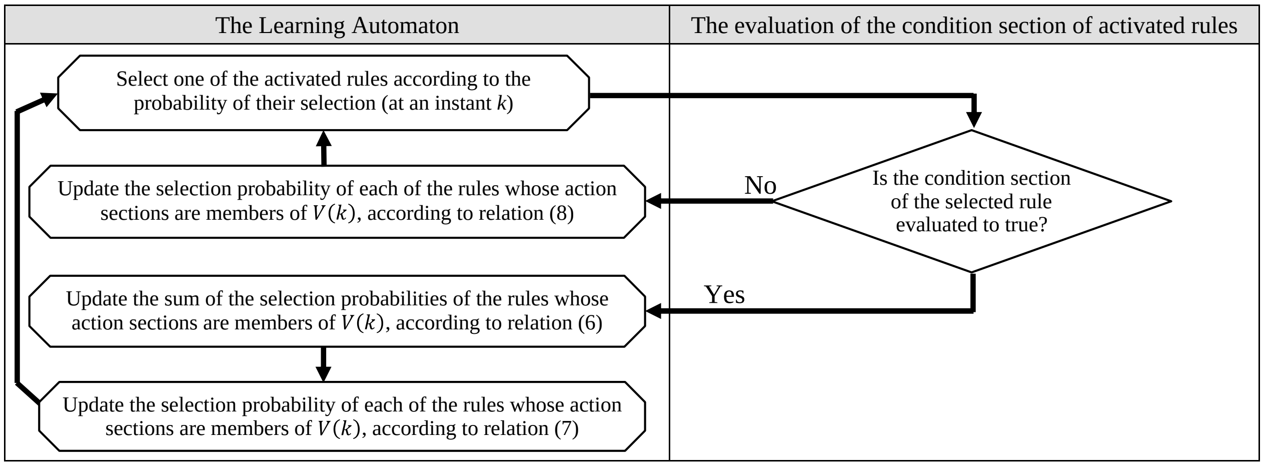

The Learning Automaton
The evaluation of the condition section of activated rules
Select one of the activated rules according to the probability of their selection (at an instant k)
Update the selection probability of each of the rules whose action sections are members of $V(k)$, according to relation (8)
Update the sum of the selection probabilities of the rules whose action sections are members of $V(k)$, according to relation (6)
Update the selection probability of each of the rules whose action sections are members of $V(k)$, according to relation (7)
Is the condition section of the selected rule evaluated to true?
No
Yes

Fig. 6 The updating cycle of the probability of truth of the condition section of each triggered rule using LA

At the run time of the system, during definite periods of time (δt_i), the probability of truth of the condition section (the probability of the execution) of each active rule is updated by the same method mentioned in section 3.6.4. Moreover, at the beginning of δt_{i+1}, the selection probabilities of the actions of the learning automaton are updated based on 1) the probability of truth of the condition section of each active rule and 2) the selection probability of each action of the learning automaton at the end of δt_i according to relation (9):

$p_{R_i,1}(\delta t_k)$: The selection probability of the i^{th}action (rule R_i) calculated by E$_X$-SJF$_{PRO}$-V.2.8 at the end of δt_k in scale 1

$p_{R_i,2}(\delta t_k)$: The selection probability of the i^{th} action (rule R_i) calculated by the learning automaton at the end of δt_k

The selection probability of the i^{th} action (rule R_i) used by the learning automaton at the beginning of δt_{k+1}:

$$P_{R_i}(\delta t_{k+1}) = \frac{\phi(t) * p_{R_i,2}(\delta t_k) + p_{R_i,1}(\delta t_k)}{\phi(t)+1} \qquad ; \phi(t) = 1 - {}^{1}/_{k} \qquad (9)$$

5. Experimental Setup and Evaluation

At first, an evaluation framework is proposed and explained in subsection 5.1. The design of the experiments, along with the evaluation results of E$_X$-SJF$_{EsTLA}$ in comparison with the other unconstrained rule scheduling approaches, are then described in subsection 5.2.

5.1. The Proposed Evaluation Framework

One of the main issues in the field of ADSs is the absence of standard test-beds for the comparison and evaluation of the solutions proposed for different sections of ADSs. For instance, no standard test-bed has been presented for the evaluation of rule scheduling mechanisms yet. We have defined a framework for the comparison and evaluation of existing rule scheduling methods. In this framework, a laboratory environment named ADSS has been designed and implemented to simulate the behavior of active database systems. Each rule scheduling approach can be implemented in ADSS for the evaluation of its effectiveness and efficiency. The details of the design and implementation of ADSS are extensively described in reference [16]. This framework contains five evaluation criteria: Average Response Time, Response Time Standard Deviation, Throughput, Time Overhead per Transaction, and CPU Utilization. The three first criteria are used to evaluate the performance and effectiveness of rule scheduling approaches, and the latter two criteria are used to evaluate the efficiency of rule scheduling approaches. These criteria are formally defined in Table (7). In this table, T is the total cumulative time that an approach has been spent on scheduling active rules in the operational environment of ADSS, and T* is the total time spent on the execution of the action sections of active rules during a test.

Table7. Formal definition of the evaluation parameters

N = The Number of Executed Rules
ART= Average Response Time
	RTSV= Response Time Standard Variance
	U$_{CPU}$ = CPU Utilization
		TOPT = Time Overhead Per Transaction	

$T_1^i = The\ activation\ time\ of\ the\ i^{th}\ rule$	$T_2^i = The\ start\ time\ of\ the\ execution\ of\ the\ i^{th}\ rule$		
$T = (T_2^N + The\ actual\ execution\ time\ of\ the\ N^{th}\ rule) - T_1^1$ $T^* = \sum_{i=1}^{N} The\ actual\ execution\ time\ of\ the\ i^{th}\ rule$	$RTSV = \sqrt{\frac{\sum_{i=1}^{N}((T_2^i - T_1^i) - ART)^2}{N}}$		
$ART = \frac{\sum_{i=1}^{N}(T_2^i - T_1^i)}{N}$	$Throughput = \frac{N}{T}$	$U_{CPU} = \frac{T^*}{T} * 100$	$TOPT = \frac{T - T^*}{N}$

There are two main goals in developing ADSS: 1) reflecting the real behavior of ADSs as much as possible, 2) simulating the various kinds of active rules with different features as well as performing different experiments in several situations (including various rule sets, diverse coupling modes, and different levels of dependency) such that all possible states, which may occur in an ADS, are covered. This is a step towards the development of an appropriate test bed for the evaluation of the effectiveness and efficiency of rule scheduling mechanisms.

There are two compilers in ADSS to compile conditions and instructions, actually. It means in ADSS, active rules are actually compiled during the evaluation and execution of their condition and action sections. These two type of compilation help reflect real behavior of ADSs as much as possible – which, in turn, increase the validity of the results of the experiments. For the evaluation of the condition section of a rule, the conditional variables and comparative operators in the condition section of the rule are identified and separated from each other, and then the following steps are performed: 1) the current value of the conditional variables are fetched from the database, 2) if there are some computational statements in the condition section (such as $x * 2$ in $x * 2 \leq 40$), the required computations are performed, 3) the conditional expressions are evaluated according to the predefined orders and principles, and 4) the evaluation result (true or false) is determined. Similarly, all instructions in the action section of each rule are actually executed. The execution of the action section of each rule causes the occurrence of corresponding events, and consequently related rules are activated.

In the development cycle of ADSS, we used object oriented principles such as modularity and polymorphism to improve its flexibility. This allows us to simply change the rule scheduling mechanism used. The ADSS has a three-tier architecture: "object manager unit", "rule manager unit," and "transaction manager unit" [16]. In ADSS, we have considered and implemented the required aspects of ADSs to cover all modes and properties of active rules in the real world. Active rules, generated in the experiments, have been designed based on the ECA model. This means that each rule has three main sections: event, condition, and action. The event section of a rule is randomly defined based on the various types of real events such as changes in the values of the data items that exist in the database of ADSS (internal events), time-sensitive events, receipt of information from sensors (external events), or a combination of them. The condition section of each rule is randomly defined based on the various types of real conditions; i.e., simple conditional expressions or different combinations of them, using logical operators ($\vee, \wedge, and\ \neg$). Simple conditional expressions are randomly defined, using various data items that exist in the database and different comparative operators. The initial values of data items are also determined randomly; however, their values may be changed by the execution of the action section of each rule at run time. The action section of each rule consists of some instructions which manipulate the data items of the database.

In an ADS, according to the requirements and goals of the system, required reactions for various events that might occur are defined in terms of some active rules with appropriate conditions and actions. In ADSS, we have tried to simulate various events of the real world. Each rule, in addition to its three main sections, contains such other sections for defining its event-condition coupling mode, condition-action coupling mode, priority, deadline, time-stamp, and execution duration. The sections assigned for the priority, deadline, time-stamp, and the execution time of each rule make it possible to use and evaluate the static priority, EDF-based, FCFS, and SJF-based rule scheduling approaches, respectively. Each of these sections is activated and initialized manually or randomly, provided that the corresponding scheduling method is used as the scheduling mechanism of the rule scheduler. A rule generator has been developed to generate various active rules with the mentioned characteristics. The rule generator gets some information as input – for example the number of data items, conditions, actions, events, and rules involved in each experiment – and generates some appropriate rules randomly, as output, according to the input information. It is possible to record the generated rules to be used in various experiments.

5.2. Evaluation of Rule Scheduling Approaches

In this section, we present the evaluation results of the proposed approach in comparison with the other unconstrained rule scheduling approaches discussed earlier. The experiments were designed in two different cases as follows: in the first case, some sets of rules with normal correlation and dependencies were used. The second case consists of the rules that are more correlated and dependent than the rules of the first case. The greater the number of the rules activated due to the execution of a given rule belonging to a rule set, the more correlated and interdependent the rules of the set will be. Each case of experiments was performed in three modes: "Deferred mode", "Immediate mode," and "Composite mode." In the first mode, ADSS uses only the rules whose event-condition and condition-action coupling modes are deferred. In the second mode, ADSS uses only rules with immediate event-condition and condition-action coupling modes, and ultimately in the third one, ADSS uses various rules with different modes.

One of the inputs of the rule generator is the "mode" of the experiments. The fields that are assigned to the event-condition coupling mode and the condition-action coupling mode of each generated rule are initiated according to the input mode. If the input mode is "deferred" or "immediate," the value "deferred" or "immediate" is assigned to the event-condition coupling mode and the condition-action coupling mode of each generated rule, respectively. If the input mode is "composite," one of the values "deferred", "immediate," or "independent" is randomly assigned to the event-condition coupling mode and the condition-action coupling mode of each generated rule. Moreover, in the composite mode, the rule generator helps define what percentage of the event-condition and condition-action coupling of generating rules are immediate, deferred, or independent modes.

To evaluate the effectiveness and efficiency of each scheduling method in each of the two cases and the three modes, 20 rule sets (rule-bases) of each size n (n ∈ {100, 200, 400, 800, 1200, 1800, 2100, 2500, 3000, 4000, 5000, 6000, 7000, 8000}) were generated in each case and mode. Each scheduling method was examined, using each rule set, 20 time periods in ADSS. Each experiment lasted t hour(s) ($t \in \{i: \mathbb{N} | 1 \leq i \leq 20\}$). In other words, to make accurate results, each scheduling method was examined under each rule-base for 20 time periods. Some of these results are illustrated in Fig. 7 to 17. Moreover, the results were statistically analyzed by t-test. The results of our statistical analysis are presented in Tables 8 to 15.

It is worth mentioning that the main task of ADSS is to prepare the grounds for evaluating the effectiveness and the efficiency of the various rule scheduling approaches using the defined rule sets. To do so, at the beginning of each experiment, a rule is randomly selected and executed. The execution of the action section of the selected rule causes the corresponding events to occur, and consequently, the related rules are activated. The rule scheduler starts scheduling the activated rules by the rule scheduling approach being evaluated. The process of rule scheduling continues until the end of the experiment. At the end of the experiment, the effectiveness and efficiency of the rule scheduling approach is calculated in terms of the defined metrics. Then, the results along with the required parameters of the experiment are saved in a table for later analysis of the results.

5.2.1. Evaluation in the First Case

We have done several experiments in the first case in the immediate, deferred, and composite modes. One of the other inputs of the rule generator is the “case” of experiments. In the first case, the input value “1” is sent to the generator to determine the case of the experiments. In the first case, the rules used in the experiments should have normal correlation and dependencies. Therefore, the event and action sections of the rules are randomly defined without any restrictions.
Fig. 7 illustrates the average response time of the unconstrained approaches to scheduling thousands active rules with various features in the composite mode. As Fig. 7 shows, E_x-SJF$_{EsTLA}$ has the least (best) average response time during the running of the system, and E_x-SJF$_{PRO\text{-}V.2.8}$ occupies the second rank from the viewpoint of this criterion. The Random scheduling approach has the greatest (worst) average response time among the evaluated approaches. Fig. 7 reveals that the Average Response Time of each of the E_x-SJF-based approaches is less than the Random and FCFS approaches. Fig. 8 shows the throughput of the unconstrained scheduling approaches to scheduling different numbers of active rules (between 100 and 8000) in the composite mode. It can be inferred that E_x-SJF$_{EsTLA}$ has the greatest (best) throughput, and E_x-SJF$_{PRO\text{-}V.2.8}$ occupies the second place based on this criterion. FCFS has the worst throughput.

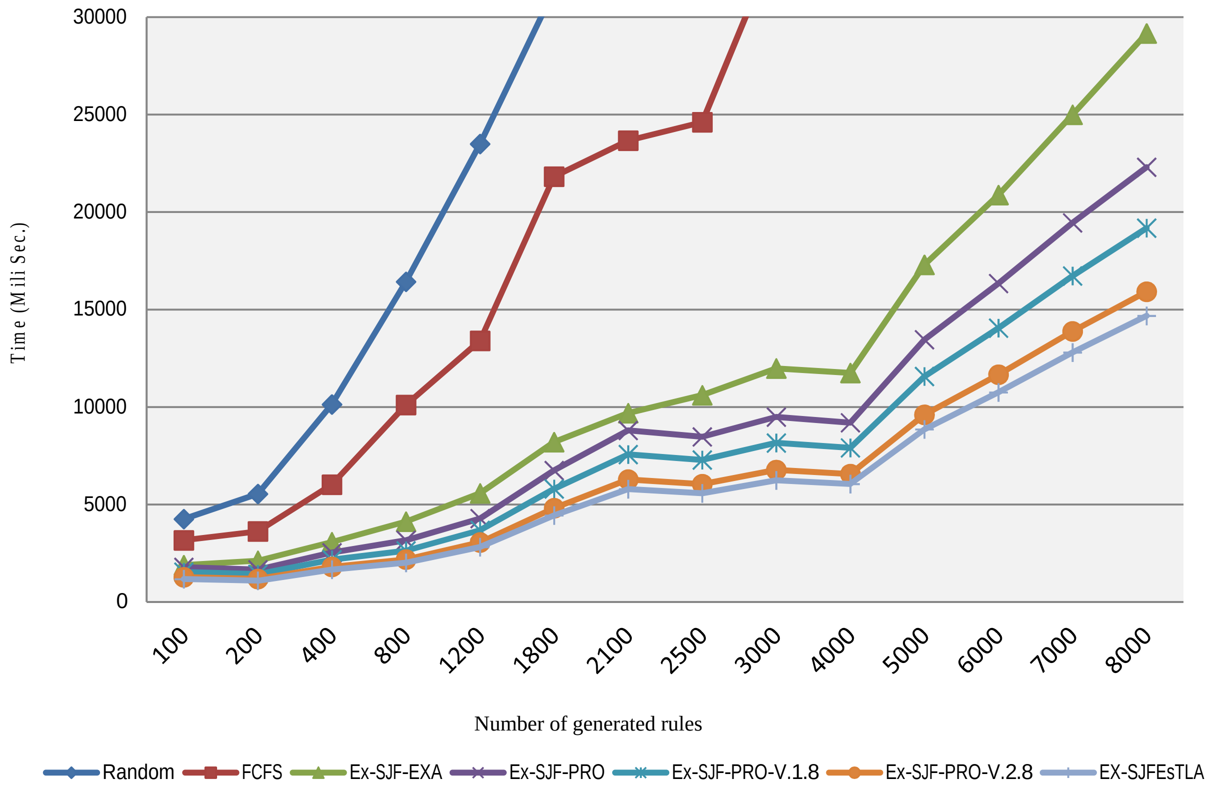


Fig. 7 Average response time diagram of the unconstrained rule scheduling approaches in the composite mode (Corresponding to Table (10))

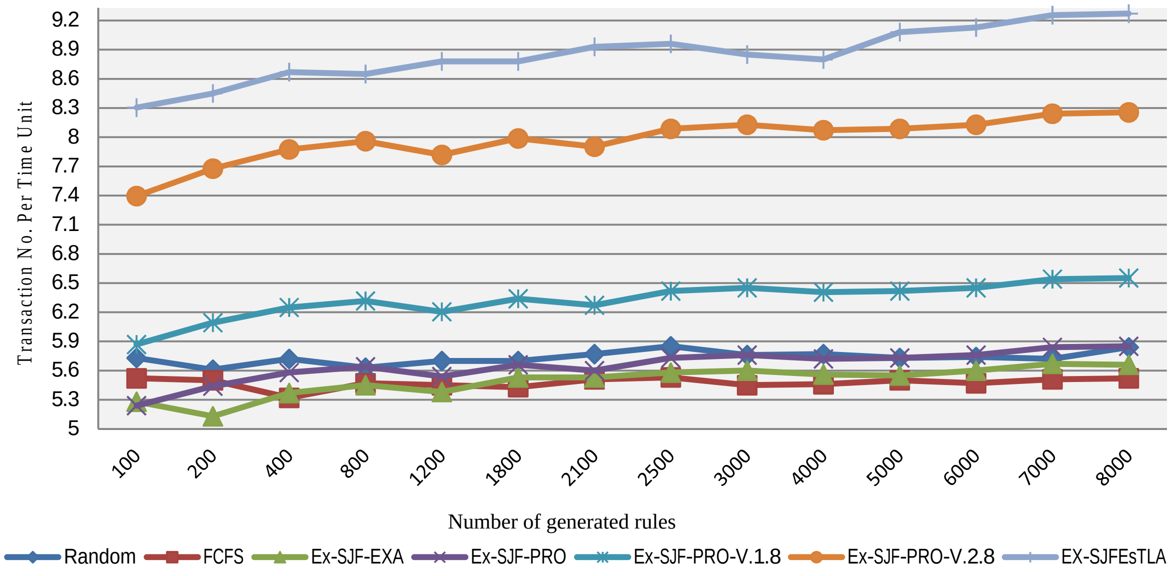


Fig. 8 Throughput diagram of the unconstrained rule scheduling approaches in the composite mode (Corresponding to Table (10))

Fig. 9 shows the time overhead per transaction of the unconstrained rule scheduling approaches for different numbers of active rules in the composite mode. Considering Fig. 9, we can easily conclude that the Random approach has the least (best) time overhead per transaction during the running of the system; however, the other scheduling mechanisms (including E_x-SJF_{EsTLA}, etc) have almost equal time overhead per transaction. Fig. 10 reveals the response time standard deviation of the unconstrained rule scheduling approaches for different rule sets in the composite mode. As Fig. 10 depicts, E_x-SJF_{EsTLA} and E_x-$SJF_{PRO\text{-}v.2.8}$, with a little difference, have the least (best) response time standard deviation among all the unconstrained scheduling mechanisms during the running of the system. E_x-$SJF_{PRO\text{-}V.1.8}$, E_x-SJF_{PRO}, and E_x-SJF_{EXA} occupy the next places, respectively, in this criterion. The Random scheduling approach has the greatest (worst) response time standard deviation. Considering Fig. 10, we can say that the Response Time Standard Deviation of E_x-SJF_{EsTLA} and E_x-$SJF_{PRO\text{-}v.2.8}$ is less than those of the Random and FCFS approaches.

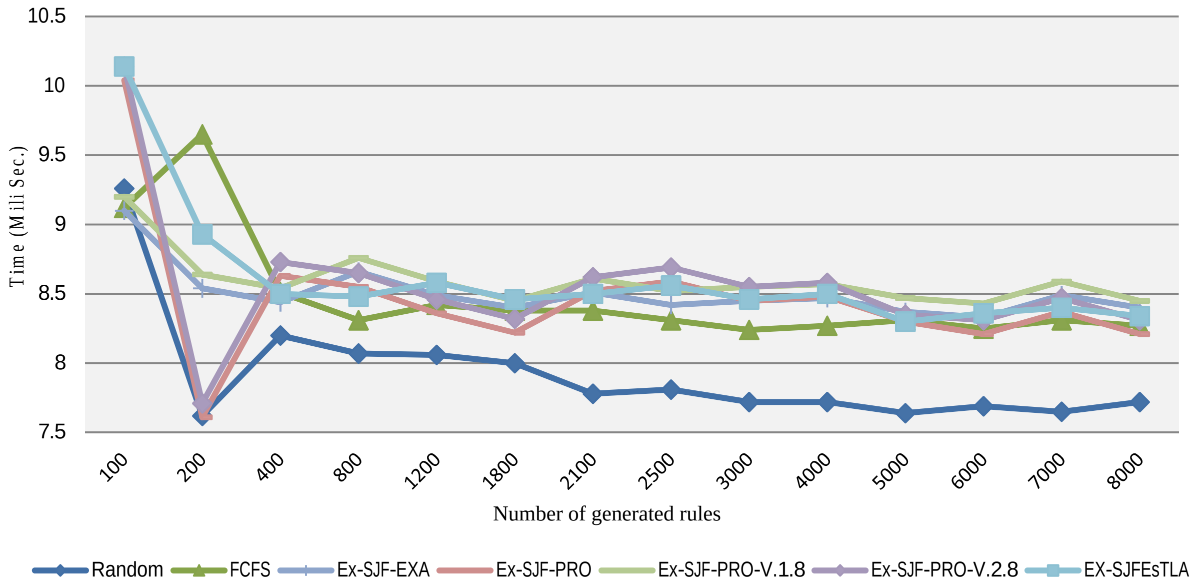


Fig. 9 Time overhead per transaction diagram of the unconstrained approaches in the composite mode (Corresponding to Table (10))

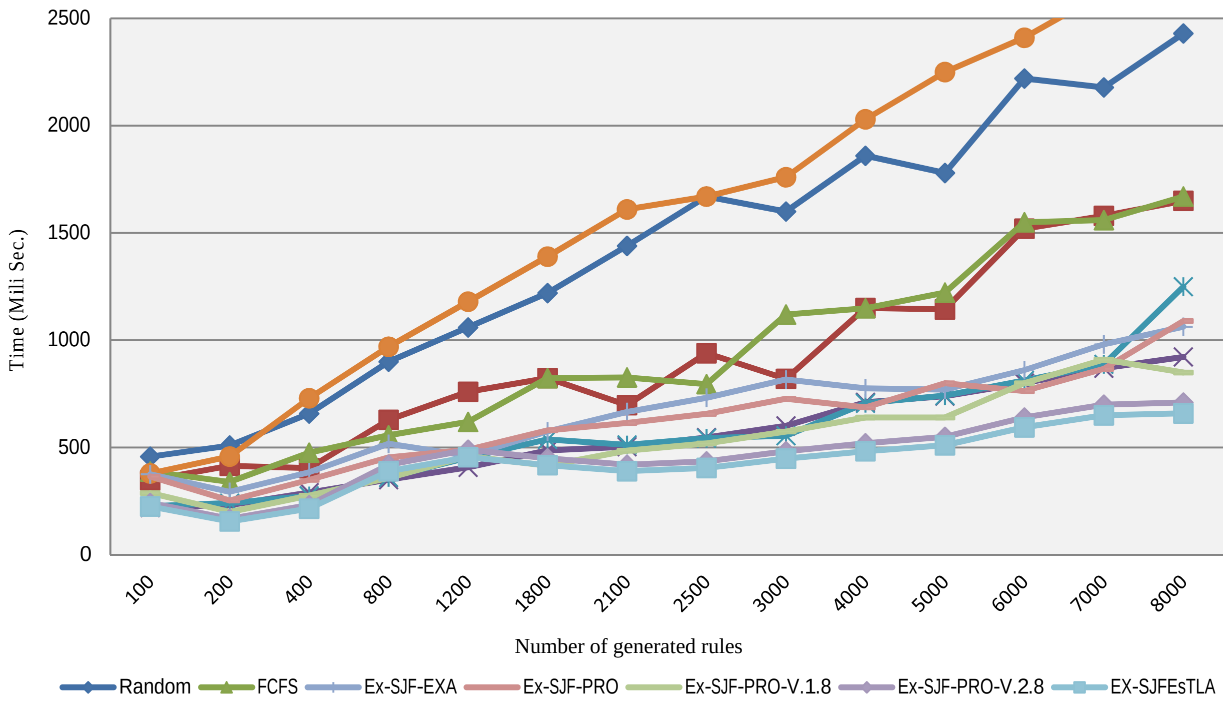


Fig. 10 Response time standard deviation diagram of the unconstrained approaches in the composite mode (Corresponding to Table (10))

Fig. 11 to 17 (see Appendix) show some evaluation results for the unconstrained scheduling approaches in the immediate and deferred modes of the first case. The results of our experiments in the deferred, immediate, and composite modes are shown in Tables (8), (9), and (10), respectively. The content of each cell shows the rank of the corresponding scheduling method based on the corresponding evaluation criterion. In each column in these tables, the lower the value is, the higher the rank will be. For example, in Table (10), the Random approach has the first rank among all the unconstrained scheduling approaches from the viewpoint of time overhead per transaction, because as indicated in Fig. 9, it has the least time overhead per transaction among the unconstrained approaches. As another instance, E$_x$-SJF$_{EsTLA}$ has the first rank among the unconstrained scheduling approaches from the viewpoint of throughput in the composite mode, because it has the greatest throughput among the evaluated approaches. It is worth mentioning that these ranks have been determined based on statistical analysis of experimental results using t-test.

Table8. The results of the evaluation of the unconstrained rule scheduling approaches in the deferred mode

Evaluation Criteria / Approaches	Average Response Time	Response Time Standard Deviation	Throughput	Time Overhead per Transaction	CPU Utilization
Random	5	6	2	1	3
FCFS	5	5	2	1	3
E$_x$-SJF$_{EXA}$	4	7	3	1	2
E$_x$-SJF$_{PRO}$	4	4	3	1	2
E$_x$-SJF$_{PRO}$-V.1.8	3	3	2	1	2
E$_x$-SJF$_{PRO}$-V.2.8	2	2	1	1	1
E$_x$-SJF$_{EsTLA}$	1	1	1	1	1

Table9. The results of the evaluation of the unconstrained rule scheduling approaches in the immediate mode

Evaluation Criteria / Approaches	Average Response Time	Response Time Standard Deviation	Throughput	Time Overhead per Transaction	CPU Utilization
Random	5	6	4	3	3
FCFS	5	6	5	1	1
E$_x$-SJF$_{EXA}$	4	5	5	3	2
E$_x$-SJF$_{PRO}$	3	4	5	2	2
E$_x$-SJF$_{PRO}$-V.1.8	2	3	3	3	2
E$_x$-SJF$_{PRO}$-V.2.8	2	2	2	2	2
E$_x$-SJF$_{EsTLA}$	1	1	1	2	2

Table10. The results of the evaluation of the unconstrained rule scheduling approaches in the composite mode

Evaluation Criteria / Approaches	Average Response Time	Response Time Standard Deviation	Throughput	Time Overhead per Transaction	CPU Utilization
Random	6	6	4	1	1
FCFS	5	5	5	2	1
E$_x$-SJF$_{EXA}$	4	4	4	2	1
E$_x$-SJF$_{PRO}$	4	4	4	2	1
E$_x$-SJF$_{PRO}$-V.1.8	3	3	3	2	1
E$_x$-SJF$_{PRO}$-V.2.8	2	2	2	2	1
E$_x$-SJF$_{EsTLA}$	1	1	1	2	1

In each case of experiments, there is an inverse relationship between the final rank of an approach k and its total score (S(k)) in comparison with the other unconstrained approaches. S(k) is calculated as:

$$S(k) = \sum_{j=1}^{3} \sum_{i=1}^{5} Rnk_{i,j}(k) \tag{10}$$

where i and j are also considered as follows:

$$i = \begin{cases} 1 : if\ metric = & Average\ Response\ Time \\ 2 : if\ metric = & Response\ Time\ Standard\ Deviation \\ 3 : if\ metric = & Throughput \\ 4 : if\ metric = & Time\ Overhead\ per\ Transaction \\ 5 : if\ metric = & CPU\ Utilization \end{cases} \qquad j = \begin{cases} 1 : if\ mode = & deferred \\ 2 : if\ mode = & immediate \\ 3 : if\ mode = & composite \end{cases}$$

In relation (10), $Rnk_{i,j}(k)$ is the rank of the approach k in the j^{th} mode in terms of the i^{th} evaluation metric. For an example, we can calculate $S(Random)$ in the first case of experiments as follows:

$$S(Random) = \sum_{j=1}^{3} \sum_{i=1}^{5} Rnk_{i,j}(Random)$$

$$= \sum_{j=1}^{3} \left[Rnk_{1,j}(Random) + Rnk_{2,j}(Random) + Rnk_{3,j}(Random) + Rnk_{4,j}(Random) + Rnk_{5,j}(Random)\right]$$

$$= [Rnk_{1,1}(Random) + Rnk_{2,1}(Random) + Rnk_{3,1}(Random) + Rnk_{4,1}(Random) + Rnk_{5,1}(Random)]$$
$$+[Rnk_{1,2}(Random) + Rnk_{2,2}(Random) + Rnk_{3,2}(Random) + Rnk_{4,2}(Random) + Rnk_{5,2}(Random)]$$
$$+[Rnk_{1,3}(Random) + Rnk_{2,3}(Random) + Rnk_{3,3}(Random) + Rnk_{4,3}(Random) + Rnk_{5,3}(Random)]$$

According to relation (10), $\mathrm{Rnk}_{5,1}(\mathrm{Random})$ is the rank of the random approach in the “deferred” mode in terms of “CPU Utilization.” According to Table (8), $\mathrm{Rnk}_{5,1}(\mathrm{Random})$ is equal to 3. $\mathrm{Rnk}_{1,2}(\mathrm{Random})$ is the rank of the random approach in the “immediate” mode in terms of “Average Response Time.” According to Table (9), $\mathrm{Rnk}_{5,1}(\mathrm{Random})$ is equal to 5. Therefore, the total score of the Random scheduling approach is calculated according to relation (10) and Tables (8), (9), and (10) as:

$$\mathrm{S(Random)} = \sum_{j=1}^{3} \sum_{i=1}^{5} \mathrm{Rnk}_{i,j}(\mathrm{Random}) = 5+6+2+1+3$$
$$+5+6+4+3+3$$
$$+6+6+4+1+1=56$$

Table (11) shows the total score and final rank of the unconstrained scheduling approaches in the first case of experiments.

Table11. Total score and final rank of the unconstrained scheduling approaches

Approaches	S(k)	Final Rank
Random	56	7
FCFS	52	6
E_x-SJF$_{EXA}$	51	5
E_x-SJF$_{PRO}$	45	4
E_x-SJF$_{PRO}$-V.1.8	36	3
E_x-SJF$_{PRO}$-V.2.8	26	2
E_x-SJF$_{EsTLA}$	18	1

It is worth mentioning that the ranks of the unconstrained scheduling approaches do not change in the experiments of the second case in comparison with those of the first one. In both first and second cases, E_X-SJF$_{EsTLA}$ and E_x-SJF$_{PRO}$-V.2.8 have the first and the second ranks among all the unconstrained approaches, respectively. However, their difference (in terms of efficiency and effectiveness) in the second case is greater than that of the first case. Such a difference is discussed statistically in the rest of this section and section 5.2.2.

As previously mentioned, the experimental results were statistically analyzed using t-test. As an instance, we use the t-test to investigate whether the effectiveness and efficiency of the rule scheduling process is significantly improved in E_X-SJF_{EsTLA} in comparison with E_x-SJF_{PRO}-V.2.8 as the most effective existing approach. The t-test is used for comparing the means of the two samples (or treatments), even if they have different numbers of replicates. In simple terms, the t-test compares the actual difference between the two means in relation to the variation in the data (expressed as the standard deviation of the difference between the means). Based on the characteristics of our experiments, an unpaired t-test, assuming unequal variances, is used to statistically analyze the significance of the experimental results [18].

To compute the *t*-statistic, we need to calculate the following values: $\bar{x}_1, \bar{x}_2$ (the means of the two samples), σ_1^2, σ_2^2 (the variances of the samples), n_1, n_2 (the sample sizes of the samples), and *df* (degrees of freedom):

$$t = \left| \frac{\bar{x}_1 - \bar{x}_2}{\sqrt{({\sigma_1^2}/{n1} + {\sigma_2^2}/{n2})}} \right|$$

The calculated *t*-value should be compared with df to determine the critical t value according to *t*-table at the chosen confidence level (normally $p = 0.05$). If the calculated t value exceeds the tabulated value, we say that the means are significantly different at that level of probability ($p = 0.05$). A significant difference at p = 0.05 means that we can be reasonably confident that the samples/treatments do differ from one another, but we still have nearly a 5% chance of being wrong in reaching this conclusion.

Table (12) presents the results of the statistical evaluation of E_X-SJF_{EsTLA} and E_x-SJF_{PRO}-V.2.8 in the first case of experiments, in terms of Average Response Time, Response Time Standard Deviation, and Throughput. Table (13) presents the results of the statistical evaluation of E_X-SJF_{EsTLA} and E_x-SJF_{PRO}-V.2.8 in the first case of experiments in terms of Time Overhead per Transaction and CPU Utilization. As previously mentioned, in each of the 14 sizes of rule sets ({100, 200, 400, 800, 1200, 1800, 2100, 2500, 3000, 4000, 5000, 6000, 7000, 8000}), 3 modes (the deferred, immediate, and independent modes), and 2 cases (the first and second cases), the effectiveness and efficiency of each rule scheduling approach has been evaluated 20 time periods using 20 rule sets. Each experiment lasted t hour(s) ($t \in \{i : \mathbb{N} | 1 \leq i \leq 20\}$). Indeed, in each size, mode, and case, 400 experiments $(20 * 20)$ were designed for the evaluation of each rule scheduling approach. Therefore, the value that has been inserted in each cell of Tables (12) to (15) is the average of the results of 400 experiments.

Table12. The evaluation results of E_X-SJF_{EsTLA} and E_x-SJF_{PRO}-V.2.8 in the first case of experiments in terms of effectiveness

Evaluation Criteria		Average Response Time						Response Time Standard Deviation						Throughput					
Modes of Experiments		Immediate Mode		Deferred Mode		Composite Mode		Immediate Mode		Deferred Mode		Composite Mode		Immediate Mode		Deferred Mode		Composite Mode	
Scheduling Approaches / Size of Rule-bases		E_x-SJF_{RRO}-V.2.8	E_x-SJF_{EsTLA}	E_x-SJF_{RRO}-V.2.8	E_x-SJF_{EsTLA}	E_x-SJF_{RRO}-V.2.8	E_x-SJF_{EsTLA}	E_x-SJF_{RRO}-V.2.8	E_x-SJF_{EsTLA}	E_x-SJF_{RRO}-V.2.8	E_x-SJF_{EsTLA}	E_x-SJF_{RRO}-V.2.8	E_x-SJF_{EsTLA}	E_x-SJF_{RRO}-V.2.8	E_x-SJF_{EsTLA}	E_x-SJF_{RRO}-V.2.8	E_x-SJF_{EsTLA}	E_x-SJF_{RRO}-V.2.8	E_x-SJF_{EsTLA}
Each result is the average of 20 experiments under 20 rule-bases of each size	100	1816	1622	1234	1110	1275	1175	305	286.2	186	163.8	242	225	5.2	5.56	8.85	9.49	7.4	8.3
	200	1770	1581	1425	1282	1584	1392	217	209.4	132	116.3	168	157	4.88	5.23	8.73	9.36	7.68	8.65
	400	1741	1554	2240	2016	2307	2166	168	156.1	100	88.1	268	250	4.7	4.92	8.64	9.26	7.87	8.77
	800	1718	1533	2800	2220	2983	2313	111	101.5	67	59.03	259	240	4.71	4.83	8.55	9.2	7.96	8.95
	1200	1745	1557	3720	3368	3857	3419	94	88.43	56	49.33	246	230	4.52	4.95	8.64	9.16	7.82	8.78
	1800	1772	1524	4553	3599	4817	3642	75	69.64	45.65	40.22	333	310	4.62	4.95	8.42	9.02	7.99	8.98
	2100	1705	1522	5281	3944	5583	4093	70	67.21	43	37.88	406	380	4.61	4.99	8.5	9.08	7.9	8.93
	2500	1660	1530	6150	4276	6449	4577	66	61.11	38.4	33.83	492	460	4.5	4.94	8.64	9.26	8.1	9.06
	3000	1640	1510	6701	4543	6874	4746	57	53.03	40	35.2	516	480	4.6	4.9	8.82	9.49	8.13	9.05
	4000	1700	1517	7102	4804	7266	5054	50	45.98	30	26.39	438	410	4.5	4.9	8.87	9.56	8.07	9.1
	5000	1660	1527	7670	5136	7806	5357	46	43.5	25	22.03	441	410	4.5	4.9	8.44	9.05	8.09	9.08
	6000	1763	1516	8930	5451	9263	5754	42	39.51	28	24.61	470	440	4.68	4.91	8.63	9.25	8.13	9.13
	7000	1763	1517	9346	6136	9883	6501	38	35.5	23.12	20.39	622	580	4.63	4.91	8.64	9.29	8.24	9.26
	8000	1680	1519	9930	6363	10921	6979	34	32.09	21.08	18.51	601	550	4.61	4.94	8.65	9.28	8.26	9.27
$\bar{x}$ (Mean)		1723.786	1537.786	5505.857	3874.857	5776.286	4083.429	98.07143	92.08571	59.66071	52.54429	393	365.8571	4.661429	4.987857	8.644286	9.267857	7.974286	8.950714

n (number of experiments)	$n_1 = 14$	$n_2 = 14$	$n_1 = 14$	$n_2 = 14$	$n_1 = 14$	$n_2 = 14$	$n_1 = 14$	$n_2 = 14$	$n_1 = 14$	$n_2 = 14$	$n_1 = 14$	$n_2 = 14$	$n_1 = 14$	$n_2 = 14$	$n_1 = 14$	$n_2 = 14$	$n_1 = 14$	$n_2 = 14$
σ^2 (Variance)	2677.874	976.489	8627415	2903692	9677180	3339732	6304.071	5638.632	2316.059	1797.302	20252.15	17350.29	0.034767	0.035249	0.019503	0.027234	0.053549	0.06673
$\sigma_d^2 = \frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}$	261.0259027		823650.5		929779.4		853.0503		293.8116		2685.889		0.005001		0.003338		0.008591	
$t = \left\| \frac{\bar{x}_1 - \bar{x}_2}{\sigma_d} \right\|$	11.51253971		1.797142		1.755619		0.204941		0.415172		0.523735		4.61587		10.7925		10.5344	
df (Degree of Freedom)= $(n_1 + n_2) - 2$	df = 26		df = 26		df = 26		df = 26		df = 26		df = 26		df = 26		df = 26		df = 26	
	• The critical t value (p = 0.05) for 26 degrees of freedom ($n_1 + n_2$ - 2) is 2.06 according to *t*-table.																	
Final Statistical Analysis	11.51254>2.06		1.8≈2.06		1.75≈2.06		0.20494<2.06		0.41517<2.06		0.52374<2.06		4.61587>2.06		10.7925>2.06		10.5344>2.06	

The calculated *t*-values presented in Table (12) imply that in the experiments of the first case, E_X-SJF$_{EsTLA}$ significantly improved the rule scheduling process in terms of throughput in comparison with E_x-SJF$_{PRO}$-V.2.8 in all the deferred, immediate, and independent modes. In addition, E_X-SJF$_{EsTLA}$ significantly decreased the average response time of the rule scheduling process in comparison with E_x-SJF$_{PRO}$-V.2.8 in the immediate mode. A close-to-significant decrease in the average response time of the rule scheduling process was also obtained by E_X-SJF$_{EsTLA}$ in comparison with E_x-SJF$_{PRO}$-V.2.8 in the deferred and independent modes. Moreover, E_X-SJF$_{EsTLA}$ decreased the response time standard deviation of the rule scheduling process in comparison with E_x-SJF$_{PRO}$-V.2.8, but not to a statistically significant degree.

Table13. The evaluation results of E_X-SJF$_{EsTLA}$ and E_x-SJF$_{PRO}$-V.2.8 in the first case of experiments in terms of efficiency

Evaluation Criteria		Time Overhead per Transaction						CPU Utilization					
Modes of Experiments		Immediate Mode		Deferred Mode		Composite Mode		Immediate Mode		Deferred Mode		Composite Mode	
Scheduling Approaches / Size of Rule-bases		E_x-SJF$_{RRO}$-V.2.8	E_X-SJF$_{EsTLA}$	E_x-SJF$_{RRO}$-V.2.8	E_X-SJF$_{EsTLA}$	E_x-SJF$_{RRO}$-V.2.8	E_X-SJF$_{EsTLA}$	E_x-SJF$_{RRO}$-V.2.8	E_X-SJF$_{EsTLA}$	E_x-SJF$_{RRO}$-V.2.8	E_X-SJF$_{EsTLA}$	E_x-SJF$_{RRO}$-V.2.8	E_X-SJF$_{EsTLA}$
Each result is the average of 20 experiments under 20 rule-bases of each size	100	7.97	8.16	10.02	10.05	10.14	10.14	97.51	97.43	95.52	95.47	94.86	94.86
	200	8.77	9	9.8	9.9	7.71	8.93	97.41	97.38	95.42	95.38	95.89	95.24
	400	8.57	8.95	9.6	9.71	8.73	8.56	97.55	97.5	95.49	95.35	95.35	95.38
	800	8.43	8.52	9.3	9.41	8.65	8.48	97.63	97.6	95.26	95.24	95.38	95.42
	1200	8.41	8.43	9.4	9.46	8.46	8.58	97.65	97.62	95.25	95.21	95.39	95.36
	1800	8.48	8.49	9.3	9.36	8.32	8.46	97.65	97.66	95.51	95.54	95.42	95.4
	2100	8.53	8.55	9.23	9.15	8.62	8.5	97.67	97.68	95.51	95.55	95.37	95.36
	2500	8.35	8.34	9.24	9.16	8.69	8.56	97.67	97.69	95.47	95.47	95.39	95.41
	3000	8.43	8.46	9.62	9.45	8.55	8.46	97.65	97.63	95.44	95.42	95.42	95.39
	4000	8.45	8.42	9.34	9.29	8.58	8.5	97.58	97.66	95.58	95.55	95.55	95.51
	5000	8.42	8.41	9.52	9.21	8.35	8.3	97.6	97.61	95.68	95.6	95.46	95.43
	6000	8.41	8.41	9.04	9.18	8.31	8.36	97.57	97.61	95.55	95.59	95.36	95.4
	7000	8.36	8.37	9.25	9.26	8.47	8.4	97.67	97.67	95.67	95.71	95.33	95.36
	8000	8.35	8.36	9.38	9.45	8.31	8.34	97.67	97.69	95.35	95.45	95.39	95.37
$\bar{x}$ (Mean)		8.423571	8.490714	9.431429	9.431429	8.563571	8.612143	97.60571	97.60214	95.47857	95.46643	95.39714	95.34929
n (number of experiments)		$n_1 = 14$	$n_2 = 14$	$n_1 = 14$	$n_2 = 14$	$n_1 = 14$	$n_2 = 14$	$n_1 = 14$	$n_2 = 14$	$n_1 = 14$	$n_2 = 14$	$n_1 = 14$	$n_2 = 14$
σ^2 (Variance)		0.029055	0.050853	0.066459	0.077029	0.270886	0.216218	0.005811	0.009495	0.016767	0.019409	0.043776	0.023146
$\sigma_d^2 = \frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}$		0.005708		0.010249		0.034793		0.001093		0.002584		0.00478	
$t = \left\| \frac{\bar{x}_1 - \bar{x}_2}{\sigma_d} \right\|$		0.88872		1.8E-14		0.2604		0.108013		0.238876		0.692195	
df (Degree of Freedom)= $(n_1 + n_2) - 2$		df = 26		df = 26		df = 26		df = 26		df = 26		df = 26	
		• The critical t value (p = 0.05) for 26 degrees of freedom ($n_1 + n_2$ - 2) is 2.06 according to *t*-table.											
Final Statistical Analysis		0.88872<2.06		1.8E-14<2.06		0.2604<2.06		0.10801<2.06		0.23888<2.06		0.6922<2.06	

The calculated *t*-values presented in Table (13) imply that the difference between the efficiency of E$_X$-SJF$_{EsTLA}$ and E$_x$-SJF$_{PRO}$-V.2.8 is negligible in terms of Time Overhead per Transaction and CPU Utilization in the experiments of the first case. In other words, neither E$_X$-SJF$_{EsTLA}$ nor E$_x$-SJF$_{PRO}$-V.2.8 is significantly more efficient than the other in terms of Overhead per Transaction and CPU Utilization because the corresponding *t*-values are much less than the critical *t*-value, i.e. 2.06.

5.2.2. Evaluation in the Second Case

In the second case, we have done several experiments, using more correlated and interdependent rules in the immediate, deferred, and independent modes. In this case, thousands of rules, which are more dependent on each other in comparison with the first case, have been generated and inserted in the rule-base. This means that the average rate of the activation of child rules in the experiments of the second case is more than that of the first case – which, in turn, makes the number of the child rules activated during the execution of their parents exponentially increase in the second case of the experiments in comparison with the first case of the experiments. In other words, if the number of the child rules activated during the execution of their parents is *n* in the first case of the experiments and is *m* in the second case of the experiments, then *m* is much greater than *n* (*m*>>*n*). For example, 800 experiments have been done in each of the immediate, deferred, and composite modes to evaluate the effectiveness and efficiency of E$_X$-SJF$_{EsTLA}$, using 20 rule sets of size 8000 (400 experiments per each of the first and second cases). The average rate of the activation of child rules in the experiments of the first case is 2. This rate has been increased to 10 in the second case. Based on the assumptions that: 1) each experiment lasts t hours ($t \in \{i: \mathbb{N} | 1 \leq i \leq 20\}$) and 2) the average execution time of the rules executed during the experiments is t', n and m are calculated as:

$$n = 1 + 2^1 + 2^2 + \cdots + 2^k = \sum_{i=0}^{k} 2^i = \frac{(1-2^{k+1})}{1-2} = 2^{k+1} - 1 < 2^{k+1}$$

$$m = 1 + 10^1 + 10^2 + \cdots + 10^k = \sum_{i=0}^{k} 10^i = \frac{(1-10^{k+1})}{1-10} = \frac{10^{k+1}}{9} - \frac{1}{9} \geq 10^k \geq ((2^3)^k = 2^{3k})$$

$$2^{k+1} \ll 2^{3k} \Rightarrow n \ll m$$

where $k \approx {}^{t}/_{t'}$. The greater the value of k is, the greater the difference of n and m will be. As previously mentioned, one of the inputs of the rule generator is the "case" of experiments. In the second case, the input value "2" is sent to the rule generator to determine the case of the experiments. To produce a rule set with more correlated and dependent rules, it is sufficient to generate rules with more correlated event and action sections. To do so, a set of correlated and dependent events and operations have been defined. In the second case, the operations are defined such that a greater number of events can be triggered during their execution in comparison with the experiments of the first case. The rule generator randomly assigned the defined events and operations to the event and action sections of the generating rules, respectively. Indeed, in the second case, we increased the correlation of the event and action sections of the generating rules. This caused more events to be triggered when the operations of the action section of a rule were executed, and this consequently caused more rules to be activated. In other words, the operations defined in the action section of each rule caused more events to occur, and the events that have occurred were used in the event section of more rules.

The results of the experiments in the second case confirmed the results obtained in the first case of the experiments. Moreover, these results indicated that as the rate of dependency and correlation between rules increases, the more effective the proposed approach becomes. The increase in the number of child rules activated while a parent rule is being executed results in an increase in the frequency of estimating the probability of truth of each condition section. As previously mentioned, estimation is the basis of the calculation of the actual execution time of each rule, so in the second case, the accuracy of the estimation process has a greater effect on the precision of the calculation of the actual execution time of each rule. In other words, the more interdependent the active rules are, the more beneficial the role of the estimation process becomes in calculating the actual execution time of each rule. Therefore, the effectiveness of those approches (such as E_X-SJF_{EsTLA}) that calculate the actual execution time of a rule based on estimating the probability of truth of the condition section of the rule is much more obvious in the second case of the experiments.

Table (14) presents the results of the statistical evaluation of E_X-SJF_{EsTLA} and E_x-SJF_{PRO}-V.2.8 in the second case of the experiments in terms of Average Response Time, Response Time Standard Deviation, and Throughput. Table (15) presents the results of the statistical evaluation of E_X-SJF_{EsTLA} and E_x-SJF_{PRO}-V.2.8 in the second case of the experiments in terms of Time Overhead per Transaction and CPU Utilization.

The calculated *t*-values presented in Table (14) imply that in the experiments of the second case, E_X-SJF_{EsTLA} significantly improved the rule scheduling process in terms of average response time and throughput in comparison with E_x-SJF_{PRO}-V.2.8 in all the deferred, immediate, and independent modes. Moreover, E_X-SJF_{EsTLA} decreased the response time standard deviation of the rule scheduling process in comparison with E_x-SJF_{PRO}-V.2.8, but not to a statistically significant degree.

Table14. The evaluation results of E_X-SJF_{EsTLA} and E_x-SJF_{PRO}-V.2.8 in the second case of experiments in terms of effectiveness

Evaluation Criteria		Average Response Time						Response Time Standard Deviation						Throughput					
Modes of Experiments		Immediate Mode		Deferred Mode		Composite Mode		Immediate Mode		Deferred Mode		Composite Mode		Immediate Mode		Deferred Mode		Composite Mode	
Scheduling Approaches / Size of Rule-bases		E_x-SJF_{RRO}-V.2.8	E_X-SJF_{EsTLA}	E_x-SJF_{RRO}-V.2.8	E_X-SJF_{EsTLA}	E_x-SJF_{RRO}-V.2.8	E_X-SJF_{EsTLA}	E_x-SJF_{RRO}-V.2.8	E_X-SJF_{EsTLA}	E_x-SJF_{RRO}-V.2.8	E_X-SJF_{EsTLA}	E_x-SJF_{RRO}-V.2.8	E_X-SJF_{EsTLA}	E_x-SJF_{RRO}-V.2.8	E_X-SJF_{EsTLA}	E_x-SJF_{RRO}-V.2.8	E_X-SJF_{EsTLA}	E_x-SJF_{RRO}-V.2.8	E_X-SJF_{EsTLA}
Each result is the average of 20 experiments under 20 rule-bases of each size	100	1872	1703	1207	870	1262	1160	296	278	180	158	242	226	4	4.23	6.81	7.2	8.1	9.2
	200	1825	1659	1382	1050	1172	1078	210	198	128	112	168	156	3.76	4.03	6.72	7.17	8.4	9.5
	400	1795	1652	2273	1615	1789	1635	158	149	97	84.5	232	215	3.65	3.9	6.65	7.09	8.6	9.8
	800	1771	1621	2716	1801	2161	1880	123	115	64.99	56.32	420	390	3.58	3.84	6.77	7.13	8.7	9.9
	1200	1764	1609	3384	2090	3027	2184	89	83	54.32	46.91	490	455	3.56	3.81	6.65	7.02	8.5	9.7
	1800	1760	1608	3826	2143	3969	2306	72	68	44.28	38.95	450	418	3.56	3.81	6.63	7.07	8.7	9.9
	2100	1758	1608	4130	2302	4220	2424	68	63	41.71	36.78	420	390	3.55	3.8	6.66	7.1	8.6	9.8
	2500	1756	1603	5458	2555	5588	2607	61	57	37.24	32.85	436	405	3.55	3.8	6.65	7	8.8	10.1
	3000	1756	1605	5653	2837	5706	2979	57	53	38.8	34.22	483	449	3.55	3.78	6.65	7.09	8.9	10.1
	4000	1753	1603	5782	2955	5801	3087	50	46	29.1	25.26	520	483	3.52	3.76	6.65	7.09	8.87	10.08
	5000	1753	1604	5819	3210	5910	3205	44	41	24.25	21.18	550	511	3.52	3.76	6.64	7.08	8.89	10.1
	6000	1752	1602	5975	3508	6047	3551	40	38	27.16	23.69	640	595	3.53	3.77	6.64	7.08	8.94	10.15
	7000	1753	1604	6192	4256	6345	4570	37	34	22.42	19.58	700	650	3.53	3.77	6.65	7.09	9.06	10.29
	8000	1732	1581	6718	4567	6761	4658	33	31	20.44	17.83	710	660	3.55	3.79	6.66	7.1	9.08	10.31
$\bar{x}$ (Mean)		1771.429	1618.714	4322.5	2554.21429	4268.429	2666	95.57143	89.57143	57.83643	50.57643	461.5	428.7857	3.600714	3.846429	6.673571	7.093571	8.724286	9.923571
n (number of experiments)		$n_1 = 14$	$n_2 = 14$	$n_1 = 14$	$n_2 = 14$	$n_1 = 14$	$n_2 = 14$	$n_1 = 14$	$n_2 = 14$	$n_1 = 14$	$n_2 = 14$	$n_1 = 14$	$n_2 = 14$	$n_1 = 14$	$n_2 = 14$	$n_1 = 14$	$n_2 = 14$	$n_1 = 14$	$n_2 = 14$
σ^2 (Variance)		1333.34066	998.5275	3514767	1194202.95	4106788	1218211	5932.879	5268.418	2170.947	1667.558	27249.65	23518.95	0.017253	0.017302	0.002917	0.002594	0.072949	0.095179
$\sigma_d^2 = \frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}$		166.5620094		336355		380357.1		800.0926		274.1789		3626.329		0.002468		0.000394		0.012009	
$t = \left\lvert \frac{\bar{x}_1 - \bar{x}_2}{\sigma_d} \right\rvert$		11.83291349		3.048972		2.598262		0.21212		0.438449		0.543255		4.94583		21.1689		10.9438	
df (Degree of Freedom)= $(n_1 + n_2) - 2$		df = 26		df = 26		df = 26		df = 26		df = 26		df = 26		df = 26		df = 26		df = 26	
		• The critical *t* value (p = 0.05) for 26 degrees of freedom ($n_1 + n_2 - 2$) is 2.06 according to *t*-table.																	
Final Statistical Analysis		11.8329>2.06		3.048972>2.06		2.59826>2.06		0.21212<2.06		0.43845<2.06		0.54325<2.06		4.94583>2.06		21.1689>2.06		10.9438>2.06	

The calculated *t*-values presented in Table (15) imply that the difference between the efficiency of E_X-SJF_{EsTLA} and E_x-SJF_{PRO}-V.2.8 was negligible in terms of Time Overhead per Transaction and CPU Utilization in the experiments of the second case. In other words, neither E_X-SJF_{EsTLA} nor E_x-SJF_{PRO}-V.2.8 is significantly more efficient than the other in terms of Overhead per Transaction and CPU Utilization because the corresponding *t*-values are much less than the critical *t*-value, i.e. 2.06.

Table15. The evaluation results of E$_X$-SJF$_{EsTLA}$ and E$_x$-SJF$_{PRO}$-V.2.8 in the second case of experiments in terms of efficiency

Evaluation Criteria		Time Overhead per Transaction						CPU Utilization					
Modes of Experiments		Immediate Mode		Deferred Mode		Composite Mode		Immediate Mode		Deferred Mode		Composite Mode	
Scheduling Approaches / Size of Rule-bases		E$_x$-SJF$_{RRO}$-V.2.8	E$_x$-SJF$_{EsTLA}$	E$_x$-SJF$_{RRO}$-V.2.8	E$_x$-SJF$_{EsTLA}$	E$_x$-SJF$_{RRO}$-V.2.8	E$_x$-SJF$_{EsTLA}$	E$_x$-SJF$_{RRO}$-V.2.8	E$_x$-SJF$_{EsTLA}$	E$_x$-SJF$_{RRO}$-V.2.8	E$_x$-SJF$_{EsTLA}$	E$_x$-SJF$_{RRO}$-V.2.8	E$_x$-SJF$_{EsTLA}$
Each result is the average of 20 experiments under 20 rule-bases of each size	100	7.97	8.16	10.02	10.05	10.14	10.14	97.51	97.43	95.52	95.47	94.86	94.86
	200	8.77	9	9.8	9.89	7.71	8.93	97.41	97.38	95.42	95.37	95.89	95.24
	400	8.57	8.95	9.6	9.71	8.73	8.5	97.55	97.5	95.49	95.35	95.35	95.38
	800	8.43	8.52	9.3	9.41	8.65	8.48	97.63	97.6	95.26	95.24	95.38	95.42
	1200	8.41	8.43	9.4	9.46	8.46	8.58	97.65	97.62	95.25	95.21	95.39	95.36
	1800	8.48	8.49	9.3	9.36	8.32	8.46	97.65	97.66	95.51	95.54	95.42	95.4
	2100	8.53	8.55	9.23	9.15	8.62	8.5	97.67	97.68	95.51	95.55	95.37	95.36
	2500	8.35	8.34	9.24	9.16	8.69	8.56	97.67	97.69	95.47	95.47	95.39	95.41
	3000	8.43	8.46	9.42	9.45	8.55	8.46	97.65	97.63	95.44	95.42	95.42	95.39
	4000	8.45	8.42	9.34	9.29	8.58	8.5	97.59	97.66	95.58	95.55	95.55	95.51
	5000	8.42	8.41	9.33	9.21	8.35	8.3	97.6	97.61	95.68	95.6	95.46	95.43
	6000	8.41	8.41	9.04	9.18	8.31	8.36	97.57	97.61	95.55	95.59	95.36	95.4
	7000	8.36	8.37	9.25	9.26	8.47	8.4	97.67	97.67	95.67	95.71	95.33	95.36
	8000	7.97	8.16	10.02	10.05	10.14	10.14	97.51	97.43	95.52	95.47	94.86	94.86
$\bar{x}$ (Mean)		8.396429	8.476429	9.449286	9.473571	8.694286	8.736429	97.595	97.58357	95.49071	95.46714	95.35929	95.31286
n (number of experiments)													
σ^2 (Variance)		0.043671	0.057732	0.090023	0.103809	0.438703	0.374209	0.006042	0.010809	0.015469	0.01953	0.064423	0.040099
$\sigma_d^2 = \frac{\sigma_1^2}{n_1} + \frac{\sigma_2^2}{n_2}$		0.007243		0.013845		0.058065		0.001204		0.0025		0.007466	
$t = \left\lvert \frac{\bar{x}_1 - \bar{x}_2}{\sigma_d} \right\rvert$		0.94		0.2064		0.17489		0.329408		0.47144		0.537337	
df (Degree of Freedom)= $(n_1 + n_2) - 2$													
		• The critical *t* value (*p* = 0.05) for 26 degrees of freedom ($n_1 + n_2$ - 2) is 2.06 according to *t*-table.											
Final Statistical Analysis		0.94<2.06		0.2064<2.06		0.17489<2.06		0.32941<2.06		0.47144<2.06		0.537337<2.0	

The results of the experiments revealed that E$_x$-SJF$_{EsTLA}$ has the greatest positive impact on the performance and effectiveness (average response time, response time standard deviation, and throughput) of ADSs. E$_x$-SJF$_{EsTLA}$ calculates the actual execution time of each rule more precisely, using a learning automaton, than other versions of the E$_x$-SJF approach, and this finally leads to the improvement of the rule scheduling process. The estimation process of the execution probability of a rule does not impose computational overhead on the system, so it does not have a negative impact on the time overhead per transaction and CPU utilization [10].Therefore, E$_X$-SJF$_{EsTLA}$ and E$_x$-SJF$_{PRO}$-V.2.8 have the same efficiency from the viewpoints of time overhead per transaction and CPU utilization. More information about the details of the results obtained through various experiments is accessible at *http://ceit.aut.ac.ir/islab/Researches/ADS/Experiment-Results.rar*.

E$_X$-SJF$_{EsTLA}$ improves the rule scheduling process of activated rules. The occurrence of such improvement is logically expected. The activated rules whose condition sections are false at the evaluation time are not executed. The time spent for the selection and evaluation of these activated rules is counted as the wasted time in the system, which decreases the effectiveness of the system. If the priority of the selection of these activated rules is decreased, the effectiveness of the system is increased. E$_X$-SJF$_{EsTLA}$ schedules activated rules based on the probability of truth of their condition sections - which, in turn, decreases the priority of the selection of the activated rules with condition sections that are more likely to be false at evaluation time. The computational overhead of E$_X$-SJF$_{EsTLA}$ in comparison with E$_X$-SJF$_{PRO}$-V.2.8 is just in the updating of the probability of truth of the condition section of some activated rules, using relation (9), which is trivial.

6. Conclusion and Future Work

In this paper, at first a set of key terms necessary to communicate the scope of ADSs, such as active work load, the rule scheduling process, and rule scheduling evaluation criteria, were defined. Moreover, the position and importance of the rule scheduling process in an ADS were explained. Afterwards, the existing rule scheduling approaches, as related works, were introduced. As mentioned in [3], [5], [24-29] one of the open problems of ADSs, which prevents the widespread use of ADSs, is lack of an effective rule scheduling approach that can be used in various hardware and software circumstances. One of the most challenges against introducing an effective rule scheduling process is the estimation of the actual execution times of active rules as accurate as possible [3],[4],[10].

In this study, to improve the most effective unconstrained rule scheduling approach (E_X-SJF_{PRO}-V.2.8), a new rule scheduling approach was developed which is equipped with a learning automaton. The new approach was called E_x-SJF_{EsTLA}. The learning automaton helps E_x-SJF_{EsTLA} in improving the effectiveness of the rule scheduling process from two viewpoints: 1) estimating the actual execution time of each active rule more precisely, 2) decreasing the priority of the selection of the activated rules whose condition sections are false at the evaluation time. The time spent for the selection and evaluation of these activated rules is counted as the wasted time in the system, which decreases the effectiveness of the system. Then the effectiveness and efficiency of the unconstrained rule scheduling approaches were evaluated in terms of five evaluation criteria in a laboratory environment called ADSS. The results of the experiments show that E_x-SJF_{EsTLA} has a positive impact on the average response time, response time standard deviation, and throughput of an ADS and does not have any negative impact on its time overhead per transaction and CPU utilization.

In the future, we intend to improve EDF_{DIV} and EDF_{SL}, as the most effective constrained rule scheduling approaches, using learning automata. As mentioned in subsection 3.5, in these two approaches, the deadline assigned to each rule is dynamically altered during its life-time according to the number of the immediate and deferred rules activated due to its execution. In EDF_{SL} , the deadline of a rule is estimated according to 1) the estimation of the execution time of the rule, 2) the number of the rules that have been activated since the start of the execution of the rule, and 3) the estimation of the number of the rules that will be activated during the remainder time of the execution of the rule. Learning automata may increase the precision of estimating the above-mentioned information. The more precise the estimation of the above-mentioned information, the more effective the rule scheduling process in the EDF-based approaches.

Moreover, considering the unsupervised nature of learning in our proposed approach, unsupervised neural networks (such as SOM and ART [34]) would be a future work for investigating the feasibility of solving the rule scheduling problem using the mentioned neural networks.

References

[1] Azmi, Z. R. M.; Abu Bakar, K.; Abdullah, A. H.; Shamsir, M. S.; Wan Manan, W. N.; "*Performance Comparison of Priority Rule Scheduling Algorithms Using Different Inter Arrival Time Jobs in Grid Environment*," International Journal of Grid and Distributed Computing Vol. 4, No. 3, pp. 61-70, 2011.

[2] Anderson, J. H.; Bud, V.; Devi, U. C.; **"***An EDF-based Restricted-migration Scheduling Algorithm for Multiprocessor Soft Real-time Systems,***"** Real-Time Systems, Vol. 38, Issue 2, pp. 58-131**,** 2008.

[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] Rasoolzadegan, A.; Alesheykh, R.; Abdollahzadeh, A.; "*A New Rule Scheduling Approach based on Estimation of Rule Execution Probability in Active Database,*" Journal of Convergence Information Technology (JCIT), Vol. 3, No. 3, PP. 6-14, 2008.

[5] Ceri, S.; Gennaro, C.; Paraboschi, S.; Serazzi, G.; "*Effective Scheduling of Detached Rules in Active Database,*" IEEE Transaction on Knowledge and Data Engineering, Vol. 15, No.1, pp. 2-13, 2003.

[6] Russell, S.; Norvig, P.; *Artificial Intelligence: A Modern Approach,* Prentice Hall, 3th edition, 2009.

[7] Kudrab, T.**; "***REaltime ACtive Heterogeneous Systems - Where Did We Reach After REACH?,***"** From Active Data Management to Event-Based Systems and More, Lecture Notes in Computer Science, Vol. 6462, pp. 44-56, 2010.

[8] Adaikkalavan, R.; Chakravarthy, S.; "*Access Control Using Active Rules,*" Data Security and Security Data, Lecture Notes in Computer Science, Vol. 6121, pp 12-24, 2012.

[9] Torkestani, J.; Meybodi, M. R.; "*A New Vertex Coloring Algorithm Based on Variable Action-Set Learning*

Automata," Computing and Informatics, Vol. 29, pp. 447–466, 2010.

[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 & Communication Technologies: From Theory To Applications, Damascus, Syria , April 24 - 28, pp. 2920 - 2925, 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] Zong, P.; Qin, J.; Yu, Q.; "*Some Key Techniques in Active Information System*," IJCSNS International Journal of Computer Science and Network Security, Vol.7, No. 3, pp. 21-26, 2007.

[13] Krosing, H.; Roybal, K.; Mlodgenski, J.; *PostgreSQL Server Programming*, PACKT publishing, 2013.

[14] Sarkar, A.; Debnath, N. C.; "*Business-object oriented requirements engineering framework*," Journal of Computational Methods in Sciences and Engineering, Vol. 12, Issue s1, pp. 39-51, 2012.

[15] Narendra, K. S.; Thathachar, A. L.; *Learning Automata: An Introduction*, Dover Publications, 2013.

[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 & Computer Engineering Department, Amirkabir University of Technology (Tehran Polytechnic), Tehran, 2006.

[17] Stankovic, J.; Natale, S. M. D.; Buttazo, G. C.; "*A Novel Priority Rule Heuristic: Learning from Justification*," Twenty-Fourth International Conference on Automated Planning and Scheduling, pp. 92-100, California, USA, 2014.

[18] Motulsky, H.; *Intuitive Biostatistics: A Nonmathematical Guide to Statistical Thinking*, Oxford University Press, USA, 2nd edition, 2010.

[19] Kasbon, R.; Shaharom, S.; Mazlan, E. M.; Mahamad, S.; "*Implementing an Active Database for Maintaining Asset Data*," Software Engineering and Knowledge Engineering: Theory and Practice, Advances in Intelligent and Soft Computing, Vol. 115, pp 99-107, 2012.

[20] Ale, J. M.; Espil, M. M.; *Active Rules and Active Databases: Concepts and Applications, Effective Databases for Text & Document Management*, IGI Global, Hershey, PA, 2003.

[21] Spokoiny, A.; Shahar, Y.; "*An Active Database Architecture for Knowledge-based Incremental Abstraction of Complex Concepts from Continuously Arriving Time-oriented Raw Data*," Journal of Intelligent Information Systems (JIIS), Vol. 28, No. 3, pp. 199-231, 2007.

[22] Badia, A.; "*Active Database Systems for Monitoring and Surveillance*," The 1st NSF/NIJ Conference on Intelligence and Security Informatics, pp. 296-307, Tucson, AZ, USA, 2003.

[23] Qiao, Y.; Zhong, K.; Wang, H.; Li, X.; "*Developing Event-Condition-Action Rules in Real-time Active Database*," The 2007 ACM Symposium on Applied Computing, pp. 511-516, Seoul, Korea, 2007.

[24] Jin, Y.; "*Management of Composite Event for Active Database Rule Scheduling*," The 10th IEEE International Conference on Information Reuse & Integration, pp. 300-304, Las Vegas, Nevada, USA, 2009.

[25] Jin, Y.; Urban, S. D.; Dietrich, S. W.; "*A Concurrent Rule Scheduling Algorithm for Active Rules*", Data and Knowledge Engineering, Elsevier Science. Vol. 61, No. 1, pp. 530-546, 2007.

[26] Meenakshi, S.; Thiagarasu, V.; "*Development of Rule Scheduler for Multiple Triggered Rules in Active Object-Relational Database Systems*," International Journal of Innovative Research in Computer and Communication Engineering, Vol. 2, No. 5, pp. 4289-4294, 2014.

[27] Meenakshi, S.; Thiagarasu, V.; "*Design of Rule Scheduler for Trigger Rule Conflict in Active Object-Relational Database Systems*," International Journal of Emerging Technologies in Computational and Applied Sciences (IJETCAS), Vol. 7, No. 2, pp. 185-189, 2014.

[28] Saravanapandi Solairajan, A.; Godwin Jose, C.; Vijaya Rajan, P.; "*Active Schedule Generation Using Priority Dispatching Rule Algorithm For Lean Manufacturing*," International Journal of Emerging Technology and Advanced Engineering, Vol. 3, No. 1, pp. 404-407, 2013.

[29] Narang, B.; Gupta, A.; Bansal, R.; "*Active Schedule Generation Using Priority Dispatching Rule Algorithm For Lean Manufacturing*," 2nd International Conference on Emerging Trends in Engineering and Management (ICETEM), pp. 40-44, 2013.

[30] Silberschatz, A.; Galvin, P. B.; Gagne, G.; *Operating System Concepts*, Wiley, 9th edition, 2012.

[31] Reid, R. D.; Sanders, N. R.; *Operations Management*, Wiley, 5th edition, 2012.

[32] Hangos, K. M.; Lakner, R.; Gerzson, M.; Intelligent Control Systems: An Introduction with Examples, Springer, 2001.

[33] Farias, O.; Labidi, S.; Neto, J. F.; Moura, J.; Albuquerque, S.; *A Real Time Expert System For Decision Making in Rotary Railcar Dumpers, Automation Control - Theory and Practice*, InTech, 2009.

[34] Haykin, S.; *Neural Networks: A Comprehensive Foundation*, Prentice Hall, 3rd edition, 2007.

Appendix

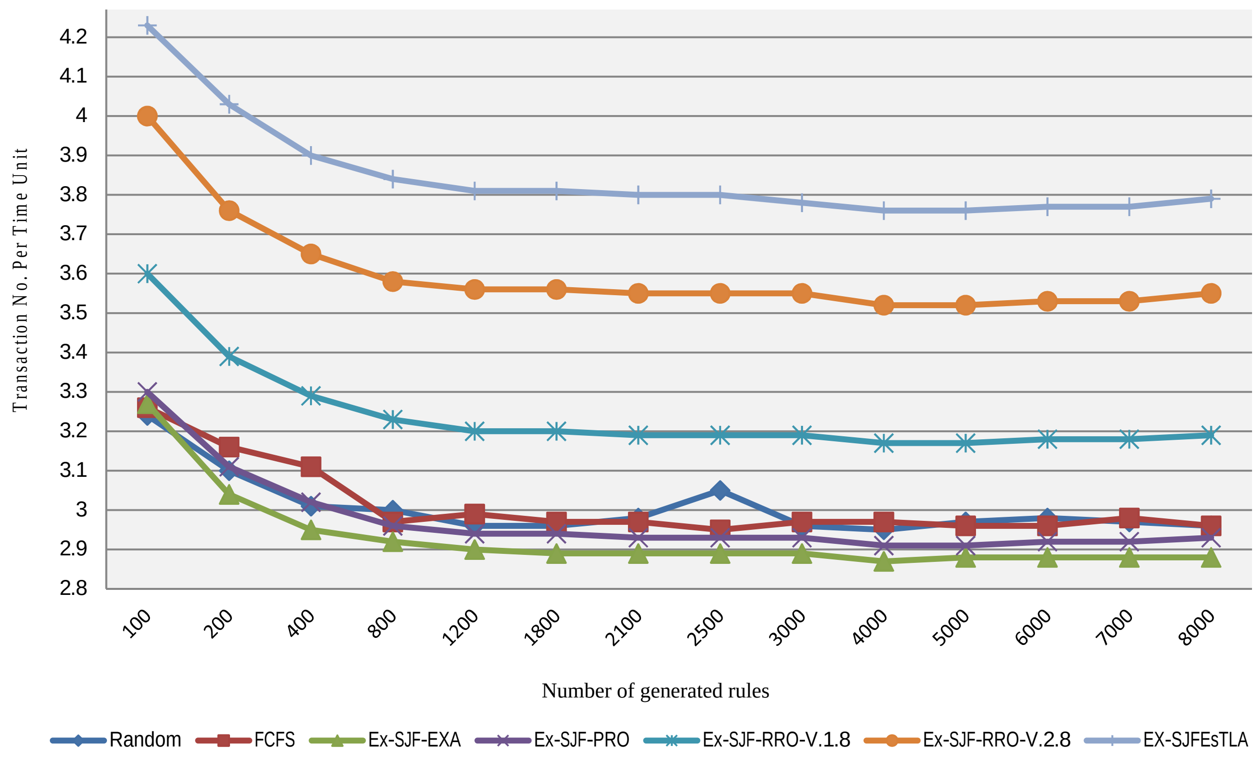


Fig. 11 Throughput of the unconstrained rule scheduling approaches in the immediate mode of the first case of experiments

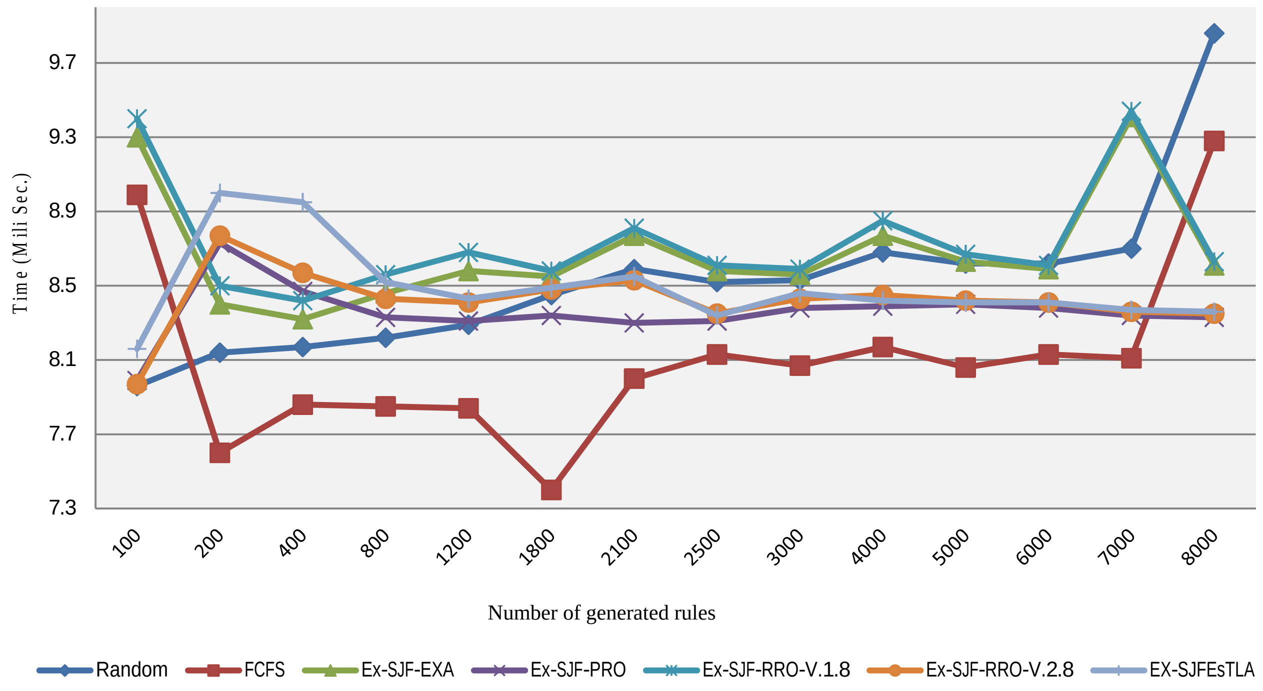


Fig. 12 Time overhead per transaction of the unconstrained rule scheduling approaches in the immediate mode of the first case of experiments

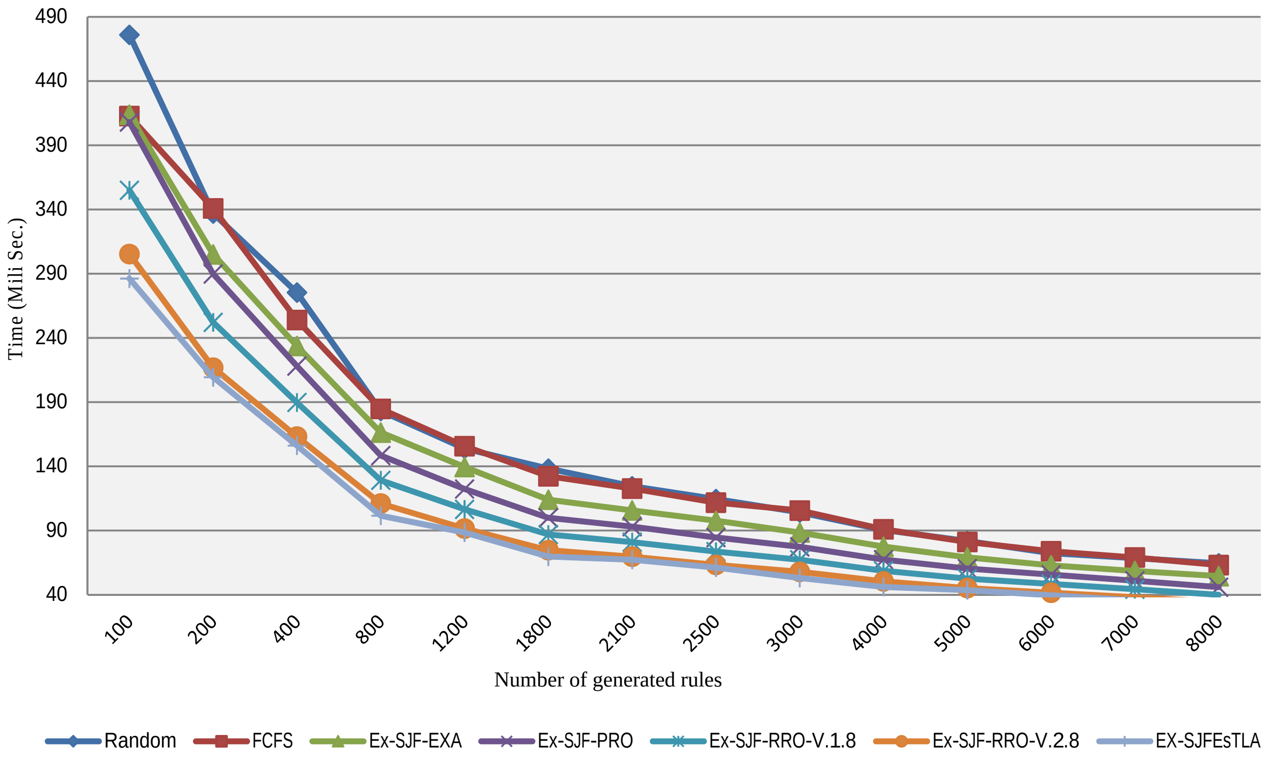


Fig. 13 Response time standard deviation of the unconstrained rule scheduling approaches in the immediate mode of the first case of experiments

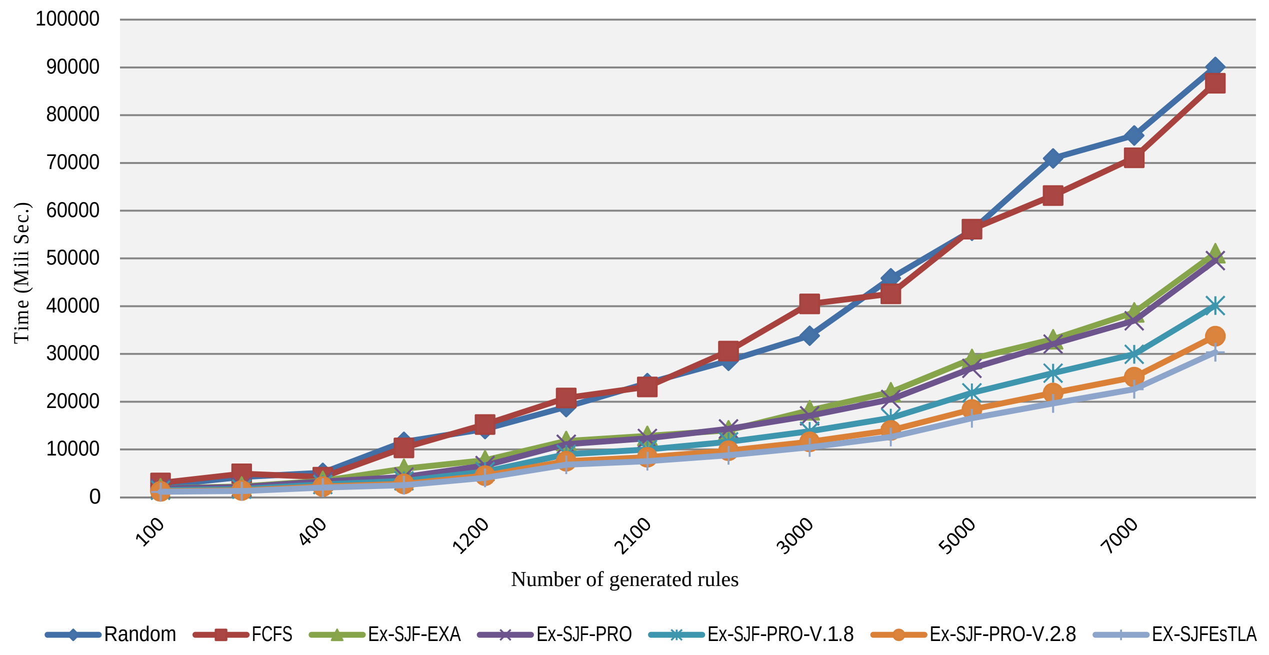


Fig. 14 Average response time of the unconstrained rule scheduling approaches in the deferred mode of the first case of experiments

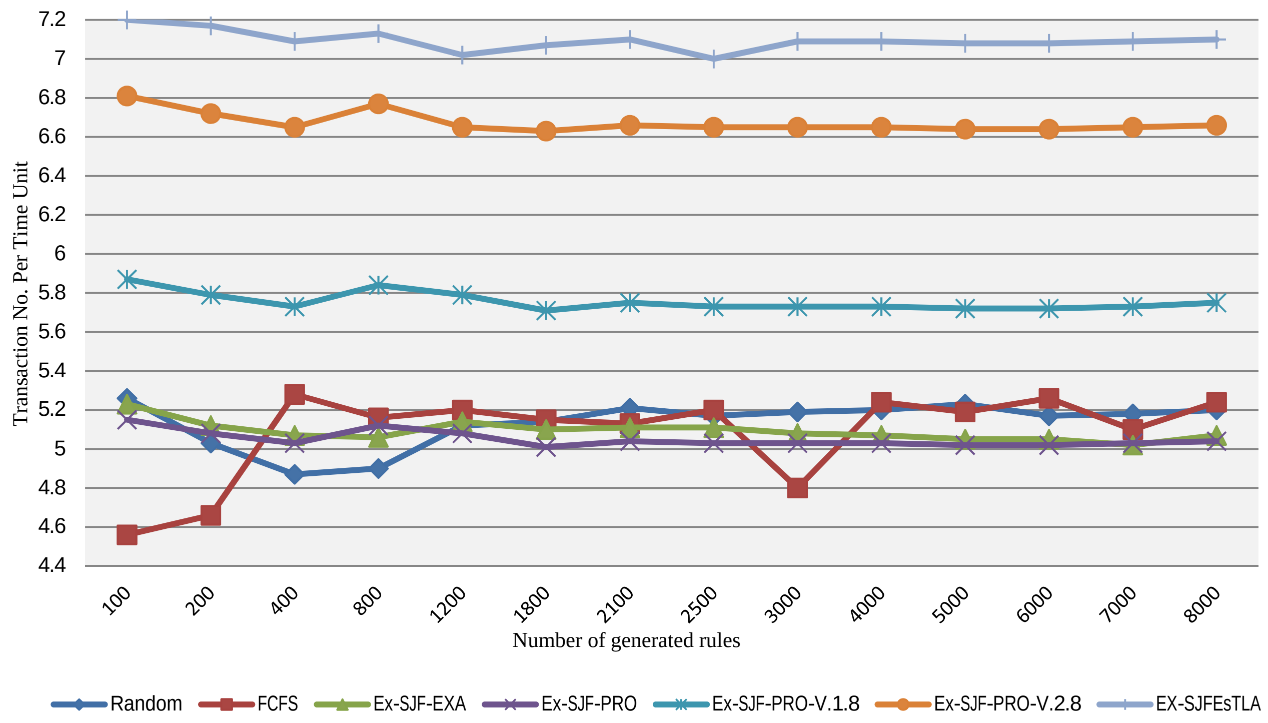


Fig. 15 Throughput of the unconstrained rule scheduling approaches in the deferred mode of the first case of experiments

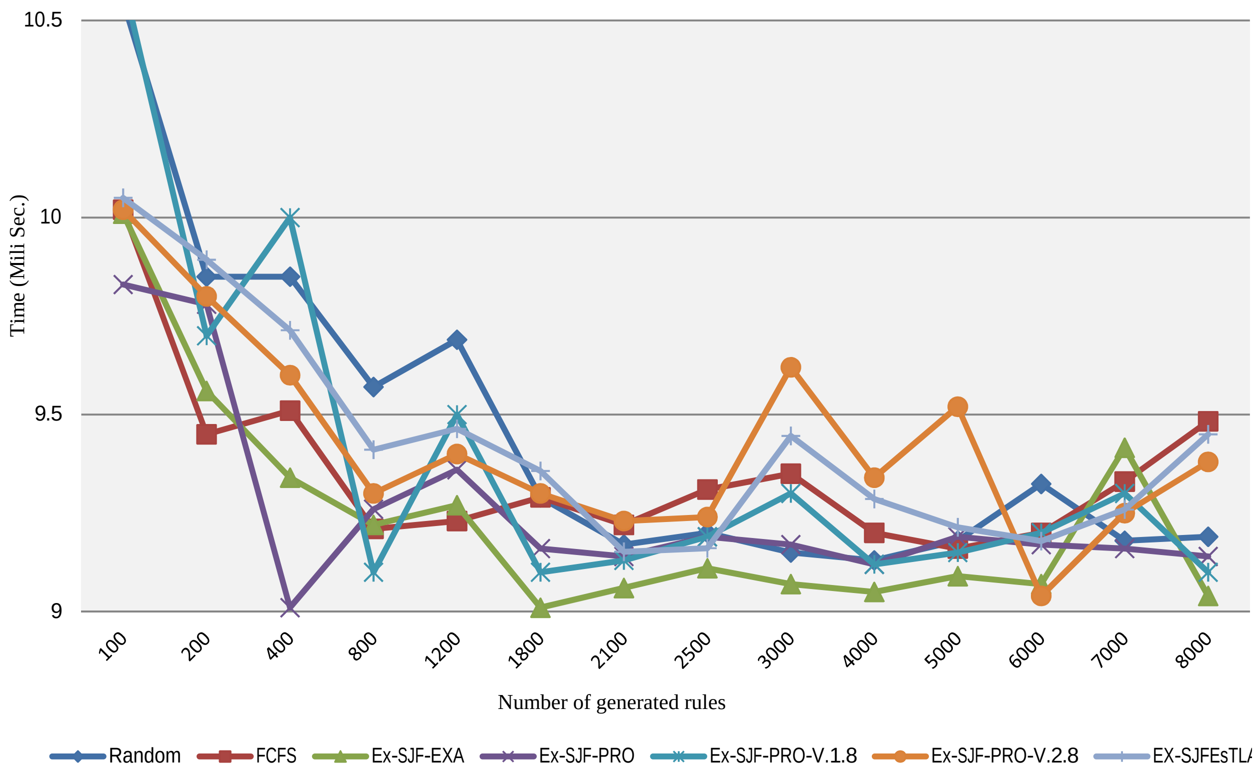


Fig. 16 Time overhead per transaction of the unconstrained rule scheduling approaches in the deferred mode of the first case of experiments

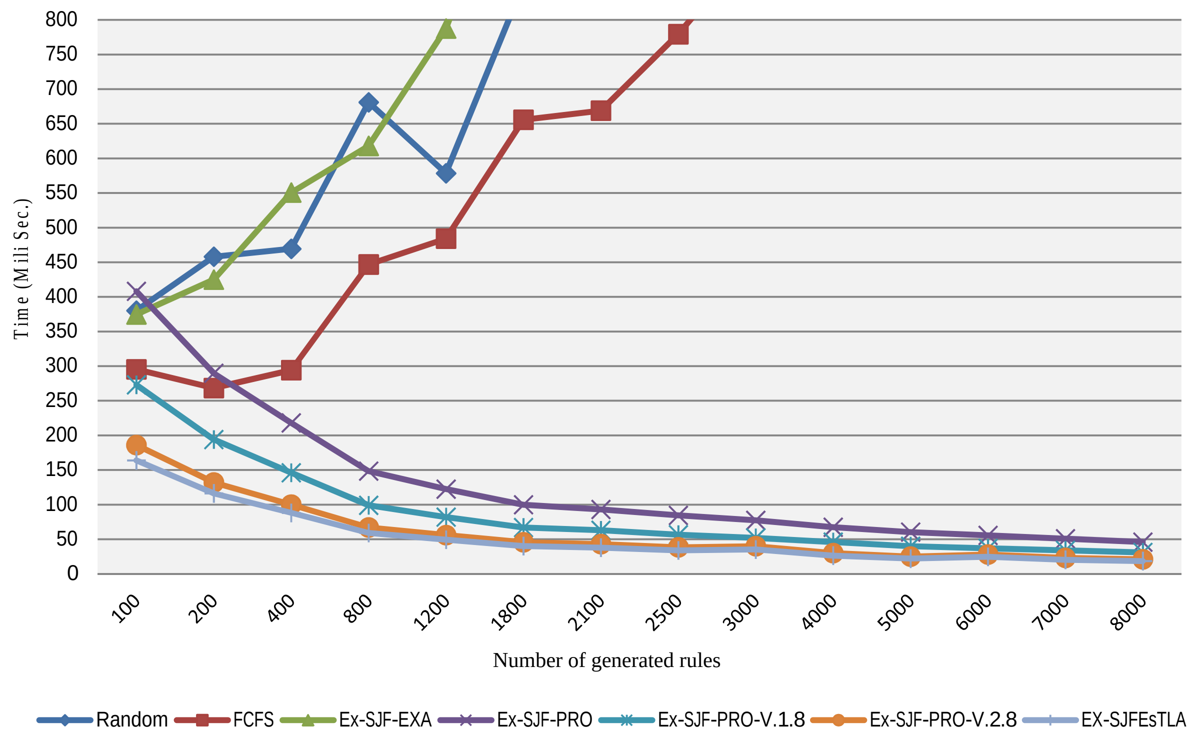


Fig. 17 Response time standard deviation of the unconstrained rule scheduling approaches in the deferred mode of the first case of experiments

Table16. Definition of some rules for a number of n users in a stock exchange example

Rules / **Users**	***Rule #1 of the i^{th} user***	***Rule #2 of the i^{th} user***		***Rule #m_i of the i^{th} user***
The rules defined by the 1st user (i = 1)	***DEFINE*** *User-1.Rule-1* ***ON*** *Stock.MercedesBenz.UpdatePrice* ***IF****Condition-1-1* ***DO*** *Stock.MercedesBenz.Buy*	***DEFINE*** *User-1.Rule-2* ***ON*** *Stock.Toyota.UpdateCount* ***IF*** *Condition-1-2* ***DO*** *Stock.Toyota.Sell*	**...**	***DEFINE*** *User-1.Rule-m_1* ***ON*** *Stock.Volvo.Insert* ***IF*** *Condition-1-m_1* ***DO*** *Stock.Volvo.Buy*
The rules defined by the 2nd user (i =2)	***DEFINE*** *User-2.Rule-1* ***ON*** *Stock.Mazda.UpdatePrice* ***IF*** *Condition-2-1* ***DO*** *Stock.Mazda.Buy*	***DEFINE*** *User-2.Rule-2* ***ON*** *Stock.MercedesBenz.UpdatePrice* ***IF*** *Condition-2-2* ***DO*** *Stock.MercedesBenz.Sell*	**...**	***DEFINE*** *User-2.Rule-m_2* ***ON*** *Stock.Ferrari.Insert* ***IF*** *Condition-2-m_2* ***DO*** *Stock.Ferrari.Buy*
. . .		. . .		
The rules defined by the n^{th} user (i = n)	***DEFINE*** *User-n.Rule-1* ***ON*** *Stock.Mazda.UpdatePrice* ***IF*** *Condition-n-1* ***DO*** *Stock.Mazda.Buy*	***DEFINE*** *User-n.Rule-2* ***ON*** *Stock.MercedesBenz.UpdateCount* ***IF*** *Condition-n-2* ***DO*** *Stock.MercedesBenz.Sell*	**...**	***DEFINE*** *User-n.Rule-m_n* ***ON*** *Stock.Ferrari.Insert* ***IF*** *Condition-n-m_n* ***DO*** *Stock.Ferrari.Buy*