*Journal of Computer Science*

Original Research Paper

# Distributed Learning Automata Approach for Workflow Mining: Discovering Process Model Using Condensate Drops Method

**Vahideh Naderifar, Zarina Shukur and Shahnorbanun Sahran**

*Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, Bangi, Malaysia*

*Article History*
Received: 12-04-2019
Revised: 14-10-2019
Accepted: 27-11-2019

Corresponding Authors:
Zarina Shukur
Faculty of Information Science and Technology, Universiti Kebangsaan Malaysia, Bangi, Malaysia
Email: zarinashuku@ukm.edu.my

**Abstract:** An information system is a process of collecting, processing, storing and distributing information, which leads to efficient decision-making and control in organizations. Examples of information systems include classical management systems, systems for workflow management, systems for case handling and middleware. Information systems collect information concerning important people, locations and other important matters in an organization and store relevant events in some form of structure. Based on event logs, from information systems, the discovery of process models can be made automatically by process mining techniques, without having an a priori model. By learning from the event logs, process mining aims to discover, monitor and improve processes. This paper proposes a method to discover a process model based on distributed learning automata and the condensate approach. In this proposed method, each event in the log is called a drop, which had its condensate and can be combined with other condensates. Each drop is connected to other drops and become a larger drop. All of those drops would obtain reward if it represents sequence of an event log. The evaluation results demonstrated that the proposed method could detect various patterns in the event log and discover a more efficient process model in terms of fitness, total node and total path of the mined process model.

**Keywords:** Condensate Drops, Distributed Learning Automata, Process Mining, Event Log

## Introduction

Process mining is a relatively new field that aims to investigate methods of uncovering actual processes that took place in organizations. The process mining method can be used to understand 'as-is' processes in organizations and to subsequently improve or change them (van der Aalst and van Dongen, 2002; Xumin *et al*., 2018; Xiao *et al*., 2016). It includes process discovery, inspection, social network/organizational mining, automated simulation model construction, extension models, repair models and forecast cases (van der Aalst and van Dongen, 2002). The fundamental idea in process mining is to start with event logs that are recorded by an information system and gain knowledge from them (van der Aalst and van Dongen, 2002). Here, each event is associated with a process instance (case) and an activity (van der Aalst *et al*., 2003). The analysis of such event logs can provide insights into how processes take place and the extent to which actual processes differ from a normative process model. The events recorded by the information system track the completion of activities of specific types, among other things. For example, an event can report that a specific license application activity has been completed (Goedertier *et al*., 2009). From event logs, roles in the organization can be discovered and these roles are used to establish relationships between individuals and activities (van der Aalst and van Dongen, 2002). Figure 1 illustrates the workflow of process mining.

There are three types of process mining from event logs, namely discovery, conformance and enhancement. Discovery techniques take an event log and builds a model with no prior assumption or information (van der Aalst *et al*., 2003). A new process model can be constructed automatically based on the behavior derived from the event log.

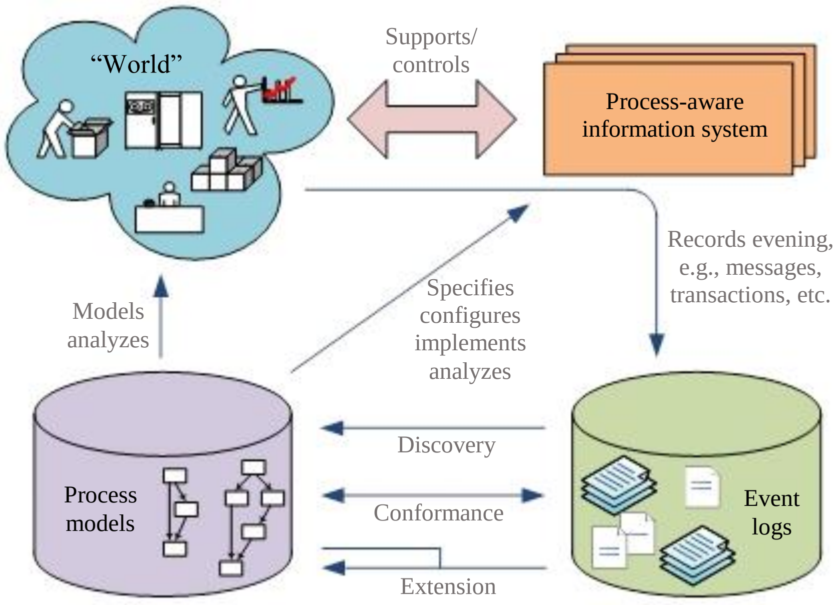

"World"
Supports/ controls
Process-aware information system
Records evening, e.g., messages, transactions, etc.
Models analyzes
Specifies configures implements analyzes
Process models
Discovery
Conformance
Extension
Event logs

**Fig. 1:** Graphic representation of process mining workflow (Source: van der Aalst and van Dongen, 2002)

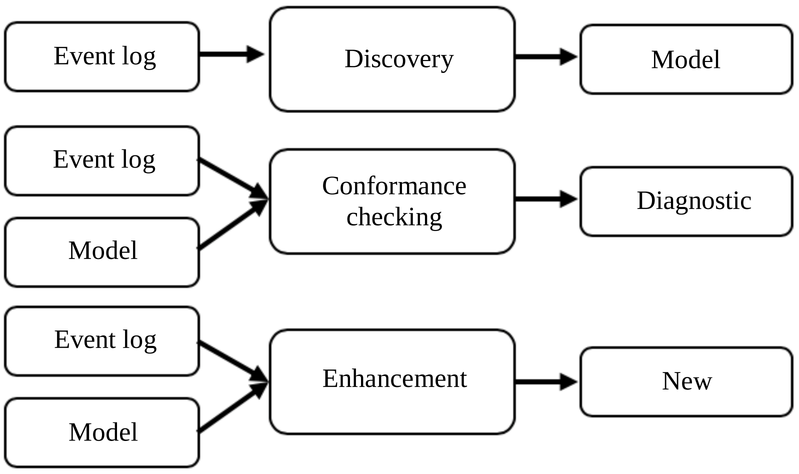


**Fig. 2:** Overview of three types of process mining based on input and output (Source: van der Aalst *et al*., 2003)

Conformance checking techniques compare an existing process model with the actual process derived from the event log and checks whether actual process conforms with the existing model. Lastly, enhancement techniques can be applied to extend or improve the model based on actual events. Figure 2 shows the three types of mining concept.

Most process mining algorithms have limitations. The constructs that cannot be mined by all techniques are loops and duplicate tasks (van der Aalst *et al*., 2010). As a result, existing process mining algorithms encounter problems with incorrect control-flow constructs. This paper proposes a method of process model discovery that can simultaneously overcome the loop and duplicate challenges.

## Related Works

Many researchers investigated the problem of process discovery. This study uses a specific website, www.processmining.org, that gives a complete overview of the entire process mining research area. Among the earliest researchers that investigated the mining of process models were Agrawal *et al*. (1998). Their mined model demonstrated the dependencies in the log between tasks. Their algorithm could not overcome duplicate tasks, loops, or a re-labeling process because the algorithm assumed that a task appeared only once in a process instance. Cook and Wolf (1998) have discovered models of software processes from data in event logs. Herbst (2000) was one of the first researchers who investigated processes with duplicate tasks. However, their proposed algorithm was unable to solve non-free-choice problems. Schimm (2000; 2002; 2003; 2004) discovered an algorithm that assumed that tasks had a starting and a completion event. Schimm's approach did not exactly tackle duplicates and non-free-choice during mining. Greco *et al*. (2004; 2005; 2006) discovered an algorithm based on a hierarchical tree of process models. Greco's algorithm described the event log at different levels of abstraction. It could tackle non-free-choice and invisible tasks, but it had problems with loops and duplicate tasks.

van der Aalst and Song (2004) compared extracting process models from data with the process of distillation. They developed an α-algorithm. However, it was unable to tackle non-free-choice. Additionally, since the α-algorithm operated based on sets, it was unable to mine models with duplicate tasks. van der Aalst *et al*. (2003) described two types of workflow: Meta-models that are graphs and block-oriented models. Each of the models had its language and graphical representation. Weijters *et al*. (2003; 2005; 2007) introduced an extension of the α-algorithm. Their approach was similar to the approach of Cook and Wolf (1998) because their algorithm was based on binary relationships and could not tackle non-free-choice constructs as well. Currently, this algorithm is implemented as the Heuristics miner plug-in in the ProM framework tool.

Alves *et al*. (2007) proposed a genetic algorithm to tackle some of the control flow problems, that is, the duplicates and non-free-choice loops. Goedertier *et al*. (2009) described a generation of artificial negative events. In this area, Cattafi *et al*. (2010) proposed an incremental declarative approach in which processes could change over time and the approach was able to revise the mined model by considering newer process traces and the possible deviations they might bring. From their method, Cattafi *et al*. (2010) obtained a new process discovery algorithm. This algorithm could address loops, duplicate activities and non-free-choice constructs. Wen *et al*. (2004; 2006; 2009) implemented two extensions for the α-algorithm and the α++ algorithm. Their approach could overcome problems like loops, concurrency, non-free-choice and noise. Burratin and Sperduti (2010a) presented Heuristics++ miner, an approach similar to the approach of Weijters *et al*. (2003; 2005; 2007) and van der Aalst *et al.* (2003) to allow the duration of process activities to become part of the process mining parameter set. Burratin and Sperduti (2010b) also explored the area of parameter setting with their Heuristics++ miner and formulated an approach potentially applicable to a wide range of process mining approaches.

van der Aalst *et al*. (2010) proposed a two-step approach for transition systems and regions to discover process models from event logs. This proposed method could offer a balance between under-fitting and over-fitting problems and address duplicate tasks and non-free-choice. They claimed that none of the existing techniques allowed a balance between over-fitting and under-fitting. The first of the two-step approach, used a configurable approach and constructed a transition system. The second step used the "theory of regions" and the model was synthesized. The model has been implemented in the ProM framework and could overcome many of the limitations of the traditional approach. Weidlich *et al*. (2011) proposed a technique called "behavioral profile". The problem with this approach is that it could not handle loops properly. The key idea is that a footprint can be based on observed and modeled behaviors.

Petri nets are most suitable for the machine learning approach because of their concurrent, asynchronous, distributed, parallel non-deterministic and stochastic properties. Leoni and van der Aalst (2013) presented a novel technique for data-aware process mining that was able to address process models with invisible transitions. Rozinat and van der Aalst (2006) proposed an incremental approach for checking process model conformance and event log. Their method checked the fit between the log and the model first and then the suitability of the model with the log. Adriansyah *et al*. (2011) provided a robust method for calculating conformance between a log and a process model. van der Aalst (2012) established a precise relationship between events and model elements. This relationship could be used to check conformance and analyze performance.

## *Control Flow Perspectives*

Depending on the type of data from an event, process mining has three types of perspectives, which are: Process perspective, organizational perspective and case perspective (van der Aalst *et al*., 2003). This paper focuses on the process perspective that centered on the control flow. The control flow characterizes all possible paths between tasks and shows given behaviors in the log via a diagram.

The control-flow perspective describes a process model by using a diagram. This diagram shows the flow from tasks in an event log. From this diagram, information about tasks, relationships between the tasks, which tasks are running, and which tasks are related to which case can be determined. Control-flow mining techniques must then be constructed from correct mining. From the aspect of common control flow, correct mining means that constructs can appear as a process model with each language notation. These are sequences, loops, concurrency, non-free-choice, duplicate tasks and invisible tasks (van der Aalst and van Dongen, 2002).

*Example 1. Driving License Process (A case study)*

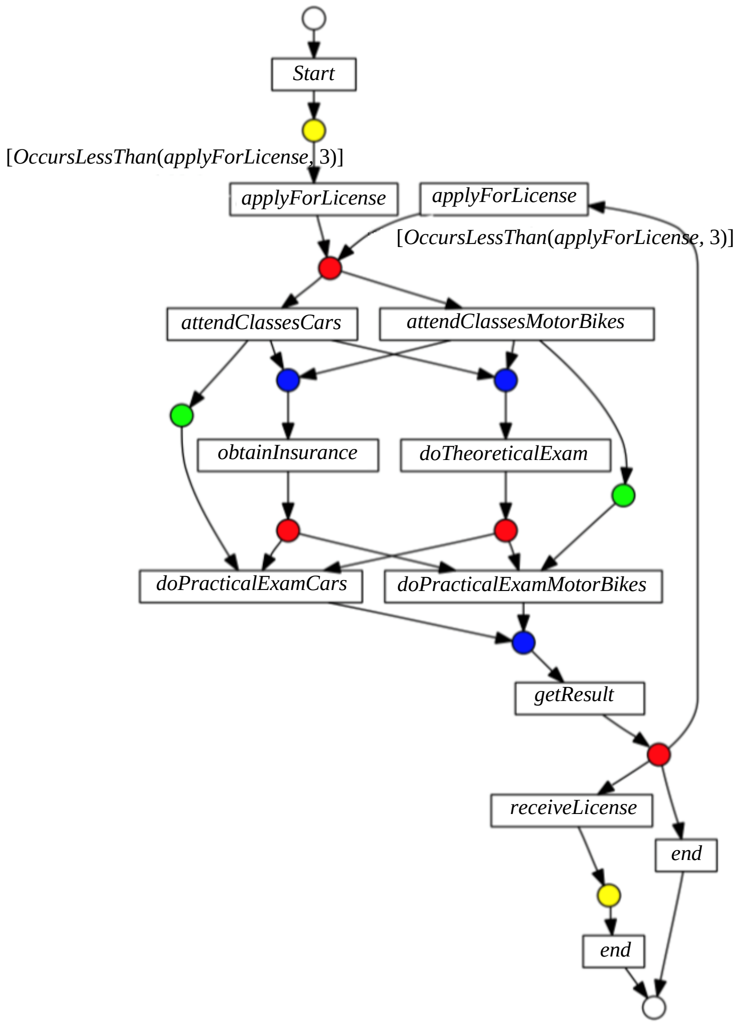


**Fig. 3:** Driving license process - petri net (Source: Goedertier *et al*., 2009)

| | *Activity* | *Precondition* |
|---|---|---|
| *a* | *start* | *true* |
| *b* | *applyForLicense* | $NS(a,b)$ |
| *b* | *applyForLicense* | $(NS(i,b) \wedge NS(i,j) \wedge NS(i,k)) \wedge OccursLessThan(b,3)$ |
| *c* | *attendClassesCars* | $NS(b,c) \wedge NS(b,d)$ |
| *d* | *attendClassesMotorBikes* | $NS(b,c) \wedge NS(b,d)$ |
| *e* | *obtainInsurance* | $NS(c,e) \vee NS(d,e)$ |
| *f* | *doTheoreticalExam* | $NS(c,f) \vee NS(d,f)$ |
| *g* | *doPracticalExamCars* | $(NS(f,g) \wedge NS(f,h)) \wedge (NS(e,g) \wedge NS(e,h)) \wedge NS(c,g)$ |
| *h* | *doPracticalExamMtrBikes* | $(NS(f,g) \wedge NS(f,h)) \wedge (NS(e,g) \wedge NS(e,h)) \wedge NS(d,h)$ |
| *i* | *getResult* | $NS(g,i) \vee NS(h,i)$ |
| *j* | *receiveLicense* | $NS(i,b) \wedge NS(i,j) \wedge NS(i,k)$ |
| *k* | *end* | $NS(j,k)$ |
| *k* | *end* | $NS(i,b) \wedge NS(i,j) \wedge NS(i,k)$ |

**Fig. 4:** Driving license process – activity preconditions (Source: Goedertier *et al*., 2009)

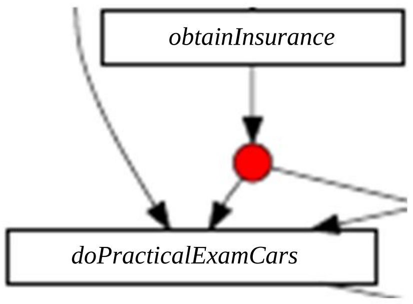


**Fig. 5:** Sequence connection

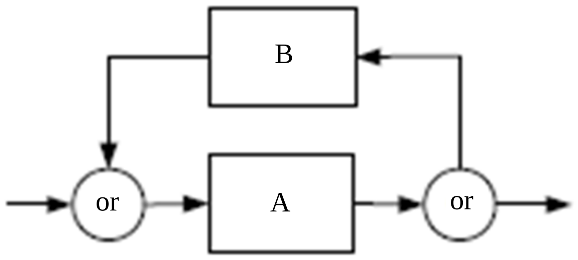


**Fig. 6:** The construct of a loop

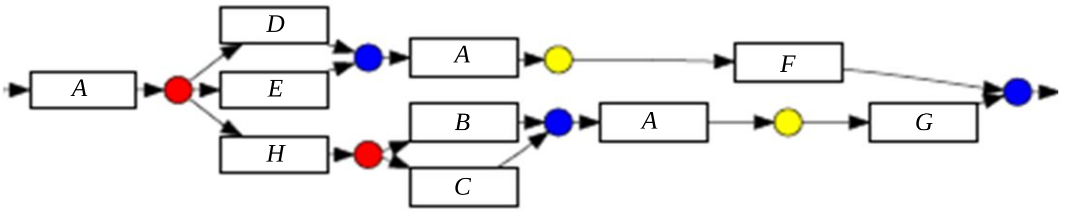


**Fig. 7:** The construct of a duplicate task

**Table 1:** Common mining challenges and techniques used

| Challenge approaches | Overcome duplicate tasks | Overcome mining loops |
|---|---|---|
| Data mining based | ● Herbst (2004) | ● Herbst (2004) |
| | ● Agrawal *et al*. (1998) | ● Agrawal *et al*. (1998) |
| Heuristic approach | | ● Weijters and van der Aalst (2003) |
| | | ● Burratin and Sperduti (2010a) |
| Soft computing algorithms | ● Alves de Medeiros *et al*. (2007) | ● Alves de Medeiros *et al*. (2007) |
| | | ● Günther and van der Aalst (2009) |
| Markova approach | | ● Kumaraguru (2013) |
| Other approaches | ● Rozinat and van der Aalst (2006) | ● Greco *et al*. (2006) |
| | ● Alves de Medeiros *et al*. (2007) | ● Alves de Medeiros *et al*. (2007) |
| | ● van der Aalst *et al*. (2010) | ● Cattafi *et al*. (2010) |
| | ● Goedertier *et al*. (2009) | ● Wen *et al*. (2009) |
| | ● Leoni and van der Aalst (2013) | ● Goedertier *et al*. (2009) |
| | | ● Leoni and van der Aalst (2013) |

A case study about Driving License process which is taken from Goedertier *et al*. (2009) will be used in this section. Fig. 3 shows the petri net diagrammatic model of the process, whereas Fig. 4 shows the model in the form activity and its precondition.

**Sequences** express which tasks are consecutive. Figure 5 shows the sequence control-flow of discovery.

**Loops** show a node or sequence path that may be executed several times as shown in Fig. 6.

**Duplicate** Tasks: There are two nodes with the same label in the process model in parallel or sequence models as shown in Fig. 7.

A duplicate task is accrued if the same activity is carried out according to different conditions. Algorithms for control flow process mining should be able to produce a correct structure.

### *Comparison between Existing Mining Models*

Even though a lot of effort has been made to discover process mining algorithms, there remain several challenges that are not addressed. The structural pattern challenges that are not effectively addressed are loops and duplicates. In this aspect, most process mining algorithms are still unable to mine duplicate tasks and loops. Table 1 shows techniques that have been used by the researchers to address these two main problems.

### *Learning Automata*

Tsypkin (1971) introduced learning automatics (LA) to solve the problems of determining the optimal parameter and applied the LA to the techniques of hill climbing. At the same time, Tsetlin (1973) began working on LA. Other researchers introduced the problems and found optimal action in stochastic automatons between permitted actions (Narendra and Viswanathan, 1972). However, most attempts in LA were due to Tsetlin (1973). Thereafter, Varshavski and Vorontsova (1963) presented LA with variable structures that updated its probability of actions and led to a decrease in the number of states in the comparison of deterministic automata. In this case, Fu and McMurtry (1966) made the first attempt for pattern recognition, parameter estimation and game theory. Najim and Poznyak (1994) have also presented several examples and applications of LA. Other LA applications include pattern recognition, graph partitioning and route planning. LA can be portrayed as an object with limited action sets. It randomly chooses one of the actions and sends it to an environment. The automaton can update its action probability based on an environment response and repeated procedures.

A learning automaton (LA) consists of two parts: (1) A stochastic automaton with a limited number of actions and a stochastic environment associated with the automaton. (2) A learning algorithm in which the automaton learns the best possible action with this action.

In practice, every action is transmitted to and evaluated by a potential environment before a stochastic automaton reacts. The stochastic automaton then uses this response as a response and selects its action for the next step. Figure 8 shows the relationship between stochastic and environmental automatons (Meybodi *et al*., 2004; Hadavi *et al*., 2014).

## *P-Model of Learning Algorithm*

All probability of past actions in any automatic form would be the same. For r-action automatons, the probability of action n is given by $P_i(n) = 1/r$, which is updated based on the reward or penalty in each repetition *i*. If $\alpha_i$ is selected between the other actions in these types of automatons, $P_i(\alpha_i)$ receives the desirable response, in which its probability increases and the opposite probabilities decrease. However, if the probabilities of $P_i(n)$ decrease, the remaining probabilities increase for unwanted answers. In any event, changes are made to the extent that the sum of $P_i(n)$ is equal to one. The following formula shows the desirable and undesirable answers.

Desirable answer:

$$P_i(n+1) = P_i(n) + a\left[1 - P_i(n)\right] \tag{1a}$$

$$P_j(n+1) = (1-a)P_j(n) \qquad \forall j, j \neq i \tag{1b}$$

Undesirable answer:

$$P_i(n+1) = (1-b)P_i(n) \tag{2a}$$

$$P_j(n+1) = (b/r-1) + (1-b)P_j(n) \quad \forall j, j \neq I. \tag{2b}$$

In Equations 1 and 2, *a* is a reward parameter, *b* is the penalty parameter and *r* is the number of actions that *a*, $b \in [0,1]$. Three states can be considered regarding the values of *a* and *b*:

- $L_{RP}$ (Linear Reward Penalty) when *a* = *b*, penalty and reward are both important
- $L_{ReP}$ (Linear Reward Epsilon Penalty) when *b* is lower than a (*a*>>*b*), reward is much more important than penalty. However, penalty still needs to be given
- $L_{RI}$ (Linear Reward Inaction) when *b* equals zero (*b = 0*), penalty is not taken into consideration

## *Distributed Learning Automata*

A distributed learning automaton (DLA) is a network of LAs working together to solve problems. Only one LA is active in colleagues ' LAs at any time. The number of LAs in a single DLA is equal to the number of actions that any of the other LAs connected to it can carry out. The choice of an action by the LAs in this network results in the isomorphic activation of other LAs connected to the network. Figure 9 shows the concept of a DLA. The DLA network is modeled by a graph, each vertex being an LA. The arrows between $LA_i$ and $LA_j$ in this graph show that the selection of actions $\alpha_j^i$ by $LA_i$ results in $LA_j$ being activated.

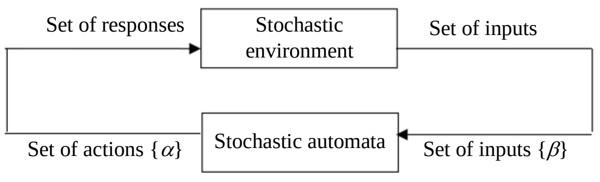


**Fig. 8:** Stochastic learning automata

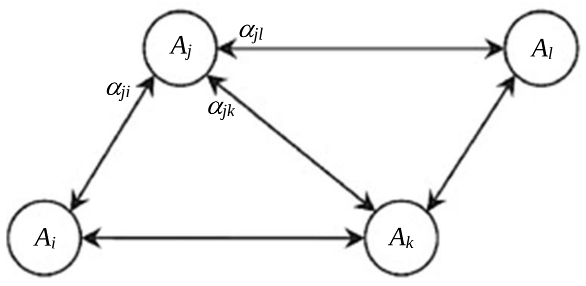


**Fig. 9:** Distributed learning automaton

## Research Framework

The outcome of the process mining shows all behaviors seen in the system. An outline of the research framework is given by the following basic steps and Fig. 10:

1. Information extraction from event logs. Several standard event logs act as important factors for process mining. For discovering rules in an organization, the event log is used. These rules are a connection factor between individuals and activities
2. Model discovering based on the event logs. Model discovering determines which perspectives of the process can be discovered. If the model is based on the event log, then the control-flow perspective can be mined
3. The learning method is to improve an existing process mining model based on the event logs by using reinforcement learning and its family
4. Performance analysis will be performed after discovering the control flow using the existing process mining model. The proposed model can be used to analyse the performance of process mining in terms of fitness, total node and total path

## Proposed Method

Distributed learning automata as a learning automata family are used in the proposed method as the environment. Each row of the event log is divided into pair-wise. Each pair-wise that is called a drop will be spread in the environment. Each node of the DLA is one learning automaton. A condensate drop is a sequence with the same elements, such as (a, c) (c, d) where c is the same between those two drops. In the first stage, the first and endpoints of each pair-wise in a row of the event log are determined by the sign. All drops are distributed in the environment; the environment is the DLA. Drops are moving on the DLA. Each drop has its condensate and can be combined with other condensate drops. After combination, they create large drops. The reinforcement value for each pair-wise in a row of the event log can be calculated as DLA conceptual. Table 2 is a result of running DLA on an event log. This paper sets the reward and penalty rate equal to 0.5. The following explains the step-by-step of how DLA works based on each pair-wise in a row of Table 4.

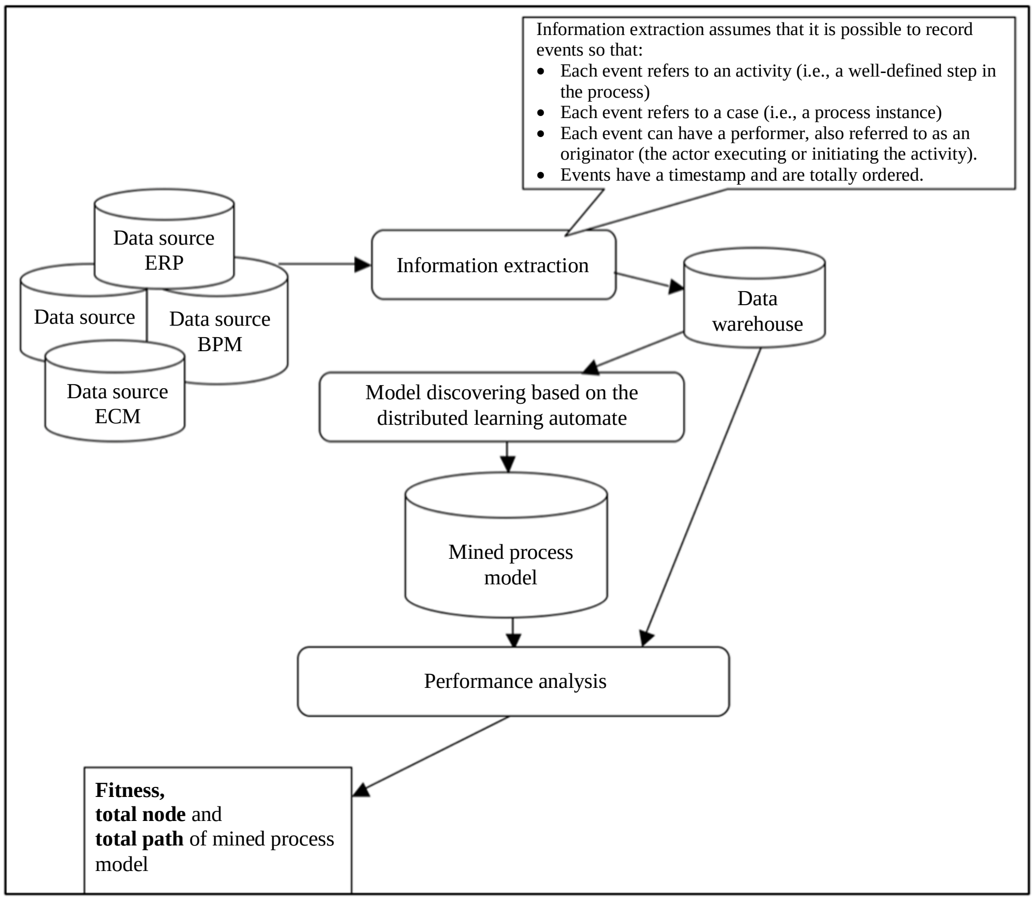
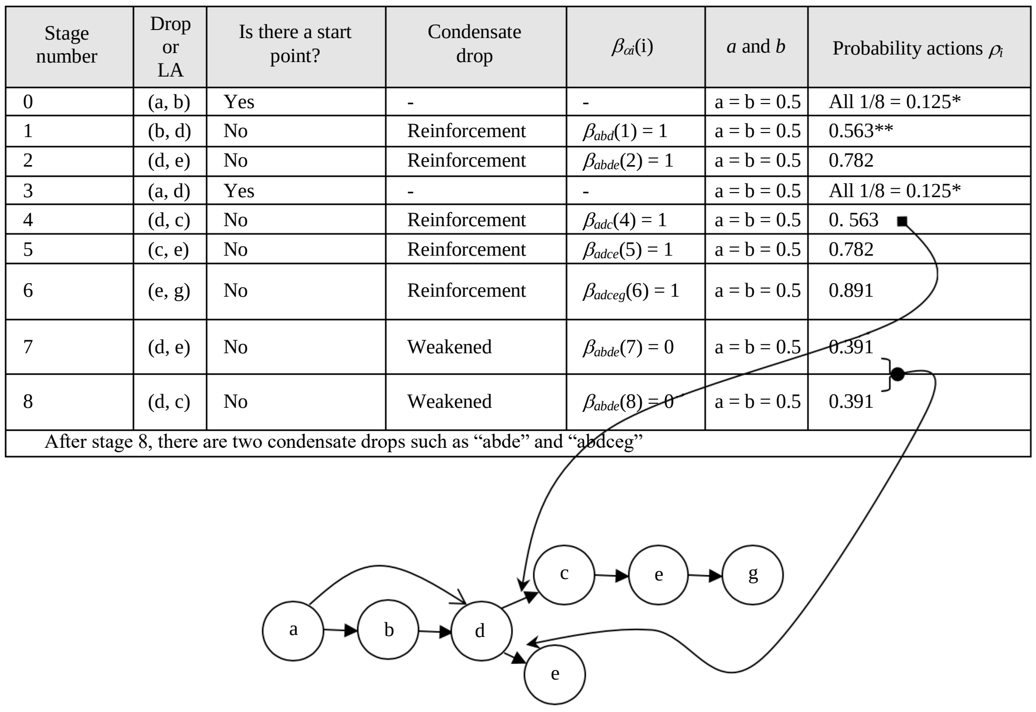

Information extraction assumes that it is possible to record events so that:
Each event refers to an activity (i.e., a well-defined step in the process)
Each event refers to a case (i.e., a process instance)
Each event can have a performer, also referred to as an originator (the actor executing or initiating the activity).
Events have a timestamp and are totally ordered.
Data source ERP
Data source
Data source BPM
Data source ECM
Information extraction
Data warehouse
Model discovering based on the distributed learning automate
Mined process model
Performance analysis
Fitness, total node and total path of mined process model

**Fig. 10:** Basic steps of the research methodology and the proposed method



| Stage number | Drop or LA | Is there a start point? | Condensate drop | $\beta_{\alpha i}(i)$ | *a* and *b* | Probability actions $\rho_i$ |
|---|---|---|---|---|---|---|
| 0 | (a, b) | Yes | - | - | a = b = 0.5 | All 1/8 = 0.125* |
| 1 | (b, d) | No | Reinforcement | $\beta_{abd}(1) = 1$ | a = b = 0.5 | 0.563** |
| 2 | (d, e) | No | Reinforcement | $\beta_{abde}(2) = 1$ | a = b = 0.5 | 0.782 |
| 3 | (a, d) | Yes | - | - | a = b = 0.5 | All 1/8 = 0.125* |
| 4 | (d, c) | No | Reinforcement | $\beta_{adc}(4) = 1$ | a = b = 0.5 | 0. 563 |
| 5 | (c, e) | No | Reinforcement | $\beta_{adce}(5) = 1$ | a = b = 0.5 | 0.782 |
| 6 | (e, g) | No | Reinforcement | $\beta_{adceg}(6) = 1$ | a = b = 0.5 | 0.891 |
| 7 | (d, e) | No | Weakened | $\beta_{abde}(7) = 0$ | a = b = 0.5 | 0.391 |
| 8 | (d, c) | No | Weakened | $\beta_{abde}(8) = 0$ | a = b = 0.5 | 0.391 |
| After stage 8, there are two condensate drops such as "abde" and "abdceg" | | | | | | |

**Fig. 11:** The condensate drop in the DLA model such as "abde" and "adceg", in which "dc" has been reinforced from "ad" and has been weakened from "abd"

**Table 2:** The DLA process on each pair-wise in a row of the event log

| Stage number | Drop or LA | Is there a start point? | Condensate Drop | $\beta_{\alpha i}(i)$ | *a* and *b* | Probability actions $\rho_i$ |
|---|---|---|---|---|---|---|
| 0 | (a, b) | Yes | - | - | a = b = 0.5 | All 1/8 = 0.125* |
| 1 | (b, d) | No | Reinforcement | $\beta_{abd}(1) = 1$ | a = b = 0.5 | 0.563** |
| 2 | (d, e) | No | Reinforcement | $\beta_{abde}(2) = 1$ | a = b = 0.5 | 0.782 |
| 3 | (a, d) | Yes | - | - | a = b = 0.5 | All 1/8 = 0.125* |
| 4 | (d, c) | No | Reinforcement | $\beta_{adc}(4) = 1$ | a = b = 0.5 | 0. 563 |
| 5 | (c, e) | No | Reinforcement | $\beta_{adce}(5) = 1$ | a = b = 0.5 | 0.782 |
| 6 | (e, g) | No | Reinforcement | $\beta_{adceg}(6) = 1$ | a = b = 0.5 | 0.891 |
| 7 | (d, e) | No | Weakened | $\beta_{abde}(7) = 0$ | a = b = 0.5 | 0.391 |
| 8 | (d, c) | No | Weakened | $\beta_{abde}(8) = 0$ | a = b = 0.5 | 0.391 |

After stage 8, there are two condensate drops such as "abde" and "abdceg"
* There are 8 drops in this example.
** Reward is obtained according to Equation 2a.

**Table 3:** A fragment (Case id #6) of some event logs (each line corresponds to an event)

| | Properties | | | | |
|---|---|---|---|---|---|
| Event id | Timestamp | Activity | Resource | Cost | … |
| 35654871 | 06-01-2011:15.02 | Register request | Mike | 50 | … |
| 35654873 | 06-01-2011:16.06 | Examine casually | Ellen | 400 | … |
| 35654874 | 07-01-2011:16.22 | Check ticket | Mike | 100 | … |
| 35654875 | 07-01-2011:16.52 | Decide | Sara | 200 | … |
| 35654877 | 16-01-2011:11.47 | Pay compensation | Mike | 200 | … |
| … | … | … | … | … | … |

*Example 1*

By reading the first row of the event log (Table 4), some drops such as (a, b) (b, d) (d, e) (e, h) will be created for the first time in the DLA. Each drop is a learning automaton (LA). For the second row of the event log (stage numbers 3 to 6), there are (a, d) (d, c) (c, e) (e, g) drops or LAs. The creation of the first condensate drop is called the first reinforcement of the neighbor if the reward is obtained by another drop; otherwise, the penalty is obtained. This process will be conducted for each drop that has the same element between two drops. Figure 11 shows the condensate drop in the DLA model.

The distributed learning automata lead to reaching better results to obtain the process model using reinforced and weakened drops of the event logs, as shown in Fig. 11.

The second step is investigated to determine whether there are large drops created in the event log. If yes, then all drops will obtain rewards and become the biggest drop concerning the combination rules; otherwise, all connections will be cut off and changed to the same initial drops in an environment. This combination will be continued until all condensate drops are combined and obtain the rewards. In the end, several large drops cannot be combined. These drops are the process models and event logs can be mined by them. Table 3 shows the raw event logs and Table 4 shows the sequence of the event log that will be used in this paper.

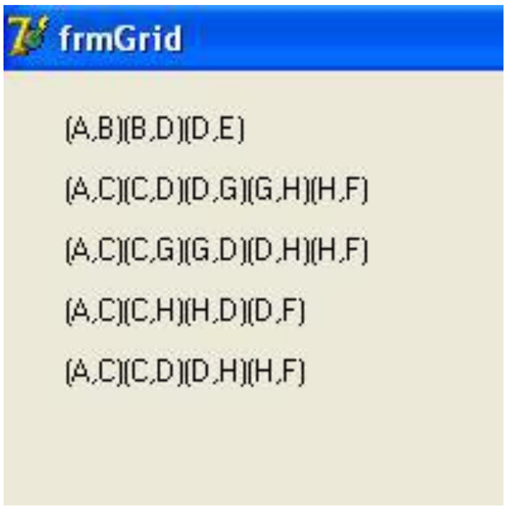
Table 3 will be converted to Table 4 by inserting the activity in a sequence of letters as below:

a = register request, b = examine thoroughly,
c = examine casually, d = check ticket, e = decide,
f = reinitiate request, g = pay compensation and
h = reject request.

Figure 12 shows a sample of the sequences of event log.

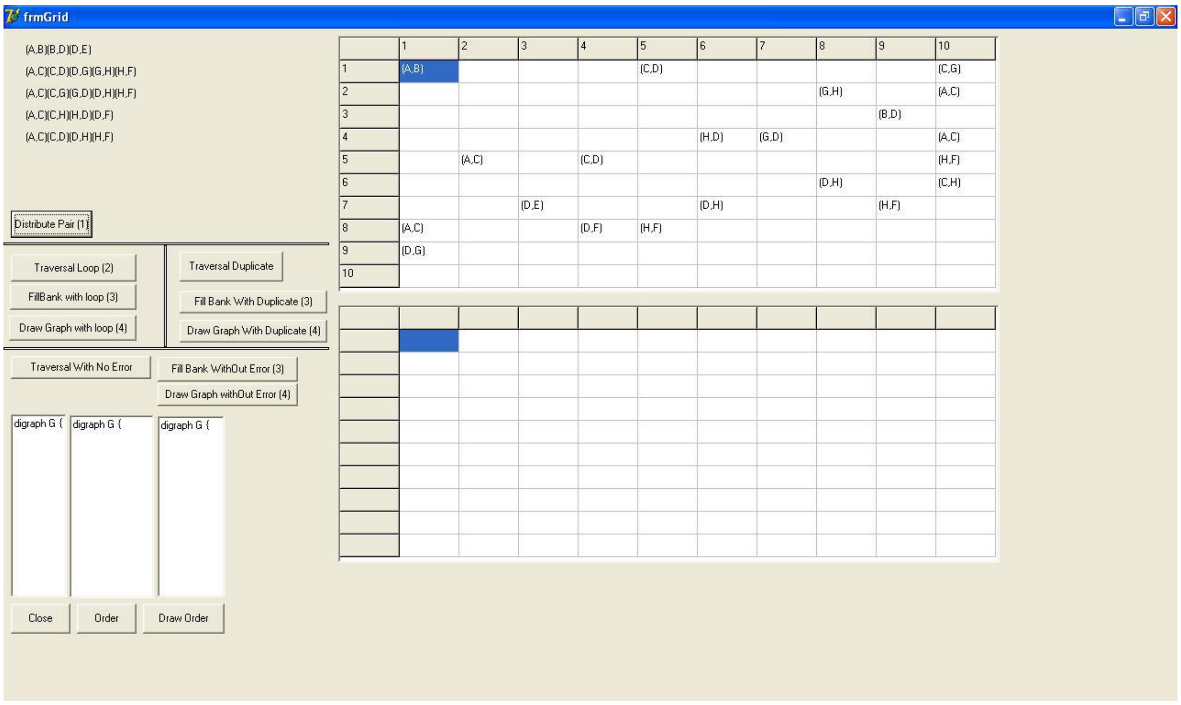
**Table 4:** A more compact display of the event log

| Case id | Trace |
|---|---|
| 1 | *(a,b,d,e,h)* |
| 2 | *(a,d,c,e,g)* |
| 3 | *(a,c,d,e,f,b,d,e,g)* |
| 4 | *(a,d,b,e,h)* |
| 5 | *(a,c,d,e,f,d,c,e,f,c,d,e,h)* |
| 6 | *(a,c,d,e,g)* |
| … | … |


frmGrid

(A,B)(B,D)(D,E)
(A,C)(C,D)(D,G)(G,H)(H,F)
(A,C)(C,G)(G,D)(D,H)(H,F)
(A,C)(C,H)(H,D)(D,F)
(A,C)(C,D)(D,H)(H,F)

**Fig. 12:** Sequences of the event log where each one is a drop



(A,B)(B,D)(D,E)
(A,C)(C,D)(D,G)(G,H)(H,F)
(A,C)(C,G)(G,D)(D,H)(H,F)
(A,C)(C,H)(H,D)(D,F)
(A,C)(C,D)(D,H)(H,F)

digraph G {
digraph G {
digraph G {

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 |
|---|---|---|---|---|---|---|---|---|---|---|
| 1 | (A,B) | | | | (C,D) | | | | | (C,G) |
| 2 | | | | | | | | (G,H) | | (A,C) |
| 3 | | | | | | | | | (B,D) | |
| 4 | | | | | | (H,D) | (G,D) | | | (A,C) |
| 5 | | (A,C) | | (C,D) | | | | | | (H,F) |
| 6 | | | | | | | | (D,H) | | (C,H) |
| 7 | | | (D,E) | | | (D,H) | | | (H,F) | |
| 8 | (A,C) | | | (D,F) | (H,F) | | | | | |
| 9 | (D,G) | | | | | | | | | |
| 10 | | | | | | | | | | |

**Fig. 13:** The condensate drops that are distributed in the environment

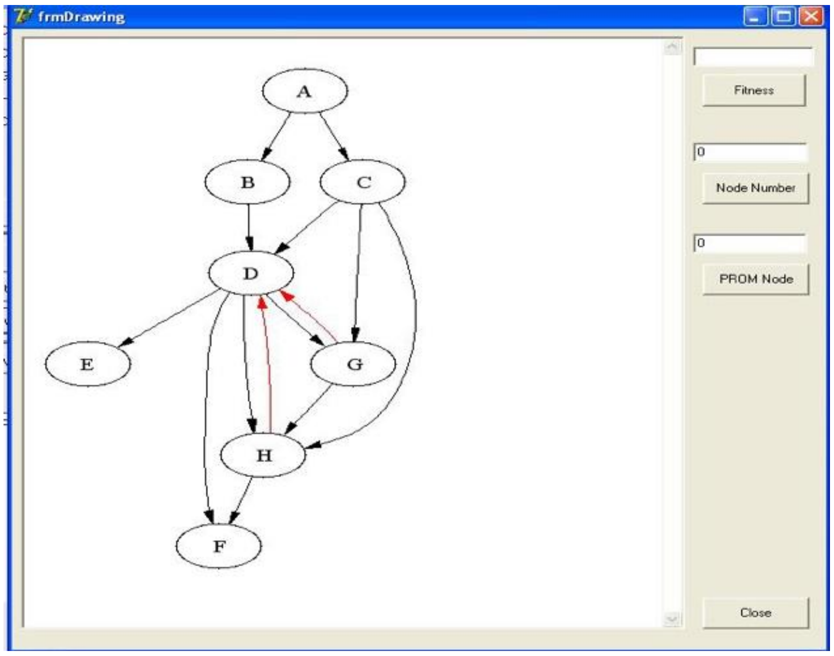


**Fig. 14:** The connections in a process model

The proposed method is based on the pair-wise fashion that will be studied step by step, as outlined below:

1. Each trace or sequence in the event log (Table 5) is divided into a pair-wise. Each pair is called a drop or a learning automaton that has its density, as shown in Fig. 12. All of the drops are distributed randomly on the DLA as an environment, as shown in Fig. 13 for the following event log
2. The first element of each sequence, as well as the last element, are indicated by a minus (-) and a plus (+) signs, respectively
3. Two pair-wise - or two LAs - will be merged into each other, if the other element is not similar. For example, (c, d), (d, c) cannot merge into each other because the c element is the same (although the d element is the same and those two drops are a condensate)
4. Based on the condensate drop rules, the same elements between condensate drops will be merged into each other
5. The sequence generated is investigated from the end state to the beginning state. If it is available in an event log, then all connections will become strange connections (Esmaeilpour *et al*., 2012; 2014) and all connections will obtain the reward. Finally, all drops in this connection will be seen as a drop as shown in Fig. 14
6. At the end, if there is a sequence that does not exist in the event log, then all of its connections will be cut off and the amount of the reward will reset (evaporating droplet).

The validity of the proposed model will be assessed in terms of fitness, total node and total path of the mined process model in the next section.

## Control-Flow Construct Study

The control-flow construct must be constructed from correct mining. These constructs are loops and duplicates that will be studied below.

### *Loop*

Variables are assigned to each pair-wise of the sequence in the event log that can be the number of the event log. After connecting two drops and if the numbers of the rows are similar, then there will be a cycle. Figure 15 shows the loop in the process model, while Fig. 16 shows the process model that has deleted the loop according to the proposed model.

### *Duplicate Tasks*

Regarding the assignment of variables to each row of the event log, when there is an event that is duplicated with another event in the appropriate event log, then that variable will be investigated and checked to which one it belongs. This variable can delete the duplicate event in the discovered process model as shown in Fig. 17.

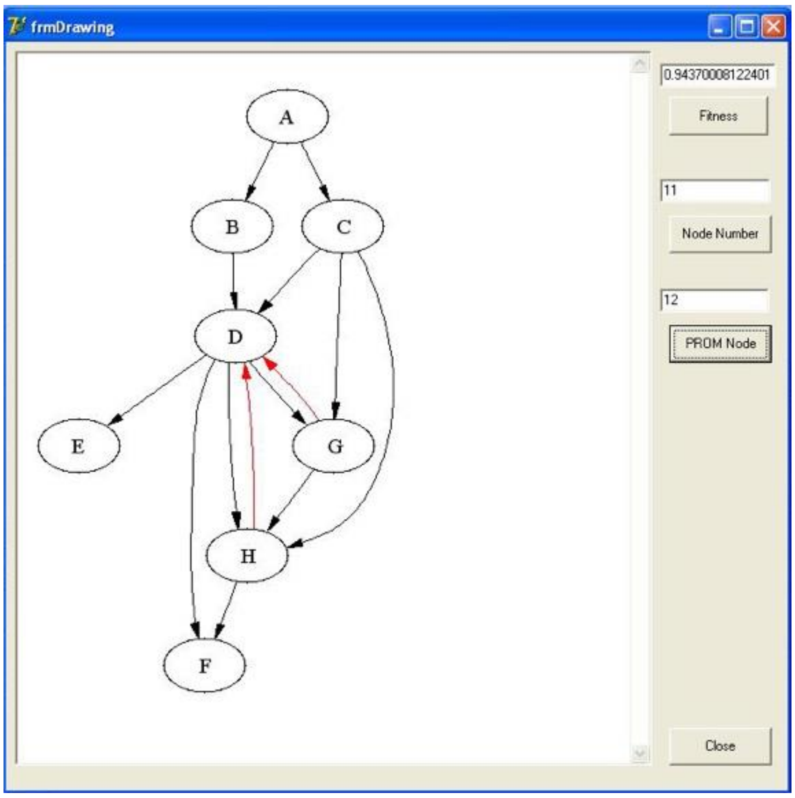


**Fig. 15:** The loop in the process model based on the event log of Fig. 12 and fitness equal to 0.943

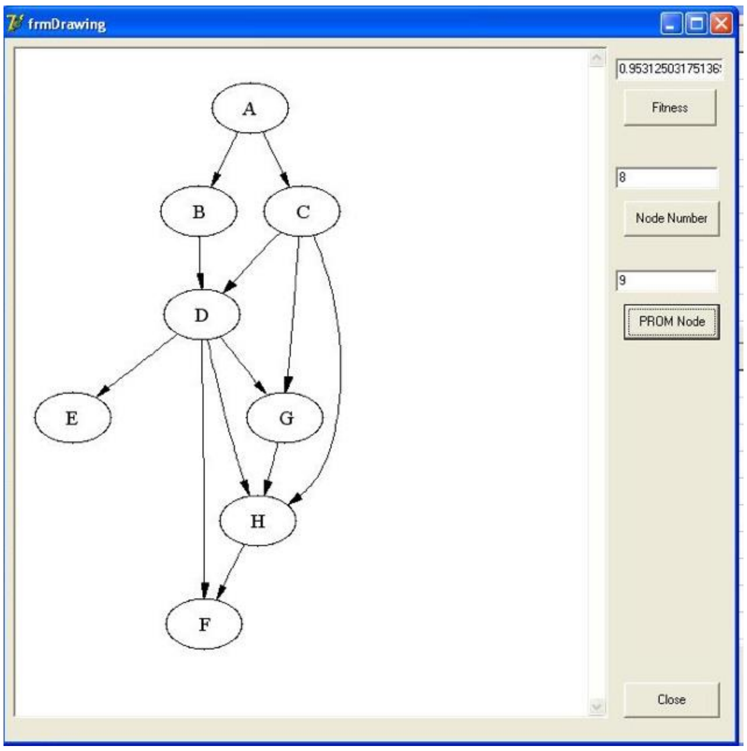


**Fig. 16:** The process model that has deleted the loop according to the proposed model, based on the event log of Fig. 12 and fitness equal to 0.943

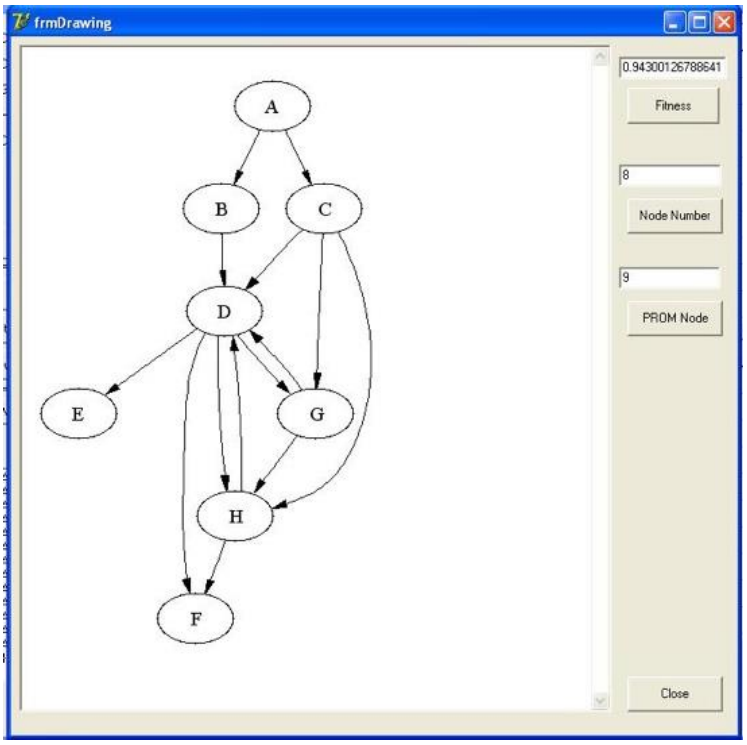


**Fig. 17:** The duplicate task before reconstruction based on the event log of Fig. 12 and fitness equal to 0.953

## Results and Discussion

An existing process model is compared to an event log of the same process after the control flow discovery. Conformity checks relate events in the event log to the process model activities and compare them. The aim is to find commonalities and differences between the behavior model and the behavior observed. Conformance checking is closely linked to measuring the fitness of a model that has been discovered and can also be used to evaluate and compare process discovery algorithms. The method proposed focuses on the problem of process discovery, in which business process models are discovered using event-based data generated by information systems. The motivation for representing process discovery as reinforcement learning is that it allows for the use of well-known learning methods and assessment techniques in the machine learning community. Hence, this paper has introduced the practical evaluation of the process discovery technique using several learning methods. In this work, distributed learning automatons are used with the *P* learning model and the $L_{RP}$ learning algorithm and *pi* is initialized at each stage equal to *1/r*, where *r* is the number of pairwise sequences.

Based on the event log, strong condensate drops are extracted. To carry out this task, DLA is used. DLA can combine two condensate drops (two LAs) and have an impact on them. An LA of DLA is one drop. Neighborhoods are a group of drops with a condensate of their own. Each LA records each neighbor's probability actions. In each LA, $\alpha_i$ is a set of condensate drops and a reward will be given ($\beta_i(n) = 1$) if "a condensate drop attracts another drop and becomes a new condensate drop". The neighbor rate is equal to the occurrence of a similar neighbor in any drop. A neighbor of DLA will be created if the drop is similar in terms of density.

The proposed reward will increase probability actions. If ($\beta_i(n) = 1$), then one drop gets the reward and uses Equation 1a. In this section, the reward and penalty rate are equal to *a* = 0.5. Reinforcement is as important as weakness and the $L_{RP}$ learning model is used. Initially, all drops have the following probability action value:

$$\forall i, i <= n \quad \rho_i = 1/n$$

where, *n* is the number of pair-wise sequences.

The system features CPU 2.0 GHz Core2Duo with RAM 2 GB, Windows XP/SP2 and Visual Basic. Net Express Edition 2008 programming language are used to do the simulation (for the proposed model), the ProM5.2 framework tool (ProM, 2010) and Log_L1.xml, Log_L2.xml and Log_L3.xml event logs that have been used for comparison of the proposed method and other models. Table 6 to 8 show the experimental result of the proposed model on three well-known models: Genetic Miner, Heuristic Miner and α++ algorithm.

The evaluation criterion, as noted above, is fitness. The measured fitness is determined by replaying the model log. To replay the log in the model, the replay of each event trace begins with the marking of the initial place in the model and the transitions belonging to the events recorded in the trace are fired one after the other. During this process, the number of tokens that had to be produced artificially and the number of tokens left in the model are counted. Only if no token was left or missing would the fitness measure be evaluated as 1.0, which indicates 100% fitness. Fitness reflects the extent to which the traces of the event can be linked to the process model's execution paths. Thus, if $f$ = 1, the log can be parsed without any error by the model. The fitness metric $f$ based on the token is formalized as follows:

$$f = \frac{1}{2}\left(1 - \frac{\sum_{i=1}^{k} n_i m_i}{\sum_{i=1}^{k} n_i c_i}\right) + \frac{1}{2}\left(1 - \frac{\sum_{i=1}^{k} n_i r_i}{\sum_{i=1}^{k} n_i p_i}\right) \tag{3}$$

where, $k$ is the number of different traces of events from an aggregate log $L$, which is a multi-set of all distinguished traces. $n_i$, the number of events traces $T_i$ ($1 \le i \le k$) appearing in the given log L, $m_i$ is the number of missing tokens, $r_i$ is the number of remaining tokens, $c_i$ is the number of consumed tokens and $p_i$ is the number of tokens produced during the log replay of the current trace. For all $i$, $m_i \le c_i$ and $r_i \le p_i$ and therefore, $0 \le f \le 1$. The maximum fitness measurement value is used as an assessment criterion, i.e., $f$ = 1. The following event logs have been used for evaluation (ProM, 2010) (Sahlabadi *et al.*, 2014; 2017).

Other evaluation criteria are the total node and the total path of the mined process model after modifying. Total node shows the total stage to reach the sequences of an event log. Less total node with more fitness shows the simplicity of the model. On the other hand, the total path shows the total way to reach the end of an event log from the start point. Less total path with more fitness shows the simplicity of the model too.

Process discovery aims to give an idea of how the event log processes took place. This objective discovers processes an inherently descriptive learning problem. To assess the simplicity of the discovered process model, it is therefore justified to compare the learned process models in the same sequence from which the process models are learned in terms of a lower total node and a lower total fitness:

- Log_L1.xml consists of 4,371 process instances and 22,457 audit trail entries
- Log_L2.xml consists of 1,459 process instance and 7,748 audit trail entries
- Log_L3.xml consists of 61 process instances and 224 audit trail entries

From Table 5 to 8, for the Log_L1.xml event log, the proposed model was, on average, 7.4%, 7.5% and 7.5% better than the genetic algorithm, region miner and α++ algorithm from the aspect of without loop tasks, respectively. Moreover, the proposed model for the Log_L2.xml event log was, on average 12.5%, 11.7% and 10.5% better than other methods. For the Log_L3.xml event log, the proposed model was on average, 5.7%, 6.2% and 6.2% better than others in terms of fitness. The total node and total path were better than other methods as the results demonstrated that the proposed model had more fitness and less total node and total path as compared to other methods.

**Table 5:** The experimental result in terms of fitness with loop and without loop tasks

| | | | Fitness | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Genetic Algorithm | | Region Miner | | α++ Algorithm | | Proposed method | |
| Event log | Process Instance | Audit trail entries | Loop | Without loop | Loop | Without loop | Loop | Without loop | Loop | Without loop |
| Log_L1.xml | 4371 | 22457 | 0.762 | 0.765 | 0.862 | 0.896 | 0.792 | 0.807 | 0.892 | 0.897 |
| Log_L2.xml | 1459 | 7748 | 0.782 | 0.820 | 0.838 | 0.851 | 0.791 | 0.811 | 0.943 | 0.953 |
| Log_L3.xml | 61 | 224 | 0.902 | 0.921 | 0.921 | 0.933 | 0.811 | 0.832 | 0.953 | 0.953 |

**Table 6:** The experimental result in terms of total node and total path of the mined process model with loop and without loop tasks

| | Total node | | | | | | | | Total path | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | aGenetic algorithm | | Region miner | | α++ Algorithm | | Proposed methodal | | Genetic gorithm | | Region Miner | | α++ algorithm | | Proposed method | |
| Event Log | L | WL | L | WL | L | WL | L | WL | L | WL | L | WL | L | WL | L | WL |
| Log_L1.xml | 11 | 8 | 11 | 8 | 11 | 8 | 11 | 8 | 9 | 7 | 9 | 8 | 9 | 7 | 9 | 7 |
| Log_L2.xml | 11 | 7 | 11 | 8 | 11 | 8 | 11 | 8 | 25 | 8 | 32 | 10 | 26 | 8 | 23 | 10 |
| Log_L3.xml | 13 | 8 | 11 | 9 | 13 | 8 | 13 | 8 | 18 | 11 | 45 | 13 | 21 | 11 | 18 | 11 |

**Table 7:** The experimental result in terms of fitness with duplicate and without duplicate tasks

| | | | Fitness | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|
| | | | Genetic algorithm | | Region miner | | α++ Algorithm | | Proposed method | |
| Event log | PI | Audit trail entries | Dplct | W/o Dplct | Dplct | W/o Dplct | Dplct | W/o Dplct | Dplct | W/o Dplct |
| Log_L1.xml | 4371 | 22457 | 0.762 | 0.768 | 0.862 | 0.898 | 0.792 | 0.813 | 0.892 | 0.902 |
| Log_L2.xml | 1459 | 7748 | 0.782 | 0.825 | 0.838 | 0.859 | 0.791 | 0.824 | 0.943 | 0.953 |
| Log_L3.xml | 61 | 224 | 0.902 | 0.921 | 0.921 | 0.933 | 0.811 | 0.832 | 0.953 | 0.958 |

Dplct: Duplicate. W/o Dplct: Without duplicate PI: Process Instance

**Table 8:** The experimental result in terms of total node and total path of the mined process model with duplicate and without duplicate tasks

| | Total node | | | | | | | | Total path | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Genetic algorithm | | Region miner | | α++ Algorithm | | Proposed method | | Genetic algorithm | | Region miner | | α++ Algorithm | | Proposed method | |
| Event log | D | WD | D | WD | D | WD | D | WD | D | WD | D | WD | D | WD | D | WD |
| Log_L1.xml | 11 | 7 | 11 | 8 | 11 | 7 | 11 | 7 | 9 | 7 | 9 | 8 | 9 | 7 | 9 | 7 |
| Log_L2.xml | 11 | 8 | 11 | 8 | 11 | 8 | 11 | 8 | 24 | 10 | 30 | 10 | 25 | 10 | 23 | 10 |
| Log_L3.xml | 13 | 8 | 11 | 8 | 13 | 8 | 13 | 8 | 18 | 11 | 31 | 12 | 21 | 11 | 18 | 11 |

D: Duplicate WD: Without Duplicate

## Conclusion and Future Works

In this method, each row of the event log is a drop. Each drop is distributed in the distributed learning automata as an environment and can move in the environment. These drops have their density and can merge with other condensate drops. This method combined all the condensate drops to become larger drops. The proposed method has been implemented and compared with the genetic algorithm, region miner and α++ algorithm on the Log_L1.xml, Log_L2.xml and Log_L3.xml event logs and the results demonstrated that the proposed method had less total node and total path in terms of better fitness. Fitness of the proposed method was, on average, more than 7% better than that of other compared methods.

For future works, we hope to examine the ability of this method in terms of other quality attributes such as performance, accuracy and resource utilization. Besides that, it is also interesting to investigate the application of this method in other models, as well as to see its performance in real cases.

## Acknowledgment

This research was funded by the Malaysian Ministry of Higher Education with a research grant code of FRGS/1/2016/ICT01/UKM/01/1.

## Author's Contributions

**Vahideh Naderifar:** Contributes in the design, development and evaluation of the algorithm.

**Zarina Shukur:** Contributes in the design of the algorithm.

**Shahnorbanun Sahran:** Contributes in the evaluation of the algorithm.

## Ethics

The authors declare that there are no ethical issues associated with this research.

## References

Agrawal, R., D. Gunopulos and F. Leymann, 1998. Mining process models from work flow logs. Proceedings of the 6th International Conference on Extending Database Technology, Mar. 23-27, Springer, Berlin, pp: 469-483. DOI: 10.1007/BFb0101003

Adriansyah, A., B. van Dongen and W. van der Aalst, 2011. Conformance checking using cost-based fitness analysis. Proceedings of the 15th International Enterprise Distributed Object Computing Conference, Aug. 29-Sept. 2, IEEE Xplore Press, Helsinki, Finland, pp: 55-64. DOI: 10.1109/EDOC.2011.12

Alves de Medeiros, A.K., A. Guzzo, G. Greco, W.M.P. van der Aalst and A.J.M.M. Weijters 2007. Process mining based on clustering: A quest for precision. Proceedings of the International Conference on Business Process Management, Sept. 25-28, Springer, Brisbane, Australia, pp: 17-29. DOI: 10.1007/978-3-540-78238-4_4

Burratin, A. and A. Sperduti, 2010a. Heuristics miner for time intervals. Proceedings of the European Symposium on Artificial Neural Networks, Computational Intelligence and Machine Learning, Apr. 28-30, Bruges, Belgium, pp: 41-46.

Burratin, A. and A. Sperduti, 2010b. Automatic determination of parameters' values for Heuristics Miner++. Proceedings of the IEEE Congress on Evolutionary Computation, Jul. 18-23, Barcelona, Spain. DOI: 10.1109/CEC.2010.5586208

Cattafi, M., E. Lamma, F. Riguzzi and S. Storari, 2010. Incremental Declarative Process Mining. In: Smart Information and Knowledge Management, Szczerbicki, E. and N.T. Nguyen (Eds.), Springer, Berlin, pp: 103-127.

Cook, J.E. and A.L. Wolf, 1998. Discovering models of software processes from event-based data. ACM Trans. Software Eng. Methodol., 7: 215-249. DOI: 10.1145/287000.287001

Esmaeilpour, M., V. Naderifar and Z. Shukur, 2012. Cellular learning automata for mining customer behaviour in shopping activity. Int. J. Innovative Comput. Inform. Control, 8: 2491-2511.

Esmaeilpour, M., V. Naderifar and Z. Shukur, 2014. Design pattern mining using distributed learning automata and DNA sequence alignment. Plos One J., 9: 1-12. DOI: 10.1371/journal.pone.0106313

Fu, K.S. and G.J. McMurtry, 1966. A study of stochastic automata as models of adaptive and learning controllers. IEEE Trans. Automatic Control, 11: 379-387. DOI: 10.1109/TAC.1966.1098374

Goedertier, S., D. Martens, J. Vanthienen and B. Baesens, 2009. Robust process discovery with artificial negative events. J. Machine Learn. Res., 10: 1305-1340.

Greco, G., A. Guzzo, L. Pontieri and D. Saccfia, 2004. Mining Expressive Process Models by Clustering Workflow Traces. Proceedings of the 8th Pacific-Asia Conference on Advances in Knowledge Discovery and Data Mining, May 26-28, Springer, Sydney, Australia, pp: 2-62. DOI: 10.1007/978-3-540-24775-3_8

Greco, G., A. Guzzo and L. Pontieri, 2005. Mining hierarchies of models: From abstract views to concrete specifications. Proceedings of the 3rd International Conference on Business Process Management, (BPM' 05), Springer, Berlin, pp: 32-47. DOI: 10.1007/11538394_3

Greco, G, A. Guzzo and L. Pontieri, 2006. Discovering expressive process models by clustering log traces. IEEE Trans. Knowl. Data Eng., 18: 1010-1027. DOI: 10.1109/TKDE.2006.123

Günther, C.W. and W.M.P. van der Aalst, 2009. Fuzzy mining: Adaptive process simplification based on multi-perspective metrics. Proceedings of the 5th International Conference on Business Process Management, Sept. 24-28, Springer, Brisbane, Australia, pp: 328-343. DOI: 10.1007/978-3-540-75183-0_24

Hadavi, N., M.J. Nordin and A. Shojaeipour, 2014. Lung cancer diagnosis using CT-scan images based on cellular learning automata. Proceedings of the International Conference on Computer and Information Sciences, Jun. 3-5, IEEE Xplore Press, Ipoh, Malaysia. DOI: 10.1109/ICCOINS.2014.6868370

Herbst, J., 2000. A machine learning approach to workflow management. Proceedings of the 11th European Conference on Machine Learning, May 31-Jun. 02, Springer, London, pp: 183-194. DOI: 10.1007/3-540-45164-1_19

Herbst, J., 2004. Workflow mining with InWoLvE. Comput. Industry, 53: 245-264. DOI: 10.1016/j.compind.2003.10.002

Kumaraguru, P.V., 2013. Machine learning approach for model discovery and process enhancement using process mining techniques. PhD Thesis, Dr. M.G.R. Educational and Research Institute, India.

Leoni, M. and W.M.P. van der Aalst, 2013. Data-aware process mining: Discovering decisions in processes using alignments. Proceedings of the 28th Annual ACM Symposium on Applied Computing, Mar. 18-22, ACM, Coimbra, Portugal, pp: 1454-1461. DOI: 10.1145/2480362.2480633

Meybodi, M.R., H. Beigy and M. Taherkhani, 2004. Cellular learning automata and its applications. J. Sci. Technol., 25: 54-77.

Najim, K. and A.S. Poznyak, 1994. Learning Automata: Theory Applications. 1st Edn., Pergamon Press, Oxford Pergamon, ISBN-10: 0080420249, pp: 225.

Narendra, K.S. and R. Viswanathan, 1972. Learning models using stochastic automata. Proceedings of the International Conference of Cybernetics and Society, (CCS' 72), Washington DC.

ProM, 2010. ProM process mining framework.

Rozinat, A. and W.M.P. van der Aalst, 2006. Decision mining in ProM. Proceedings of the 4th International Conference on Business Process Management, Sept. 05-07, Springer, Vienna, Austria, pp: 420-425. DOI: 10.1007/11841760_33

Sahlabadi, M., R.C. Muniyandi and Z. Shukur, 2014. Detecting abnormal behavior in social network websites by using a process mining technique. J. Comput. Sci., 10: 393-402. DOI: 10.3844/jcssp.2014.393.402

Sahlabadi, M., A. Sahlabadi, R.C. Muniyandi and Z. Shukur, 2017. Evaluation and extraction factual software architecture of distributed system by process mining techniques. Asia-Pacific J. Inform. Technol. Multimedia, 6: 77-90.

Schimm, G., 2003. Mining most specific workflow models from event-based data. Proceedings of the International Conference on Business Process Management, Jun. 26-27, Springer, Eindhoven, The Netherlands, pp: 25-40. DOI: 10.1007/3-540-44895-0_3

Schimm, G., 2004. Mining exact models of concurrent workflows. Comput. Industry, 53: 286-293. DOI: 10.1016/j.compind.2003.10.003

Schimm, G., 2000. Generic linear business process modeling. Proceedings of the Workshops on Conceptual Modeling Approaches for E-Business and the World Wide Web and Conceptual Modeling: Conceptual Modeling for E-Business and the Web, Oct. 09-12, Springer, Utah, USA, pp: 31-39. DOI: 10.1007/3-540-45394-6_4

Schimm, G., 2002. Process miner - a tool for mining process schemes from event-based data. Proceedings of the European Conference on Logics in Artificial Intelligence, Sept. 23-26, Springer, Cosenza, Italy, pp: 525-528.

Tsetlin, M.L., 1973. Automata Theory and Modeling of Biological Systems. 1st Edn., Academic Press, New York, ISBN-10: 0127016503, pp: 288.

Tsypkin, Y.Z., 1971. Adaptation and Learning in Automatic Systems. 1st Edn., Academic Press, New York, ISBN-10: 0080955827, pp: 290.

van der Aalst, W.M.P. and M. Song, 2004. Mining social networks: Uncovering interaction patterns in business processes. Proceedings of the 2nd International Conference on Business Process Management, Jun. 17-18, Potsdam, Germany, pp: 244-260. DOI: 10.1007/978-3-540-25970-1_16

van der Aalst, W.M.P., V. Rubin, H.M.W. Verbeek, B.F. van Dongen and E. Kindler *et al*., 2010. Process mining: A two step approach to balance between under-fitting and over-fitting. Software Syst. Modell., 9: 87-111. DOI: 10.1007/s10270-008-0106-z

van der Aalst, W.M.P., B.F. van Dongen, J. Herbst, L. Maruster and G. Schimm *et al*., 2003. Workflow mining: A survey of issues and approaches. Data Knowl. Eng., 47: 237-267.

van der Aalst, W.M.P. and B.F. van Dongen, 2002. Discovering work of performance models from timed logs. Proceedings of the International Conference on Engineering and Deployment of Cooperative Information Systems, (CIS' 02), Springer, Berlin, pp: 45-63. DOI: 10.1007/3-540-45785-2_4

van der Aalst, W.M.P., 2012. Decomposing process mining problem using passages. Proceedings of the 33rd International Conference on Application and Theory of Petri Nets, Jun. 25-29, Springer, Hamburg, Germany, pp: 72-91. DOI: 10.1007/978-3-642-31131-4_5

Varshavski, V.I. and I.P. Vorontsova, 1963. On the behavior of stochastic automata with variable structure. Automatika Telemechanika, 24: 353-360.

Weidlich, M., J. Mendling and M. Weske, 2011. Object-Sensitive action patterns in business process model repositories. IEEE Trans. Software Eng., 37: 410-429. DOI: 10.1109/TSE.2010.96

Weijters, A.J.M.M. and W.M.P. van der Aalst, 2003. Rediscovering workflow models from event-based data using little thumb. Integrated Computer-Aided Eng., 10: 151-162. DOI: 10.3233/ICA-2003-10205

Weijters, A.J.M.M., W.M.P. van der Aalst, B. van Dongen, C. Günther and R. Mans *et al*., 2007. Process mining with ProM. Proceedings of the 19th Belgium-Netherlands Conference on Artificial Intelligence, (AIC' 07).

Weijters, A.J.M. and W.M.P. van der Aalst, 2005. Process Mining. In: Process-Aware Information Systems: Bridging People and Software Through Process Technology, Dumas, M., W.M.P. van der Aalst and A.H. Ter Hofstede (Eds.), John Wiley and Sons Inc.

Wen, L., J. Wang, W.M.P. van der Aalst, Z. Wang and J. Sun, 2004. A novel approach for process mining based on event types. BETA Working Paper Series (WP 118), Eindhoven University of Technology, Eindhoven.

Wen, L., J. Wang and J. Sun, 2006. Detecting implicit dependencies between tasks from event logs. Proceedings of the 8th Asia-Pacific Web Conference on Frontiers of WWW Research and Development, Jan. 16-18, Springer, Harbin, China, pp: 591-603. DOI: 10.1007/11610113_52

Wen, L., J. Wang, W.M.P. van der Aalst, Z. Wang and J. Sun, 2009. A novel approach for process mining based on event types. J. Intell. Inform. Syst., 32: 163-190. DOI: 10.1007/s10844-007-0052-1

Xumin, L., A. Moayad, D. Chen and Y. Qi, 2018. Log sequence clustering for workflow mining in multi-workflow systems. Data Knowl. Eng., 117: 1-17. DOI: 10.1016/j.datak.2018.04.002

Xiao, Y., J. Pallavi, X. Jianwu, J. Guoliang and Z. Hui *et al*., 2016. CloudSeer: Workflow monitoring of cloud infrastructures via interleaved logs. Proceedings of the 21th International Conference on Architectural Support for Programming Languages and Operating Systems, Apr. 02-06, ACM, Atlanta, Georgia, USA, pp: 489-502. DOI: 10.1145/2872362.2872407