*International Review on Computers and Software (I.RE.CO.S.), Vol. 11, N. 3*
*ISSN 1828-6003* *March 2016*

# Modified Time-Based Heuristics Miner for Parallel Business Processes

Riyanarto Sarno, Yutika Amelia Effendi, Fitrianing Haryadita

**Abstract** – *Process Mining, or Process Discovery, is a method for modeling the workflow of a business process from event logs. Business process models contain sequential and parallel traces. In this paper, a modification of the frequently used process-mining algorithm Heuristics Miner is proposed. The proposed algorithm is called Modified Time-based Heuristics Miner because it considers not only the sequence of activities but also the time-based information from the event log. It can effectively distinguish parallel (AND), single choice (XOR) and conditional (OR) patterns; the latter cannot be discovered by the original Heuristics Miner. The threshold intervals are determined on the basis of the average dependency measure in the dependency graph. The experimental results show that the proposed algorithm is able to discover concurrent business processes formed by parallel (AND) and conditional (OR) patterns, whereas the existing Heuristics Miner algorithm can only discover concurrent business processes formed by parallel (AND) patterns. This paper also provides an evaluation of validity and fitness of the discovered process model.* ***Copyright © 2016 Praise Worthy Prize S.r.l. - All rights reserved.***

***Keywords**: Process Mining, Process Discovery, Heuristics Miner, Time-based Interval, Parallel Business Process, Activity Lifespan, Fitness, Completeness, Double Time-stamped Event Log*

## Nomenclature

| | |
|---|---|
| $X =>_w Y$ | Value of dependency from X to Y |
| $\|X >_w Y\|$ | Frequency activity X is direct followed by activity Y |
| $\|Y >_w X\|$ | Frequency activity Y is direct followed by activity X |
| $X =>_w X$ | Value of length-one loop dependency |
| $\|X >_w X\|$ | Frequency of activity X directly followed by activity X |
| $max\,\{\|X >_w A\|\,\|\, A \in e\}$ | Frequency of activity X directly followed by activity A, where A is one of the activities |
| $X =>_{2w} Y$ | Value of length-two loop dependency |
| $\|X \gg_w Y\|$ | Frequency of XYX in event log |
| $\|Y \gg_w X\|$ | Frequency of YXY in event log |
| $ed$ | Execution duration of activity |
| $et_{activity_{output}}$ | Execution time of output activity |
| $et_{activity}$ | Execution time of activity |
| $X =>_w Y \wedge Z$ | Parallel measure between Y and Z, split in X |
| $\|Y \;\; _w Z\|$ | Frequency of activity Y directly followed by activity Z |
| $\|Z \;\; _w Y\|$ | Frequency of activity Z directly followed by activity Y |
| $\|X _w Y\|$ | Frequency of activity X directly followed by activity Y |
| $\|X _w Z\|$ | Frequency of activity X directly followed by activity Z |
| $\|Y\|\|wZ\|$ | Number of times (per case activity) that activity Y and Z are in a parallel relation in an event log |
| $\|Y >>> not_w Z\|$ | Frequency of activity Y not followed by activity Z |
| $\|Z >>> not_w Y\|$ | Frequency of activity Z not followed by activity Y |
| $Avg\ PDM$ | Average from arc weight of dependency graph |
| $e_i$ | Arc weight of dependence graph |
| $n_e$ | Number of edges |
| $Avg\ PM$ | Average of parallel measure which is calculated from Eq. (12) and has the same predecessor or parent |
| $PM$ | Parallel measure from Eq. (12) |
| $n_{PM}$ | Number of parallel measures |
| $PDM$ | Positive dependency measure from dependency graph |
| $Minimum\ PDM$ | Minimum value of positive dependency measure in matrix dependency measure |
| OR | OR relation. |
| XOR | XOR relation. |
| PM | Parsing measure |
| *c* | The number of correctly parsed traces in the event log |
| *t* | The total number of traces in the event log |
| CPM | Continuous parsing fitness measure |
| *e* | The total number of activities in the event log |
| *m* | The number of missing input in the event log |
| *r* | The number of remaining output in the event log |

| | |
|---|---|
| RBT | Relative-to-best threshold |
| POT | Positive observations threshold |
| DT | Dependency threshold |

## I. Introduction

A business process is a set of activities created to produce a specific output for specific purposes [1].

The analysis of the variety of activities and the creation of a model of a business process from event logs is the purpose of the process discovery [2].

Process discovery is one of the most challenging tasks in process mining and the large unsupervised learning task in nature, because event log rarely contains negative events [3]. A process discovery consists of a set of techniques that automatically construct a model of current activities of an organization on the basis of event logs. The business process model is then analyzed to reveal problems in the activities and to find some ways for solving them.

Recently Process Mining has been implemented in many fields, such as business [4], environment [5], [6], smartphone [7], and fraud [8], [9]. Each method has its own disadvantages resulting in the activities of the organization not being represented effectively in the discovered model [10].

The structure of a business process model can be expressed as a graph that contains a set of nodes connected by the edge [11]. Many algorithms are used for the process discovery, e.g. Alpha, Alpha+, Alpha++, and Heuristics Miner [12]. These algorithms are able to discover parallel (AND) or single choice (XOR) patterns [13] but none of them can discover conditional (OR) patterns. Hence, this paper introduces a modification of the existing Heuristics Miner (HM) that can discover conditional (OR) patterns in process models by considering an activity and its duration, i.e. time-based intervals. The proposed Modified Time-based Heuristics Miner (MTHM) basically extends the existing methods Heuristics Miner for Time Intervals [14] and Process Model Discovery based on Activity Lifespan [15].

The proposed algorithm takes benefit from recognizing start time and finish time of the event logs, especially when their interval times fully or partially overlap while there is actually no causal dependency among the activities. A previous research [15] introduced non-linear dependency with the same concept of time-based intervals by considering activities and their time intervals. The difference between HM and MTHM is in the mining of parallel activities and the calculation of frequency relations among activities. HM classifies the sequence and parallel relation of activities from every trace in the event log and it counts the frequency of the discovered relations in the event log, whereas MTHM adds to this the calculation of the frequency of activities in parallel choice not followed by another activity in parallel choice. The parallel measure is computed based on the causal matrix. Hence, the threshold of the average parallel measure is used to decide the types of parallel activities. In Section IV, the proposed method has been evaluated demonstrating that it can discover correct models containing parallel (AND) and conditional (OR) patterns whereas the existing HM cannot discover conditional (OR) patterns. After discovering the correct business process model, it is important to evaluate its validity and to optimize the model fitness. Aprocess model is called valid if there exists at least one trace from the beginning to the end [12]. Meanwhile, a model with a good fitness fits well with the reality; all the traces in the event logs are represented in the discovered process model. A process model is overfitted when the fitness is low, because only high frequency traces are selected, being the threshold too high. A process model is an underfitted process when the fitness is high because all frequency traces are selected, being the threshold too low. The notion of precision is related to the overfitted process models, while the notion of generalization is related to the underfitted process models [16].

## II. Research Method

In this section, the Heuristics Miner algorithm is modified so to be able to discover conditional (OR) patterns based on time interval. The modification involves activities in event logs expressed as time-based intervals to deal with incompleteness and distinguishing parallel (AND) from conditional (OR) patterns. Time-based interval utilizes a double time-stamped event log to discover sequential and concurrent activities:

- *Parallel (AND)* is used when a parallel split pattern occurs. A parallel split is defined as the mechanism that allows to split or to join so that it will be performed concurrently rather than serially. Two or more paths are created from the split of a single path, so at the same time two or more activities can start or a split activity will be executed. All the process discovery algorithms can model AND [12].
- *Conditional (OR)* is used when a multiple choice pattern occurs. In a multiple choice pattern, alternative paths may be chosen, from one to all. Technically, one path is allowed to be chosen, but when the process flow stops unexpectedly, this could be considered an invalid situation [16]. When mining an event log, some process discovery algorithms are not able to represent a multiple-choice workflow correctly. Some researchers have identified the pattern as XOR-split, while others have identified it as AND-split [17]. Alpha ++, which uses Petri net for modeling [12], is not able to discover OR.
- *Single Choice (XOR)* is used when only one activity is executed [18]. All the process discovery algorithms can model single choice (XOR) patterns.

### *II.1. Activities in Event Log Expressed as Time-based Intervals*

The Heuristics Miner recognizes not only the start and finish time as log events but also each activity as an instantaneous event. The original Heuristics Miner algorithm has been modified in this research so it can deal with activities expressed as time-based intervals. [14] introduced a new definition of the direct succession relation based on time-based intervals.

A single event is represented as an activity. Let's have the definition $X > w$ Y iff $\exists\sigma = t_1,\ldots, t_n$ and a $\in \{1,\ldots, n{-}1\}$ such that $\sigma \in E$, $t_i = X$ and $t_{i+1} = Y$.

This definition should be modified in order to be able to deal with activities expressed as time-based intervals.

There is an event log E. First, Name of Activity [E], the name of the activity that the event log belongs to, will be determined. Next Class Type of [E], the class type (start or end) of the event log is determined.

The new direct relation between two activities $X\ w\ Y$ is defined in Definition 1:

*Definition 1 (Direct relation)*

Given log event E and let activities X and Y be two time-based interval activities (not instantaneous), then:

X $w$Y iff $\exists\sigma = t_1,\ldots, t_n$ and a$\in \{2,\ldots, n-2\}$, $b \in \{3,\ldots, n-1\}$ such that $\sigma \in E$, $t_i = X_{end}$ and $t_j = Y_{start}$ and $\forall k$ such that $a < c < b$, we have defined that ClassTypeOf[$t_k$] $\neq$ start.

Based on this definition, it can be stated that two conditions must be met for two activities to be in a direct relation: (1) the start of the second activity occurs after the end of the first activity, and (2) no other activity should start between both activities. The paper introduces a new idea about the parallelism relation between two activities. There are two instantaneous activities X and Y; if activities X and Y are noticed in no specific order, e.g. activity X is occasionally noticed before activity Y and once in a while activity Y is noticed before activity X, so (X $>_W$ Y) $\wedge$ (Y $>_W$ X), then they are considered parallel. It may appear that this definition seems odd, but it is complicated to define a parallelism relation without the notion of duration. This can be more easily understood with a new definition of parallel relation:

*Definition 2 (Parallelism relation, k)*

Given log event E and let X and Y be two time-based interval activities (not instantaneous), then:

$X \|_W Y$ iff$\boldsymbol{\sigma} \in = t_1,\ldots,t_n$ and
$a, b, e, f \in \{1, \ldots, \mathrm{n}\}$
with $t_i = X_{start}$, $t_j = X_{end}$ and $t_u = Y_{start}$, $t_v = Y_{end}$
such that $e < a < f \vee a < e < b$

As stated before, this can be more easily understood when two activities overlap or when one contains the other. This definition shows two activities as having a parallel relation. [14] modified the notion of a parallel relation between two activities and the formulas of the algorithm for computing the statistical dependency to decide the parallel relation type: single choice (XOR) or parallel (AND). The new formulas are:

$$X =>_w Y = \left(\frac{|X\ \ _w Y| - |Y\ \ _w X|}{|X\ _w Y| + |Y\ _w X| + 2|X||wY| + 1}\right) \quad (1)$$

$$X =>_w (Y \wedge Z) = \left(\frac{|Y\ \ _w Z| + |Z\ _w Y| + 2|Y||wZ|}{|X\ _w Y| + |X\ _w Z| + 1}\right) \quad (2)$$

where $|Y||wZ|$ refers to the number of times (per case activity) that activities $Y$ and $Z$ are in a parallel relation in an event log $E$. Reference [14] introduced the notion of activities being in a succession relation if they overlap in the event log, so in the discovered model they are in a parallel relation. Using the two new formulas, [14] obtained the 'backward compatibility' with the original Heuristics Miner algorithm.

Using Eq. (1), if two activities X and Y do not overlap, they are not in a parallel relation, i.e. $|X||wY| = 0$. Both formulas can be used if activities are expressed as time-based intervals or as instantaneous events.

They can also improve their performance without losing the benefits of Heuristics Miner.

### II.2. Evaluating the Validity of the Discovered Process Model

The semantic discovered process model produced by Modified Time-based Heuristics Miner is considered not valid because it is not produced according to the formal theoretical rules, that describe the transformation from a dependency graph to a semantic discovered process model [12], [18].

A semantic discovered process model is a possible mined process model of an event log enriched with semantic split and join information.

However, it is important to distinguish two steps while creating the semantic discovered process model, i.e. (1) creating the dependency graph that becomes a causal net, (2) remodeling the causal net that becomes the semantic discovered process model.

*From dependency graph to causal net*

Causal nets are a representation toward process mining. A causal net is a graph where nodes represent activities and arcs represent the causal dependencies. Causal nets provide a better representational bias than conventional languages, which are too restrictive, such as Petri nets, BPMN, BPEL, EPCs, YAWL, and UML activity diagrams. In causal nets, each activity has a set of possible input bindings and a set of possible output bindings [16], [22].

When mining an event log, the Modified Time-based Heuristics Miner algorithm transforms the dependency graph into a causal net. Dependency graphs constructed by the Modified Time-based Heuristics Miner algorithm are built correctly. However, having a closer look, it is clear that the result does not correspond to a causal net, built according to the formal theoretical rules. Each activity in the causal net contains the correct input and output activities, but sometimes the construction of the subsets is not correct: two activities in the same subset are in an AND-relation and activities in different subsets are in an XOR-relation [19].

*From causal net to semantic discovered process model*

When the causal net result is converted to a semantic discovered process model, first, the activities are added to the semantic discovered process model. Second, the input and output gateways for each activity are added.

When adding a gateway, it is selected regardless of the relation between two activities. Finally, a selected gateway is added for each input and output activity.

Clearly this is incorrect, since an activity with several possible inputs (or outputs) should have only one gateway (XOR, AND or OR) as opposed to as many as its number of input (or output) activities. Furthermore, since the erratic causal net representation is used, the algorithm considers an XOR-relation as an AND-relation and vice versa.

Overall, in order to resolve these validity problems, first, when creating the causal net from the dependency graph, the parallel relations AND, OR and XOR must be interpreted correctly from the inputs and outputs in the dependency graph.

Next, the algorithm has to be adjusted to differentiate between a sequential relation and a parallel relation (an AND, OR, or XOR) between activities. The appropriate gateway is selected and added depending on the relation between the activities. Finally, the appropriate selected gateways are correctly connected to each other.

### II.3. Evaluating the Fitness of the Discovered Process Models

In the process mining literature, there is no common framework for evaluating discovered process models.

It is only stated that for comparing and evaluating a discovered process model one can use the results from different algorithms. Fitness, precision, generalization and structure are used to evaluate the quality of a model.

The fitness of discovered process models can be used to evaluate if the final decision of one method is better than others. A model that fits well with the reality is said to have good fitness. A process model has perfect fitness if all the traces in the event log from the beginning to the end are represented by the model.

Precision is related to the overfitted models, generalization is related to the underfitted models, while definition structure is the pragmatic quality of a model [12]. In Heuristics Miner, the calculation to quantify fitness uses two metrics: the parsing measure (PM) and the continuous parsing measure (CPM). The parsing measure is defined as follows:

$$PMw = \frac{c}{t} \tag{3}$$

where the symbols mean:

| | |
|---|---|
| $PM$ | : parsing measure; |
| $c$ | : the number of correctly parsed traces in the event log; |
| $t$ | : the total number of traces in the event log. |

[12] defines the formula to calculate the continuous parsing fitness measure, for example for identifying activities rather than traces, as follows:

$$CPMw = \frac{1}{2}\frac{(e-m)}{e} + \frac{1}{2}\frac{(e-r)}{e} \tag{4}$$

where the symbols mean:

| | |
|---|---|
| $CPM$ | : continuous parsing fitness measure; |
| $e$ | : the total number of activities in the event log; |
| $m$ | : the number of missing input in the event log; |
| $r$ | : the number of remaining output in the event log. |

## III. Proposed Method

As described in [18], there are three steps involved in discovering a model from an event log using Heuristics Miner. In this section, the formulas used in the three steps of Heuristics Miner will be modified.

### III.1. Mining the Dependency Graph

The construction of the dependency graph is the starting point of the Heuristics Miner algorithm. The true dependency between two events, X and Y (notation $X =>_W Y$) is indicated using a frequency based metric.

The frequency produces the direct successor frequency matrix. The heuristic search uses calculated $=>_W$ values between events in the event log. Eq. (5) is used to decide the dependency relation between the activities in the control-flow network:

$$X =>_w Y = \left( \frac{|X >_w Y| - |Y >_w X|}{|X >_w Y| + |Y >_w X| + \\ + 2|X||wY| + 1} \right) \tag{5}$$

where the symbols refer to:

| | |
|---|---|
| $X =>_w Y$ | : value of dependency from X to Y; |
| $\lvert X >_w Y \rvert$ | : frequency activity X is direct followed by activity Y; |
| $\lvert Y >_w X \rvert$ | : frequency activity Y is direct followed by activity X. |

Eq. (5) produces the dependency measure matrix used to make the dependency graph. Heuristics Miner as described in Weijters [18], [19] and Cnudde [12] uses a number of thresholds to determine the dependency measure:

a. *Relative-to-best threshold (RBT)*

This paper suggests that the relative-to-best threshold value is close to the average value of the dependency measure in a control-flow network:

- The first step to get the relative-to-best threshold value is to calculate the average value (Avg) of the positive dependency measure (PDM) in the dependency measure matrix.

- The second step is to determine the upper and lower limits of the threshold interval using standard deviation. This paper suggests the average value (Avg) of the positive dependency measure (PDM) as the upper limit and the average value of the dependency measure minus the standard deviation (SD) as the lower limit.
- The last step is to determine the relative-to-best threshold. This paper suggests the formula in Eq. (6) to determine the relative-to the best threshold:

$$RBT = Avg\ PDM - \left(\frac{SD\ PDM}{2}\right) \tag{6}$$

*b. Positive observations threshold (POT)*

The positive observations threshold controls the minimum frequency of dependency between activities.

The relation is used only when the frequency of dependency between activities is the same as or higher than the threshold value (i.e. there is dependency between two activities, *A* and *B*, so the positive observation threshold is $|X >_w Y| + |Y >_w X|$).

The positive observations threshold uses the minimum direct successor frequency from the direct successor frequency matrix [17], [20].

*c. Dependency threshold (DT)*

The dependency threshold indicates the edge that is going to be accepted (i.e. the edge that is inserted into the discovered process model). The edge will be accepted if the value of the dependency measure in the dependency measure matrix is higher than the value of the dependency threshold.

This paper suggests the formula in Eq. (7) to determine the dependency threshold:

$$RBT = Avg\ PDM - SD\ PDM \tag{7}$$

### *III.2. Checking Short Loop (Length-one Loop and Length-two Loop)*

One activity may be executed by the process multiple times. This condition is referred to as a loop. By using the dependency measure, Heuristics Miner has not any problem in detecting long-distance loops, but it has a problem for detecting short loops [12].

The formula for length-one loop is:

$$X =>_w X = \left(\frac{|X >_w X|}{\max\{|X >_w A| \mid A \in e\}}\right) \tag{8}$$

where the symbols refer to:

$X =>_w X$ : value of length-one loop dependency

$|X >_w X|$ : frequency of activity X directly followed by activity X

$max\,\{|X >_w A| \mid A \in e\}$ : frequency of activity X directly followed by activity A, where A is one of the activities in the event log.

The formula for length-two loop:

$$X =>_{2w} Y = \left(\frac{|X \gg_w Y| + |Y \gg_w X|}{|X \gg_w Y| + |Y \gg_w X| + 1}\right) \tag{9}$$

where the symbols refer to:

$X =>_{2w} Y$ : value of length-two loop dependency

$|X \gg_w Y|$ : frequency of XYX in event log

$|Y \gg_w X|$ : frequency of YXY in event log

### *III.3. Mining Parallel Activities*

The parallel measure calculation requires the frequency of parallel activities appearing in the event log. This frequency is obtained by counting the concurrent relation of all the parallel activities [21].

The concurrent relation discovery needs a double time-stamped event log. A common event log that was single time-stamped was acquired from an organization, but the event log used in this research was a double time-stamped event log that was extended from the single time-stamped event log. It was necessary to estimate the duration of each activity to convert the single time-stamped event log into a double time-stamped event log.

The determination of the duration of each activity in the event log was done by taking the average execution duration of each activity in the event log.

The execution duration was obtained by subtracting the output of an activity with the activity (i.e. if activity A has activity B as output then the execution duration of activity A is the execution time of B minus the execution time of A):

$$\begin{gathered}\text{if } activity_{output} = \text{true then} \\ ed = et_{activity_{output}} - et_{activity}\end{gathered} \tag{10}$$

where the symbols in the above equations refer to:

$ed$ : execution duration of activity;

$et_{activity_{output}}$ : execution time of output activity;

$et_{activity}$ : execution time of activity.

After the conversion of the event log, the next step is to calculate the frequency of the relations between activities. Based on [19], there are two steps to calculate the frequency, described as follows:

- To classify the sequential (>) and parallel (||) relation activities from every trace ($>_i$) in the event log based on Definition 3 and Definition 4.
- To count the frequency of the relations discovered in the event log.

*Definition 3.*

Given event log ($L$) and trace ($\sigma$) such that $\sigma \in L$. The causal relation between two activities $X(X_s, X_f)$ and $Y(Y_s, Y_f)$ according to which $X, Y \in L$ can be differentiated as follows:

$Before\ and\ meets, X > Y\ iff\ X_f \leq Y_s$

$Overlaps, X\square Y\ \ iff\ X_f > Y_s\ and\ X_f < Y_f$

$Contains, X@Y\ iff\ X_s < Y_s\ and\ Y_f > X_s\ and\ X_f > Y_f$

$Finished\ by, X_fY\ iff\ X_f = Y_f\ and\ X_s < Y_s\ and\ Y_s < X$

$Equals, X \lozenge Y \; iff \; X_s = Y_s \; and \; X_f = Y_f$
$Starts, X_p Y \; iff \; X_s = Y_s \; and \; X_f > Y_f$

*Definition 4.*

Given event log ($L$) and trace ($\sigma$) such that $\sigma \in L$. The control flow between two activities $X(X_s, X_f)$ and $Y(Y_s, Y_f)$ according to which $X, Y \in L$ can be differentiated as follows:

$$Sequence, X \rightarrow Y \; iff \; X > Y$$
$$Parallel, X \parallel Y \; iff \; X > Y \; and \; Y > X \; or$$
$$\{XY or \; X@Y \; or \; X_f Y \; or \; X \lozenge Y \; or \; X_p Y\}$$

Using the calculated frequency, the parallel measure is calculated using a modified version of the original formula for parallel measure from Heuristics Miner.

The original formula is:

$$X =>_w (Y \wedge Z) = \left( \frac{|Y \;_w Z| + |Z \;_w Y|}{|X \;_w Y| + |X \;_w Z| + 1} \right) \quad (11)$$

where the symbols refer to:

| | |
|---|---|
| $X =>_w Y \wedge Z$ | : parallel measure between Y and Z, split in X |
| $\|Y \;_w Z\|$ | : frequency of activity Y directly followed by activity Z |
| $\|Z \;_w Y\|$ | : frequency of activity Z directly followed by activity Y |
| $\|X \;_w Y\|$ | : frequency of activity X directly followed by activity Y |
| $\|X \;_w Z\|$ | : frequency of activity X directly followed by activity Z |
| $\|Y\|\|wZ\|$ | : number of times (per case activity) that activity Y and Z are in a parallel relation in event log E |

The existing method sets the frequency of direct activities in the event log. However, there are indirect activities that can increase the chance of discovering parallel (AND) patterns. Therefore, the modified method additionally sets the frequency of indirect activities in the event log. For example, the original Heuristics Miner uses Equation 11 to calculate the parallel measure of activities Y and Z which split into activity X.

This equation uses the direct activity frequency of parallel choice, i.e. activities Y and Z ($|Y >_w Z|$) and activities Z and Y ($|Z >_w Y|$), whereas parallel (AND) activities can be executed as far as all the split and join activities are executed in all the traces in the event log.

For counting the parallel measure of activities Y and Z which split into activity X, Modified Time-based Heuristics Miner therefore uses the frequency of direct or parallel choice which splits into its parent activity, i.e. activities Y and Z ($|Y >>>_w Z|$) and activities Z and Y ($|Z >>>_w Y|$), and for the XOR and OR threshold it uses the frequency of parallel choice activities that are not followed by another parallel choice activity.

The formula for parallel measure in Modified Time-based Heuristics Miner is written in Eq. (12).

From the frequency calculated using the time-based intervals, if a parallel activity does not appear in the causal matrix then it is necessary to look for its predecessor or successor.

Next, the parallel choice activities should be summed from the predecessor or successor into the frequency of the predecessor or successor as it appears in the causal matrix. The formula for parallel measure is*:*

$$X =>_w (Y \wedge Z) = \left( \frac{|Y \;_w Z| + |Z \;_w Y| + 2|Y||wZ|}{\begin{array}{c}|X \;_w Y| + |X \;_w Z| + |Y \gg> not_w Z| + \\ + |Z \gg> not_w Y| + 1\end{array}} \right) \quad (12)$$

where the symbols refer to:

$X =>_w Y \wedge Z$ : parallel measurebetween Y and Z, split in X

$|Y \;_w Z|$ : frequency of activity Y directly followed by activity Z

$|Z \;_w Y|$ : frequency of activity Z directly followed by activity Y

$|X \;_w Y|$ : frequency of activity X directly followed by activity Y

$|X \;_w Z|$ : frequency of activity X directly followed by activity Z

$|Y||wZ|$ : number of times (per case activity) that activity Y and Z are in a parallel relation in an event log

$|Y >>> not_w Z|$ : frequency of activity Y not followed by activity Z

$|Z >>> not_w Y|$ : frequency of activity Z not followed by activity Y

After discovering all the relations in the event log, the parallel relations are classified into parallel (AND), conditional (OR), and single choice (XOR).

The classification utilizes the following equations:

$$Avg\ PDM = \frac{\sum_{i=1}^{n_e} e_i}{n_e} \quad (13)$$

$$Avg\ PM = \frac{\sum_{i=1}^{n_{PM}} PM_i}{n_{PM}} \quad (14)$$

The intervals for discovering XOR, OR, and AND are:

➢ XOR:

$$If\ Avg\ PM \leq Minimum\ PDM\ then\ XOR \quad (15)$$

➢ OR:

$$If\ Minimum\ PDM \leq Avg\ PM \leq Avg\ PDM\ then\ OR \quad (16)$$

➢ AND:

$$If\ Avg\ PDM \leq Avg\ PM\ then\ AND \quad (17)$$

where the symbols in the above equations are defined as:

$Avg\ PDM$ : average from arc weight of dependency graph;

$e_i$ : arc weight of dependence graph;
$n_e$ : number of edges;
$Avg\,PM$ : average of parallel measure which is calculated from Equation 12 and has the same predecessor or parent;
$PM$ : parallel measure from Eq. (12);
$n_{PM}$ : number of parallel measures;
$PDM$ : positive dependency measure from dependency graph;
$Minimum\ PDM$: minimum value of positive dependency measure in matrix dependency measure.

## IV. Results and Analysis

This research uses a double time-stamped event log containing the execution of activities of more than one department in an organization. The information attributes contained in the event log are: the number of case id, the activity in process, and the timestamp of activity execution (start time and finish time).

### *IV.1. Experiment Data*

Table I contains the event log used for the experiment, which includes the case id, the activity in process, and the timestamp of activity execution as follows:

L = {(ABCDEFGHIJK), (ABCDEFGIJK), (ABDCEFHIJK), (ABDCEFHGIJK), (ABCEFGHIJK), (ABCDEFGIJK), (ABCDEFGHIJK), (ABCDEFGIJ), (ABDCEFHIJJK), (ABDCEFHGIJKJK),(ABCDEFGIJJK), (ABCDEFGIJKJK), (ABCDEFGHIJJK), (ABDCEFHIJKJK), (ABDCEFHGIJJK), (ABDCEFHGIJKJK), (ABCDEFGHIJKJK), (ABDCEFHIJKJK)}.

### *IV.2. Mining Using Modified Time-based Heuristics Miner*

In this section, an example of process discovery using Modified Timed-based Heuristics Miner will be illustrated.

#### *IV.2.1. Mining the Dependency Graph*

The first step is to create the activity dependence relation frequency matrix from the event log.

This research uses the dependency mined by Heuristics Miner to determine the relations of each activity. The event log in Table I produces the matrix shown in Table II. The second step is to count the dependency measure using Eq. (5). The formula produces the dependency measure matrix. The third step is to determine the thresholds. This step needs the average and standard deviation of the positive dependency measure calculated from the dependency measure matrix:

- Average of positive dependency measure = 0.884;
- Standard deviation of positive dependency measure = 0.102.

The value of the relative-to-best threshold is calculated using Eq. (6).

TABLE I
EVENT LOG

| Case ID | Activity | Start Stamp | End Stamp |
|---|---|---|---|
| PP1 | A | 6/20/2014 8:32 | 6/20/2014 13:42 |
| | B | 6/20/2014 13:42 | 6/20/2014 23:41 |
| | C | 6/20/2014 23:41 | 6/21/2014 9:30 |
| | D | 6/21/2014 8:16 | 6/21/2014 10:46 |
| | E | 6/21/2014 9:46 | 6/21/2014 16:57 |
| | F | 6/21/2014 16:57 | 6/21/2014 18:09 |
| | G | 6/21/2014 18:09 | 6/22/2014 4:14 |
| | H | 6/21/2014 19:15 | 6/22/2014 10:51 |
| | I | 6/22/2014 10:51 | 6/22/2014 16:28 |
| | J | 6/22/2014 16:28 | 6/22/2014 23:42 |
| | K | 6/22/2014 23:42 | 6/23/2014 4:48 |
| PP2 | A | 6/23/2014 4:48 | 6/23/2014 16:44 |
| | B | 6/23/2014 16:44 | 6/23/2014 23:26 |
| | C | 6/23/2014 23:26 | 6/24/2014 5:40 |
| | D | 6/24/2014 4:19 | 6/24/2014 7:48 |
| | E | 6/24/2014 7:48 | 6/25/2014 1:08 |
| | F | 6/25/2014 1:08 | 6/25/2014 3:02 |
| | G | 6/25/2014 3:02 | 6/25/2014 5:11 |
| | I | 6/25/2014 5:11 | 6/25/2014 8:25 |
| | J | 6/25/2014 8:25 | 6/25/2014 12:45 |
| | K | 6/25/2014 12:45 | 6/26/2014 0:12 |
| … | ... | …. | … |
| PP18 | …. | …. | …. |

TABLE II
THE DIRECT SUCCESSOR FREQUENCY MATRIX

| No | Frequency of activity 1 directly followed by activity 2 | Frequency number |
|---|---|---|
| 1 | $\|A >_w B\|$ | 18 |
| 2 | $\|B >_w C\|$ | 10 |
| 3 | $\|B >_w D\|$ | 8 |
| 4 | $\|C >_w D\|$ | 9 |
| 5 | $\|C >_w E\|$ | 9 |
| 6 | $\|D >_w C\|$ | 8 |
| 7 | $\|D >_w E\|$ | 9 |
| 8 | $\|E >_w F\|$ | 18 |
| 9 | $\|F >_w G\|$ | 10 |
| 10 | $\|F >_w H\|$ | 8 |
| 11 | $\|G >_w H\|$ | 5 |
| 12 | $\|G >_w I\|$ | 9 |
| 13 | $\|H >_w G\|$ | 4 |
| 14 | $\|H >_w I\|$ | 9 |
| 15 | $\|I >_w J\|$ | 18 |
| 16 | $\|J >_w K\|$ | 23 |
| 17 | $\|K >_w J\|$ | 6 |

TABLE III
DEPENDENCY MEASURE MATRIX

| No | Value of dependency measure from activity 1 to activity 2 | Value of dependency measure |
|---|---|---|
| 1 | $A =>_w B$ | 0.947 |
| 2 | $B =>_w C$ | 0.9 |
| 3 | $B =>_w D$ | 0.889 |
| 4 | $C =>_w D$ | 0 |
| 5 | $C =>_w E$ | 0.9 |
| 6 | $D =>_w C$ | 0 |
| 7 | $D =>_w E$ | 0.889 |
| 8 | $E =>_w F$ | 0.947 |
| 9 | $F =>_w G$ | 0.9 |
| 10 | $F =>_w H$ | 0.889 |
| 11 | $G =>_w H$ | 0 |
| 12 | $G =>_w I$ | 0.9 |
| 13 | $H =>_w G$ | 0 |
| 14 | $H =>_w I$ | 0.9 |
| 15 | $I =>_w J$ | 0.947 |
| 16 | $J =>_w K$ | 0.923 |
| 17 | $K =>_w J$ | -0.923 |

*Relative-to-best threshold = 0.83*

The positive-to-best threshold uses the minimum dependency from the direct successor frequency matrix.

*Positive observations threshold = 4*

The value of the dependency threshold is calculated using Eq. (7).

*Dependency threshold = 0.84*

This threshold indicates the dependency relation between the activities that will be included. The relations between activities that have a dependency measure equal to or higher than the dependency threshold will be included. Based on this threshold, the dependency graph can be modeled as follows (Fig. 1).

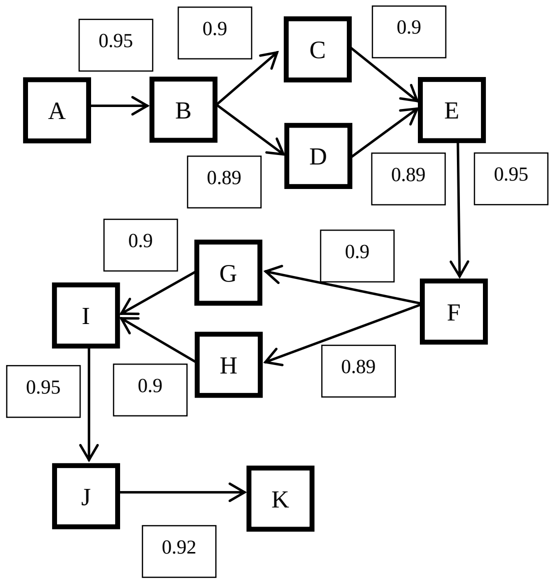

Fig. 1. Dependency Graph

### IV.2.2. Checking Short Loop (Length-one Loop and Length-two Loop)

Length-one loop needs the frequency matrix for length-one loop to calculate the dependency of length-one loop.

TABLE IV
LENGTH-ONE LOOP FREQUENCY MATRIX

| No | Frequency of activity 1 directly followed by activity 2 | Frequency number |
|---|---|---|
| 1 | $\|J >_w J\|$ | 4 |

Then using the length-one loop formula from Eq. (8), the matrix shown in Table V is obtained.

TABLE V
LENGTH-ONE LOOP DEPENDENCY MATRIX

| No | Value of length-one loop dependency | Value of L1L |
|---|---|---|
| 1 | $J =>_w J$ | 0.8 |

Length-two loop needs the length-two loop frequency matrix to calculate the length-two loop dependency.

TABLE VI
LENGTH-TWO LOOP FREQUENCY MATRIX

| No | Frequency of aba in event log | Frequency number |
|---|---|---|
| 1 | $\|J >_w K\|$ | 6 |
| 2 | $\|K >_w J\|$ | 6 |

Then, using the length-two loop formula from Eq. (9), the matrix shown in Table VII is obtained

TABLE VII
LENGTH-TWO LOOP DEPENDENCY MATRIX

| No | Value of length two-loop dependency | Value of L2L |
|---|---|---|
| 1 | $J =>_w K$ | 0.923 |
| 2 | $K =>_w J$ | 0.923 |

From Table V and Table VII, it can be seen that length-one loop JJ and length-two loop JKJ are contained in the model.

Hence the model can be described as in Fig. 2. The final graph from the proposed method is as follows.

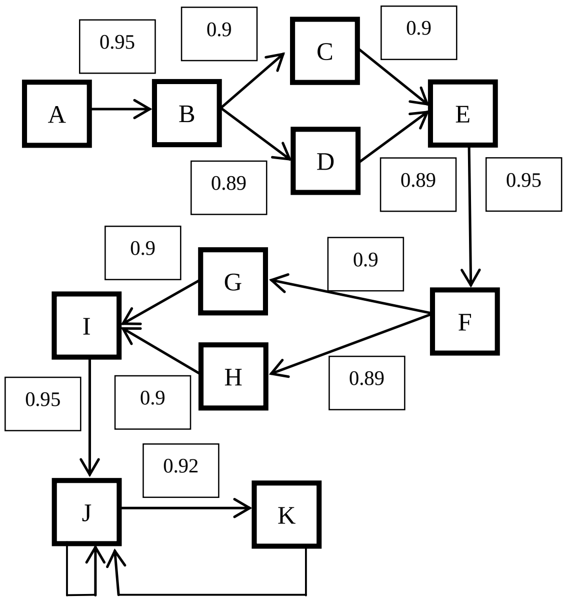

Fig. 2. Dependency Graph with Length-one Loop and Length-two Loop

### IV.2.3. Mining Parallel Activity

The first thing to do is create the causal matrix.

TABLE VIII
CAUSAL MATRIX

| INPUT | ACTIVITY | OUTPUT |
|---|---|---|
| {} | A | B |
| A | B | C,D |
| B | C | E |
| B | D | E |
| C,D | E | F |
| E | F | G,H |
| F | G | I |
| F | H | I |
| G.H | I | J |
| I | J | K |
| J | K | {} |

Since there are inputs and outputs with more than one activity, the parallel activities are $B \to w\, C \wedge D$, $E \to$

$w\ C \wedge D$, $F\ \rightarrow w\ G \wedge H$ and $I\ \rightarrow w\ G \wedge H$. The parallel measure is calculated using Eq. (10).
$B\ \rightarrow w\ C \wedge D$ = 1.378, $E\ \rightarrow w\ C \wedge D$ = 1.378, $F\ \rightarrow w\ G \wedge H$ = 0.729 and $I\ \rightarrow w\ G \wedge H$ = 0.729.

The next step is to check if both parallel measurements can be averaged or not. Because the inputs and outputs have one same activity and the number of input and output activities are the same, the parallel measurement can be averaged using Eqs. (11) and (12).

Next, XOR, OR, and AND are determined using Eq. (15), (16), (17). Eq. (15), (16), and (17) need the average dependency measure and the threshold interval of the dependency measure from the dependency graph.

The average dependency measure using Equation 13 is 0.884. The minimum dependency measure using Eq. (14) is 0.567. The average parallel measure of $B\ \rightarrow w\ C \wedge D$ and $E\ \rightarrow w\ C \wedge D$ is 1.378 and the average parallel measure of $F\ \rightarrow w\ G \wedge H$ and $I\ \rightarrow w\ G \wedge H$is 0.729. Eqs. (15), (16), and (17) show that the split and join patterns used in this model are AND and OR.

Hence, the model becomes as shown in Fig. 3.

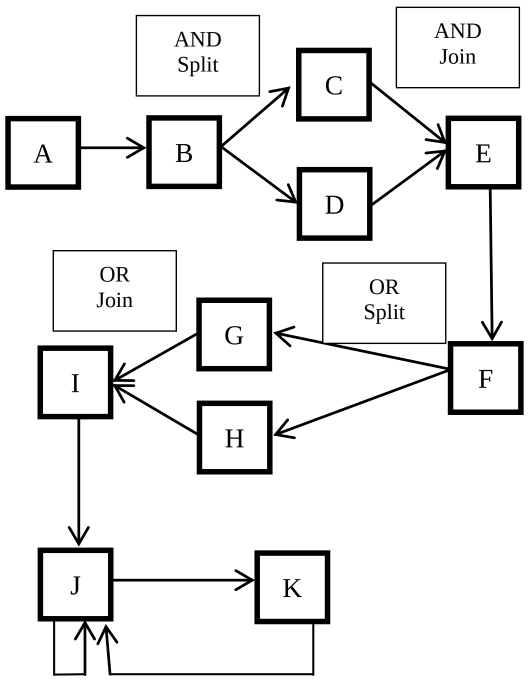

Fig. 3. Final Model from Modified Time-based Heuristics Miner

The above experiments use 18 cases from the event log and discover the process model using the Modified Time-based Heuristics Miner algorithm.

Additionally, we used 50 cases from the event log and the results were the same as when 18 cases from the event log were used. The Modified Time-based Heuristics Miner algorithm can find input, output, parallel relation (OR, AND, XOR), and short loop.

### IV.3. Mining Using the Original Heuristics Miner

Based on [14], the event log in Table I is discovered using the original Heuristics Miner with the direct successor frequency matrix as shown in Table II. The basic formula has been applied for mining the dependency measure from Heuristics Miner [13], [14].

Then the following data have been calculated: the relative-to-best threshold (RBT), positive observations threshold (POT), and dependency threshold (DT) using Eqs. (6) and (7).

The results of RBT, POT and DT were 0.622, 4, 0.472. For mining short loop, length-one loop and length-two loop were used in the graph model because they don’t use the rule of completeness. Thus, the process model contained the same result of length-one loop and length two-loop as in Table V and Table VII in Section IV.2.

The same causal matrix as shown in Table VIII was used, but the calculation of the parallel measure was different. The original Heuristics Miner used Equation 11 to calculate the parallel measure, getting $B\ \rightarrow w\ C \wedge D$ = 0.895, $E\ \rightarrow w\ C \wedge D$ = 0.895, $F\ \rightarrow w\ G \wedge H$ = 0.474, and $I\ \rightarrow w\ G \wedge H$ = 0.474.

Using the rule of the original Heuristics Miner, if the parallel measure is more than 0.1, then the split and join used is AND. Thus, the model discovered using the original Heuristics Miner appears as in Fig. 4.

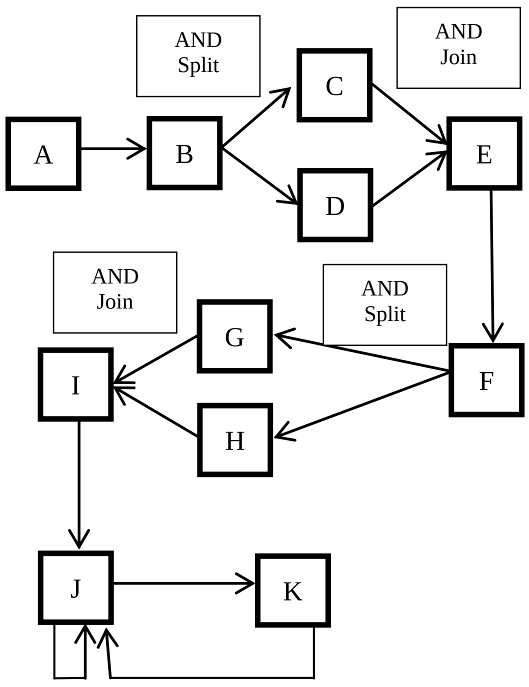

Fig. 4. Final Model From Original Heuristics Miner

### IV.4. Mining Using Process Model Discovery Based on Activity Lifespan

The algorithm introduced by Rizka et al. [11] was used to discover a model from the event log in Table I.

The relations discovered in every trace are united in each classification:

- *Step 1.* Input (T1) and output (T0) from each trace:

$$I = A,\ O = K$$

- *Step 2.* All sequence relations:

$$> =\{\{\} \to A, A \to B, B \to C, B \to D, C \to E, D \to E, E \to F, F \to G, F \to H, G \to I, H \to I, I \to J, J \to J, J \to K, K \to J, J \to K, K \to \{\}\}$$

- *Step 3.* All parallel relations

$$\| = \{C\|D, G\|H\}$$

- *Step 4.* Classify parallel relations into parallel (AND) and conditional (XOR)

$$\oplus \text{ (Parallel AND)} = \{C \oplus D, G \oplus H\}$$
$$\otimes \text{ (Conditional XOR)} = \{\}$$

- *Step 5. parallel AND relation*

$$\oplus = \{B \oplus (C, D)\}$$
$$\oplus = \{F \oplus (G, H)\}$$

- *Step 6.* Graph composes XOR or AND relations

$$G = \{ B \oplus (C, D) \text{ and } F \oplus (G, H)\}$$

- *Step 7.* Adding sequence relations and input/output in the graph

$$G \leftarrow G \cup (>, I, O)$$

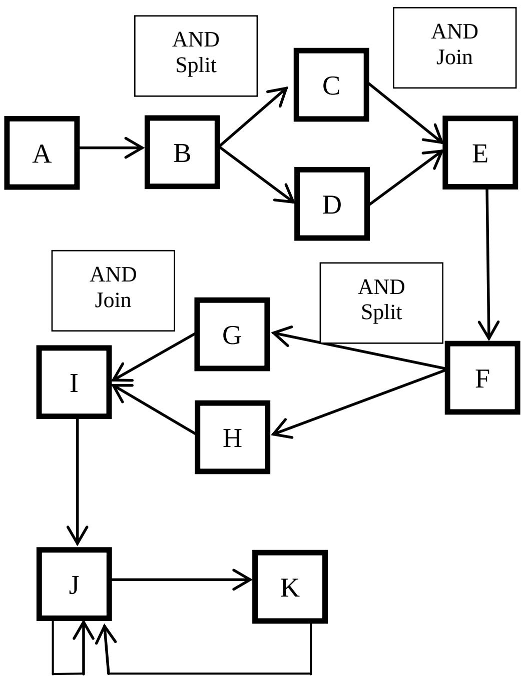

Fig. 5. Final Model from Process Model Discovery Based on Activity Lifespan

### IV.5. Mining Using Heuristics Miner for Time Intervals

The Heuristics Miner for Time Intervals algorithm introduced by Burattin [2] was used to discover a model from the event log in Table I. The following data were calculated: the relative-to-best threshold (RBT), positive observations threshold (POT), and dependency threshold (DT) using Eqs. (6) and (7). The results of RBT, POT and DT were 0.833, 4, and 0.782 respectively.

Using the same causal matrix as shown in Table VIII, the calculation of the parallel measure was different.

Heuristics Miner for Time Intervals used Eq. (2) to calculate the parallel measure, getting $B \to_w C \wedge D$ = 2.684, $E \to_w C \wedge D$ = 2.684, $F \to_w G \wedge H$ = 1.421 and $I \to_w G \wedge H$ = 1.421.

According to the rule of Heuristics Miner for Time Intervals, if the parallel measure is higher than 0.1, then the split and join used is AND. The obtained model discovered using Heuristics Miner for Time Interval is represented in Fig. 6.

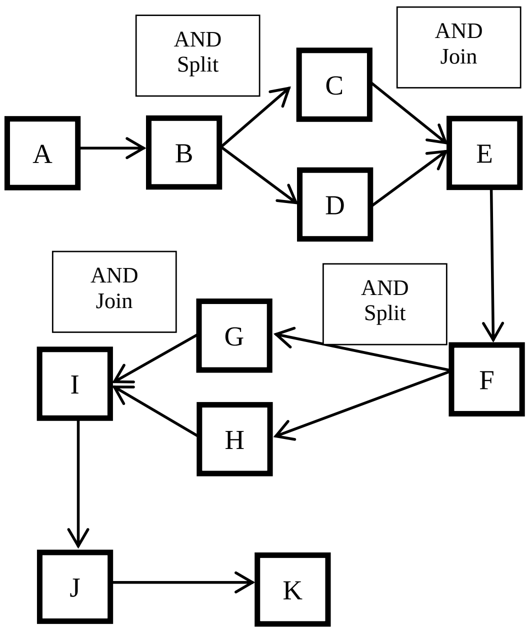

Fig. 6. Final Model from Heuristics Miner for Time Intervals

### IV.6. Validity of the Discovered Process Model

Using Modified Time-based Heuristics Miner, the semantic discovered process model could be generated from the discovered process model in Section IV.2 (Fig. 3), as shown in Table IX and Table X.

The Modified Time-based Heuristics Miner algorithm transformed the dependency graph shown in Table IX into the causal net in Table X. Each activity in the causal net in Table X contains its correct input and output activities.

The last part of the transformation step connects the gateways to each other in order to form a connected semantic discovered process model. The resulting semantic discovered process model of event log L is shown in Fig. 7. Following the rules of validity, it can be concluded that the results of Modified Time-based Heuristics Miner generated the semantics in a valid way.

The final graph from the proposed method is shown in Fig. 7.

TABLE IX
DEPENDENCY GRAPH OF EVENT LOG GENERATED BY THE PROPOSED METHOD

| INPUT SET | ACTIVITY | OUTPUT SET |
|---|---|---|
| {∅} | A | {B} |
| {A} | B | {C,D} |
| {B} | C | {E} |
| {B} | D | {E} |
| {C,D} | E | {F} |
| {E} | F | {G,H} |
| {F} | G | {I} |
| {F} | H | {I} |
| {G.H} | I | {J} |
| {I} | {J} | {K} |
| {J} | K | {∅} |

TABLE X
CAUSAL NET OF EVENT LOG GENERATED BY THE PROPOSED METHOD

| INPUT SET | ACTIVITY | OUTPUT SET |
|---|---|---|
| {∅} | A | {{B}} |
| {{A}} | B | {{C},{D}} |
| {{B}} | C | {{E}} |
| {{B}} | D | {{E}} |
| {{C},{D}} | E | {{F}} |
| {{E}} | F | {{G},{H}} |
| {{F}} | G | {{I}} |
| {{F}} | H | {{I}} |
| {{G}.{H}} | I | {{J}} |
| {{I}} | {J} | {{K}} |
| {{J}} | K | {∅} |

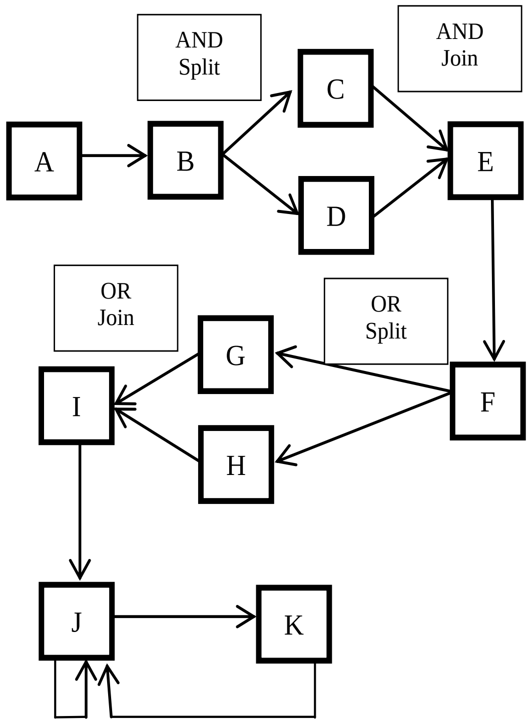

Fig. 7. Final Model from Modified Time-Based Heuristics Miner

### IV.7. *The Fitness of the Discovered Process Models*

Considering the fitness definition given in Section II.3, it is possible to evaluate and to compare the fitness of the process models discovered by the four different algorithms. Using the parsing measure (Eq. (3)) and the continuous parsing measure (Eq. (4)), the fitness values for each method were listed in Tables XI, XII.

TABLE XI
FITNESS VALUES USING THE PARSING MEASURE (PM)

| PM | C | T | Method/algorithm |
|---|---|---|---|
| 0.941 | 191 | 203 | Proposed Method |
| 0.818 | 166 | 203 | Original Heuristics Miner |
| 0.887 | 180 | 203 | Time-Based Discovery |
| 0.852 | 173 | 203 | Heuristics Miner for Time Intervals |

TABLE XII
FITNESS VALUES USING THE CONTINUOUS PARSING MEASURE (CPM)

| CPM | e | M | r | Method/algorithm |
|---|---|---|---|---|
| 0.955 | 198 | 10 | 8 | Proposed Method |
| 0.954 | 198 | 10 | 8 | Original Heuristics Miner |
| 0.952 | 198 | 11 | 8 | Time-Based Discovery |
| 0.909 | 198 | 20 | 16 | Heuristics Miner for Time Intervals |

The fitness values and their constituting variables for the event log in Table I mined by the four different algorithms are shown in Table IX and Table X.

The number of correctly parsed traces c has increased and the number of missing and remaining activities (*m* and *r*) has decreased, which means that the process models discovered by Modified Time-based Heuristics Miner fit the reality better than the process models discovered by the original Heuristics Miner, Process Model Discovery Based on Activity Lifespan, and Heuristics Miner for Time Intervals.

## V. Conclusion

This paper presented a modification of the Heuristics Miner algorithm using the activity expressed as time-based intervals and not as single time-stamped.

This new approach has been introduced into the original heuristics miner algorithm paying attention to backward compatibility. It has been shown that the proposed Modified Time-based Heuristics Miner algorithm can discover business process models containing conditional (OR) parallel relations, which cannot be discovered by the existing algorithms, such as original Heuristics Miner, Process Model Discovery based on Activity Lifespan, and Heuristics Miner for Time Intervals. The Modified Time-based Heuristics Miner method elaborates the mining of parallel activities from the causal matrix and computes the parallel measure. Hence, the threshold interval of the average parallel measure is used to decide the types of parallel split and join patterns. The experimental results showed that Modified Time-based Heuristics Miner can discover correct process models that contain single choice (XOR), parallel (AND) and conditional (OR) parallel relations. It does not only discover correct models containing parallel relations but it can also improve the validity and fitness of the discovered process models.

## Acknowledgements

Authors would like to thank the Institut Teknologi Sepuluh Nopember and the Ministry of Research, Technology and Higher Education of Indonesia for supporting this research.

## References

[1] R. Sarno, C. A. Djeni,I. Mukhlash, D. Sunaryono, Developing A Workflow Management System for Enterprise Resource Planning,(2015) *Journal of Theoretical and Applied Information Technology*,72 (3), pp. 412-421.

[2] R. Sarno, B. A. Sanjoyo,I. Mukhlash, H. M. Astuti, Petri Net Model of ERP Business Process Variation for Small and Medium Enterprises, (2013) *Journal of Theoretical and Applied Information Technology*,54, pp. 412-421.

[3] S. Goedertier, J. De Weerdt, D. Martens, J. Vanthienen, and B. Baesens, Process Discovery in Event Log: An Application in the Telecom Industry,(2011) *Applied Soft Computing*, 11 (2), pp. 1697- 1710.

[4] O. T. Baruwa, M. A. Piera, Identifying FMS repetitive patterns for efficient search-based scheduling algorithm: A colored Petri Net approach, (2015)*Journal of Manufacturing Systems*, 35, pp. 120-135.
http://dx.doi.org/10.1016/j.jmsy.2014.11.009

[5] A. Sanaa, S. B.Abid, A.Boulila, C.Messaoud, M.Boussaid, N. B. Fadhel, Modellinghydrochory effects on the Tunisian island populations of Pancratiummaritimum L. using colored Petri Nets,(2015)*BioSystems*, 129, pp. 19-24.
http://dx.doi.org/10.1016/j.biosystems.2015.02.001

[6] J. Yuan, D. Oswald, W. Li, Autonomous tracking of chemical plumes developed in both diffusive and turbulent airflow environment using Petri Nets,(2015) *Expert Systems with Application*, 42, pp. 527-538.
http://dx.doi.org/10.1016/j.eswa.2014.08.005

[7] V. R.L. Shen, H.-Y. Lai, A.-F. Lai, The implementation of a smartphone-based fall detection system using a high-level fuzzy Petri Net, (2015) *Applied Soft Computing*, 26, pp. 390-400.
http://dx.doi.org/10.1016/j.asoc.2014.10.028

[8] R. Sarno, H. Ginardi, E. W. Pamungkas and D. Sunaryono.Clustering of ERP Business Process Fragments.*Proceedings IEEE International conference on computer, control, informatics, and its applications*.(Page: 319-324 Year of Publication: 2013).
http://dx.doi.org/10.1109/IC3INA.2013.6819194

[9] Huda, S., Ahmad, T., Sarno, R., Santoso, H.A, Identification of process-based fraud patterns in credit application, *2nd International Conference on Information and Communication Technology (ICoICT)* (Year of Publication: 2014).
http://dx.doi.org/10.1109/ICoICT.2014.6914045

[10] R. Sarno, Pamungkas, E.W., Sunaryono, D., Sarwosri, Business process composition based on meta models, *International Seminar on Intelligent Technology and Its Applications (ISITIA)* (Year of Publication: 2015).
http://dx.doi.org/10.1109/ISITIA.2015.7219998

[11] R.Sarno, R. D. Dewandono, T. Ahmad, M. F. Naufal, F.Sinaga, Hybrid Association Rule Learning and Process mining for Fraud Detection, (2015)*IAENG International Journal of Computer Science* 42 (2), pp. 59-72.

[12] S. De Cnudde, J. Claes, G. Poels, Improving the Quality of the Heuristics Miner in Prom 6.2, (2014) *Expert System with Application*,41, pp. 7678-7690.

[13] W. Philip, Principled Approach to Mining From Noisy Log Using Heuristics Miner, (2013) *IEEE Symposium on Computational Intelligence and Data Mining.*

[14] A. Burattin, and A. Sperduti. *Heuristics Miner for Time Intervals*.ESANN 2010 proceedings, European Symposium on Artificial Neural Networks - Computational Intelligence and Machine Learning.(Page: 41-46 Year of Publication: 2010)

[15] R. A. Sutrisnowati, HyerimBae, L. Dongha and K. Minsoo.*Process Model Discovery based on Activity Lifespan*. International Conference on Technology Innovation and Industrial Management Seoul.(Page: 137-156 Year of Publication: 2014).
http://dx.doi.org/10.1016/j.eswa.2014.05.055

[16] W. M. P. van der Aalst, *Process mining: discovery, conformance and enhancement of business processes*(Springer Science and Business Media, 2011).
http://dx.doi.org/10.1007/978-3-642-19345-3

[17] Sarno, R., Sari, P.L.I., Sunaryono, D., Amaliah, B., Mukhlash, I., Mining decision to discover the relation of rules among decision points in a non-free choice construct, *Proceedings of International Conference on Information, Communication Technology and System (ICTS)* (Year of Publication: 2014).
http://dx.doi.org/10.1109/ICTS.2014.7010557

[18] A. J. M. M. Weijters, W. M. P. van der Aalst, A. K.Alves de Medeiros, Process mining with the Heuristics-miner algorithm, (2006) *Technische Universiteit Eindhoven, Tech. Rep. WP*, 166, pp. 1-34.

[19] Weijters, A.J.M.M.,Ribeiro, J.T.S., Flexible Heuristics Miner (FHM), *Proceedings of IEEE Symposium on Computational Intelligence and Data Mining* (Year of Publication: 2011).
http://dx.doi.org/10.1109/cidm.2011.5949453

[20] R. Sarno, P. L. I. Sari, HariGinardi, DwiSunaryono and Imam Mukhlash. *Decision Mining for Multi Choice Workflow Patterns*. International Conference on Computer, Control, Informatics ad Its Applications. (Page: 337-342 Year of Publication: 2013).
http://dx.doi.org/10.1109/IC3INA.2013.6819197

[21] R. Sarno, W. A. Wibowo, Kartini, F. Haryadita, Determining Model Using Non-Linear Heuristics Miner and Control-Flow Pattern, (2016) *TELKOMNIKA (Telecommunication Computing Electronics and Control)*, 14 (1).
http://dx.doi.org/10.12928/telkomnika.v14i1.3257

[22] W.M.P. van der Aalst, A. Adriansyah, B. Van Dongen,Causal Netss: A Modeling Language Tailored towards Process Discovery, In J.P.K.B. Konig, *CONCUR 2011-Concurrency Theory* (Berlin: Springer Berlin Heidelberg, 2011, 28-42).
http://dx.doi.org/10.1007/978-3-642-23217-6_3

## Authors' information

Informatics Department, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia.

**Riyanarto Sarno** received M.Sc and Ph.D in Computer Science from the University of Brunswick Canada in 1988 and 1992. His research includes Internet of Things, Enterprise Computing, Information Management, Intelligent Systems and Smart Grids.
E-mail: riyanarto@if.its.ac.id

**Yutika Amelia Effendi** was born in Solok, Indonesia. She is a fresh graduate from Informatics Department, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia.Her major skills and researches are in Process Mining, ERP, IT Governance and Knowledge Engineering.
E-mail: yutika.effendi@gmail.com

**Fitrianing Haryadita** was born in Klaten, Indonesia. She was an alumnus of Informatics Department, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia.Her major skills and researches are in Process Mining and Web Development.
E-mail: fitrianing.haryadita@gmail.com