Received: January 7, 2019

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Heuristic Linear Temporal Logic Pattern Algorithm in Business Process Model

Kelly Rossa Sungkono[1] **Ulva Erida Nur Rochmah[1]** **Riyanarto Sarno[1]***

[1]*Department of Informatics, Institut Teknologi Sepuluh Nopember, Indonesia*
* Corresponding author's Email: riyanarto@if.its.ac.id

Abstract: Process discovery obtains a process model of activity records. There are two representations of process model, i.e. a probabilistic model and a deterministic model. A deterministic model takes all of activity records to depict a process model, however, the probabilistic model chooses several activity records that satisfy a threshold. Determination of the right threshold leads the emergence of many discovery algorithms of probabilistic models, such as Heuristic Miner, Fodina, Modified Heuristic Miner, and Modified Time-Based Heuristic Miners. Those algorithms determine a threshold based on users or an average of probabilities of activity records, so the quality of the model depends on user proficiency or frequent activities. This paper proposes a new algorithm of probabilistic model discovery, i.e. Heuristic Linear Temporal Logic (HLTL), which determines the threshold based on four quality aspects, i.e. Fitness, Precision, Generalization, and Simplicity. HLTL utilizes Linear Temporal Logic to create a formal representation of process model and store the weight of relationships used for the threshold formation. The result shows that the process model constructing by HLTL has better quality aspects than the process model constructing by Modified Heuristic Miners and Modified Time-Based Heuristic Miners. The generalization value of HLTL is 0.8422 and the generalization value of Modified Heuristic Miner and Modified Time-Based Heuristic Miners are 0.8421.

Keywords: Linear temporal logic, Heuristic miner, Process discovery.

1. Introduction

Process discovery [1, 2], as a study of process mining, discovers process model automatically based on activity records. Several algorithms of process discovery produce different representations of the process model. All representations are divided into two types, i.e. deterministic process model and probabilistic process model.

The deterministic model depicts all activity records. There are several deterministic process models, such as Data Flow Diagram (DFD) [3], Business Process Modeling Notation (BPMN) [4, 5], Activity Diagram [6, 7], and Linear Temporal Logic (LTL) [8, 9]. The probabilistic model removes relationships of activity that have small occurrences in its model. The example of the probabilistic model is Heuristic Net [10].

A reason for the probabilistic model formation is avoiding a spaghetti model as a weakness of the deterministic model. The spaghetti model occurs when a large activity record with a lot of relationships among those activities is modeled in a deterministic model. The deterministic model shows all activities relationships, so the model is complicated, like a bowl of spaghetti. The effect of the spaghetti model is users had difficulty to analyze it. Spaghetti model is found in various fields, such as product development, production, logistics, resource management, and functional area finance or accounting deals [11]. The probabilistic model wants to model a lasagna model, a non-complex model that provides enough information for users.

The pioneer algorithm of the probabilistic model is Heuristic Miner [12, 13]. This algorithm provides a threshold that can be regulated by the user to determine which activities are depicted in the probabilistic model. The threshold is a lower limit of the probability of activities relationships to be modeled. The best threshold for creating a lasagna

model is a topic that should be considered by researchers.

Heuristic Miner continues to be developed by other researchers. The development of Heuristic Miner is Modified Heuristic Miner [14], Modified Time-based Heuristic Miner [14], and Fodina [15]. Fodina focused on reducing duplicate tasks in the result of Heuristic Miner and left it entirely on users to determine the thresholds. Both of Modified Time-based Heuristic Miner and Modified Heuristic Miner uses an average of probabilistic of activities as the threshold. Those algorithms depend on user proficiency or frequent activities to determine the threshold.

This research proposed a new algorithm, namely HLTL, which determines the threshold based on four quality aspects, i.e. Fitness, Precision, Generalization, and Simplicity. HLTL utilizes Linear Temporal Logic [16, 17] to create a formal representation of process model and store the weight of relationships for the threshold formation. Then, HLTL set the threshold to produce the process model. HLTL will uses the threshold that provides a model with the highest averages of those four qualities. In the evaluation, Modified Heuristic Miner is chosen because this algorithm also proposes the automatic threshold (without user proficiency).

To sum up, this research proposes HLTL which has several advantages, which are:

a) providing an automatic threshold considering four quality aspects, i.e. Fitness, Precision, Generalization, and Simplicity,
b) providing formal representations of process model by utilizing Linear Temporal Logic.

2. Preliminary study

2.1 Linear temporal logic (LTL)

LTL is a formal representation that describes some temporal logic referring to time consisting of constants, a group of proportional variable prepositions, logical operator, such as ¬ and ∨, and temporal operators, such as ○, ◊, □, U [16]. LTL use logical operators, such as ∨, ∧, →, ↔, true and false. Logical operators show the relationship between points.

To show the sequence of the process model, LTL uses temporal operator. Sungkono and Sarno [16] describes the temporal operator in LTL in Table 1. ○ is used to described activities happened sequentially. It means if there is activity $U \rightarrow \bigcirc(K)$, Activity *U* must be executed before the process can proceed to activity *K*. Another temporal operator that is used is □. If there is $\Box((K))$, it means that *U* and *K* can be executed in the first, middle, or the last processes. Another operator is ◊. If there is $U \rightarrow \Diamond(K)$, it means that after executing activity *X*, other activities can be executed before executing activity *K*. The last temporal operator is U. *U* U *K* means that *U* will continue to be executed until *K* is ready to use.

Table 1. Operator of Linear Temporal Logic

Symbol	Explanation
○s	s has to hold at the next stage
◊s	s has to hold on the entire sequence path
□s	s has to hold somewhere on the path
sUI	s has to hold until some position I hold

2.2 Heuristic miner algorithm

Heuristic Miner deletes the unnecessary activity by considering the frequency and sequence of events in the process model constructing [18]. The first step in the Heuristic Miner is obtaining the dependency model. The step goal is constructing dependency model which provides the activity records [19]. There are several steps that are used to determine the dependency model, such as calculating the Dependency Threshold [14].

2.2.1. Dependency measure

Dependency measure is calculated by using the frequency-based metrics of each relations. Frequency-based metric is used to indicate the dependency between two event *U* and *V* ($U =>_W V$). For example, if there are 4 activity *U* to activity *V* in the activity records, then the frequency of $U \rightarrow V$ is equal to 4. The obtained frequency will be used as the value of Dependency Measure (DM). The equation to calculate the DM is explained in the Eq. (1). The DM is calculated by reducing frequency of $U \rightarrow V$ with frequency of $V \rightarrow U$, and then the result will be divided using the sum of frequency of $U \rightarrow V$, frequency of $V \rightarrow U$, and 1.

$$U =>_w V = \left(\frac{|U >_w V| - |V >_w U|}{|U >_w V| + |V >_w U| + 1} \right) \quad (1)$$

where:

$U =>_w V$ is value of Dependency Measure from *U* to *V*,

$|U =>_w V|$ is frequency of activity *U* that follows activity *V* directly,

$|V=>_w U|$ is frequency of activity *V* that follows activity *V* directly.

2.2.2. Relative-to-best threshold (RBT)

RBT is used to measures the average of positive Dependency Measure. The equation to calculate the Relative-to-best Threshold is explained in Eq. (2). RBT value is calculated by reducing the average of Positive Dependency Measure (PDM Average) with Standard Deviation of Positive Dependency Measure (SD PDM) that has been divided by 2.

$$RBT = PDM\ Average - \left(\frac{SD\ PDM}{2}\right) \quad (2)$$

where:
PDM Average is the average of positive Dependency Measure,
SD PDM is Standard Deviation of positive Dependency Measure.

2.3 Dependency threshold

Dependency threshold is used to determine the value of Dependency Measure that is below the threshold. If the value of the Dependency Measure is below the Dependency Threshold, the activity will not be used. The equation to calculate the Dependency Measure is explained in the Eq. (3). Dependency Threshold is the result of Average of Positive Dependency Measure that is reduced by Standard Deviation of Positive Dependency Measure.

$$DT = PDM\ Average - SD\ PDM \quad (3)$$

PDM Average is the average of positive Dependency Measure,
SD PDM is standard deviation of positive Dependency Measure.

2.4 Quality aspects

The quality of the business process model can be determined by using four aspects: Fitness, Precision, Simplicity, and Generalization. Fitness is used to measure compatibility between activity records and process models. It means that model can produce every variation in the activity records. Fitness value has a range between 0 to 1. The closer Fitness value to 1, the better the process model is. The equation to calculate Fitness is shown in Eq. (7). The Fitness value is obtained by dividing the number of variation in the activity records that is shown on model with the number of variation in activity records.

$$PCM = \frac{c}{t} \quad (7)$$

where:
c is number of variation in activity records that is shown on model,
t is variation total of activity records.
Precision shows whether the variations of processes derived from a model is same with the variations of activity records. Precision focuses on determining the capability of a model to describe founded behavior in the activity records. The equation to calculate Precision is shown in Eq. (7). The Precision value is obtained by dividing the number of *tp* in a model with the addition of *tp* and *tn* in a model.

$$Q_p = \frac{tp}{p'} \quad (8)$$

where:
tp is true positive number (variation on the activity records that is described in the process model),
p' is addition of true positive (*tp*) and false negative (*tn*) (all variations described in the process model).
To calculate Generalization and Simplicity, the business process model must be converted into tree. Generalization states that a process model should show generalization of the sample processes that exist in the activity records. The equation to calculate Generalization is shown in Eq. (9). The generalization value is obtained by dividing the sum of node operators that is implemented variation on the activity records with the number of node operators implemented variation on the activity records.

$$Q_g = 1 - \frac{\sum_1^{nt}\left(\sqrt{\#executions}\right)^{-1}}{Nt} \quad (9)$$

where:
#nt is node operators number in the process tree,
Nt is node operator that is implemented in the activity records,
#executions is node operators number implemented variation on the activity records.

Simplicity measures whether the process model that is made is as simple as possible without losing the realization of the process captured from the activity records. The equation to calculate

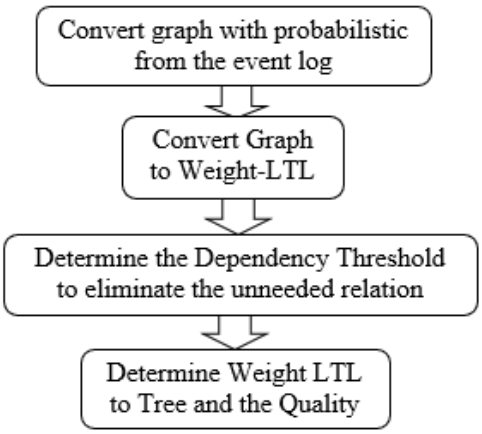

Figure.1 The process of Heuristic LTL

Simplicity is shown in Eq. (10). The simplicity value is obtained by reducing 1 with the sum of activity types number in the activity records and activity types number in the activity records that are not displayed in the process tree that have been divided by the sum of leaf points number and activities number.

$$Q_p = 1 - \frac{\#da + \#ma}{\#npt + \#eventclass} \qquad (10)$$

where:
#da is activity types number in the activity records that is displayed in redundant in the process tree,
#ma is the activity types number in the activity records that are not displayed in the process tree and the number of leaf points that are not in the activity records,
#npt is leaf points number in the process tree,
#eventclass is activities number.

3. Method

This research method proposed in this paper is shown in Fig. 1. This research uses HLTL to build the process model. For the first step, activity records are constructed into a process model. The Weight-Linear Temporal Logic will be determined based on a graph-based model. To eliminate the unneeded activities or relations, Dependency Measure (DM) and Dependency Threshold (DT) are calculated based on the weight in the graph-based model. The quality of Heuristic LTL will be determined after the process tree is formed.

3.1 Heuristic linear temporal logic (HLTL)

HLTL is a combination of Linear Temporal Logic and Heuristic Miner. This method aims to model the activities in the activity records into a simply process model. In Heuristics Linear Temporal Logic, the Dependency Measure that is lower than the Dependency Threshold will be removed so only the important relationships of activities are remained. Frequency of each activity is determined using graph algorithm by calculating the relations of the activity records. The used activity records in this research is explained in the Table 2. The activity records contain information, such as id, activities, Resource and time. Id contains a set of variation in activity records. Activities has several information, i.e. activity name, resource that contains the used threshold, and time that shows the time of activity is executed.

Table 2. Activity records

Id	Activities	Resource	Time
1	Initializing Game	Game System	29-08-2018 16:46:00
1	Collecting company expenditure data every month	Player	29-08-2018 16:46:01
. . .	. . .	. . .	. . .
1	Calculating the highest profit item	BI	29-08-2018 16:46:21
1	Displaying Dashboard		29-08-2018 16:46:22
2	Initializing Game	Game System	29-08-2018 16:46:23
2	Collecting company expenditure data every month	Player	29-08-2018 16:46:24
. . .	. . .	. . .	. . .
2	Calculating ROE	BI	29-08-2018 16:46:44
2	Displaying Dashboard		29-08-2018 16:46:45
. . .	. . .	. . .	. . .
39	Initializing Game	Game System	29-08-2018 16:49:13
39	Collecting company expenditure data every month	Player	29-08-2018 16:49:14
. . .	. . .	. . .	. . .

39	Calculating the highest income item	BI	29-08-2018 16:49:50
39	Displaying Dashboard		29-08-2018 16:49:51
. . .	. . .	. . .	. . .
40	Initializing Game	Game System	29-08-2018 16:49:13
40	Collecting company expenditure data every month	Player	29-08-2018 16:49:14
. . .	. . .	. . .	. . .
40	Calculating the highest income item	BI	29-08-2018 16:49:50
40	Displaying Dashboard		29-08-2018 16:49:51

HLTL is utilizing graph algorithm to determine the relations of each activities. Graph-based algorithm contains points and relations that describe the relationship between activity and sequence of activity. Graph-based algorithm uses operations, such as NEXT, AND JOIN, AND SPLIT, XOR JOIN and XOR SPLIT. NEXT shows the next activity that must be executed. XOR SPLIT shows the activities that must be done next by selecting two of the existing paths. XOR JOIN is the activity that is selected from the previous selected activity. AND SPLIT breaks the activity into 2 parts. It will execute all activities based on sequence. AND JOIN is an activity to unite the previous activities that are divided because of AND SPLIT. In this process, probability of each activity needs to be calculated. The result of graph-based model is shown in the Fig. 2. This graph-based model consists of NEXT, XOR, and AND relations.

The weight is used to determine the weight of each relation. This weight will be used to determine the Dependency Measure and Dependency Threshold. The equation to determine the weight is explained in the Eq. (4) and Eq. (5). The used weight is obtained using probability equation. The probability is determined by calculating the number of frequencies of each relation.

$$Weight\ X \rightarrow Y = P\ (X \rightarrow Y) \qquad (4)$$

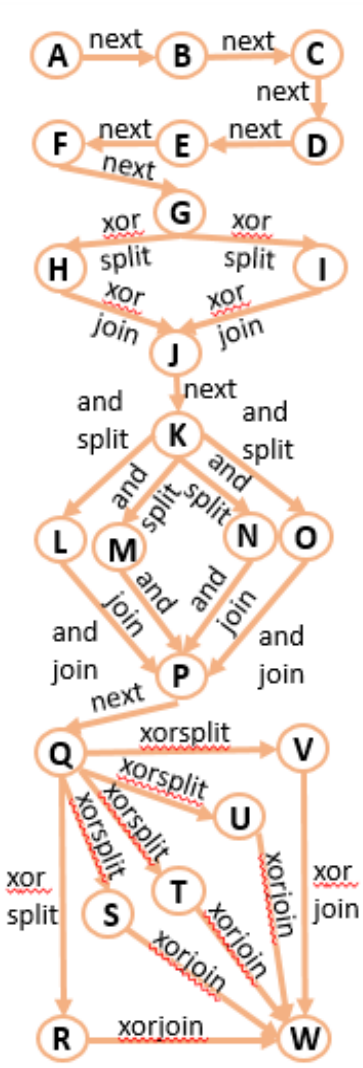

Figure.2 Graph-based model

where:

$$P\ (X \rightarrow Y) = \frac{\sum Relation\ X \rightarrow Y}{\sum Activity\ X} \qquad (5)$$

or

$$P\ (X \rightarrow Y) = \frac{\sum Relation\ X \rightarrow Y}{\sum Another\ Relation \rightarrow Y} \qquad (6)$$

There are two condition to determine the weight of $X \rightarrow Y$. The condition to determine the weight is shown in the Definition 1. If the number of relations $X \rightarrow Y$ is larger than the number of activity X, then the weight is determined using Eq. (4). If the number of relations $X \rightarrow Y$ is smaller than the number of activity X, then the weight is determined using Eq. (5). The result of the weighted graph is shown in the Fig. 3. Each relation has been added with weight based on frequency.

Definition 1 Let the X is activity X and Y is activity Y, $X \rightarrow Y$ *is activity X to Y,* $Y \rightarrow X$ *is activity Y to X,* P

is probability. If $\sum(X \rightarrow Y) \geq \sum(Y \rightarrow X)$ *then* $P(X \rightarrow Y) = \sum(X \rightarrow Y) / \sum(X)$ *else* $P(X \rightarrow Y) = \sum(X \rightarrow Y) / \sum(AR \rightarrow X)$.

To determine the Weigh-LTL, we need to convert the graph-based model to Weigh-LTL model. The result of Weight-LTL is shown in fig. 4. The LTL consist of AND, XOR and NEXT relations. Each activity has a value that shows their weight.

3.2 Calculating the dependency measure and dependency threshold

Dependency Measure table needs to be determined before determining the Dependency Threshold. The weight of each relation is obtained from the graph-based model. Dependency Measure is determined using Eq. (1). Table 3 shows the probability of each relation and its Dependency Measure. The Dependency Measure that is used is consists of 20 data.

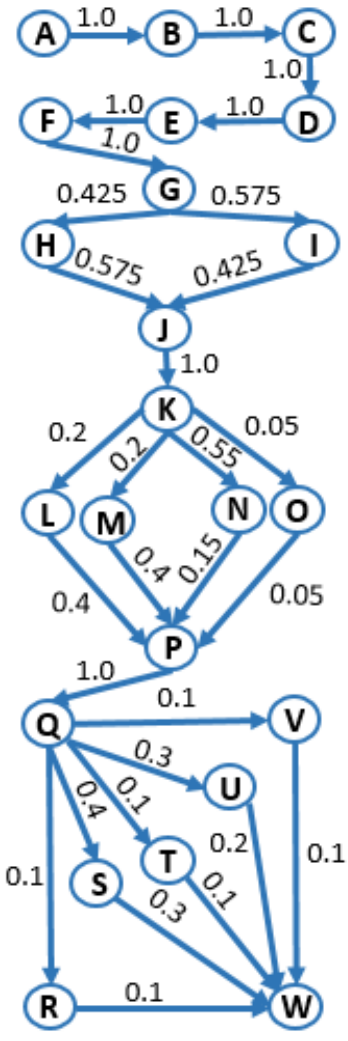

Figure.3 Result of weighted graph

```
Firstactivity (Initializing_Game)
Lastactivity (Displaying_Dashboard)

Initializing_Game -> _O
(Collecting_company_expenditure_data_every_month
1 )

Collecting_company_expenditure_data_every_month
 -> _O ( Updating_product_configuration_data 1 )

Updating_product_configuration_data -> _O
(Choosing_Supplier 1)

Choosing_Supplier -> _O
(Doing_restock_automatically 1)
Doing_restock_automatically -> _O
(Receiving_items_automatically 1)

Receiving_items_automatically -> _O
(Adding_Items_Automatically 1)

Adding_Items_Automatically -> _O
( ( Recording_items_purchase_journal 0.425
 \/ Calculating_Market_Share 0.575 ) )

_O ( ( Recording_items_purchase_journal 0. 575 \/
Calculating_Market_Share 0.425) ) ->
 _O ( Selling_Items_based_on_Market_Share )

Selling_Items_based_on_Market_Share -> _O
( Delivering_Items_automatically 1)

Delivering_Items_automatically -> <>
( ( Calculating_Supplier_Selection 0.2
/\ Calculating_Optimal_Price 0.2
/\ Calculating_ROP 0.55
/\ Calculating_EOQ 0.05) )

<> ( ( Calculating_Supplier_Selection 0.4
 /\ Calculating_Optimal_Price 0.4
 /\ Calculating_ROP 0.15
 /\ Calculating_EOQ 0.05) ) -> _O
(Recording_items_sales_journal )

Recording items_sales_journal -> _O
(Making_maximum_round_checks 1)

Making_maximum_round_checks -> <>
((Calculating_the_highest_profit_item 0.1
 /\ Calculating_the_highest_income_item 0.3
 /\ Calculating_ROI 0.1
 /\ Calculating_ROE 0.4 /\ Calcuating_ROA 0.1))

<> ( ( Calculating_the_highest_profit_item 0.1
 /\ Calculating_the_highest_income_item 0.2
 /\ Calculating_ROI 0.3 /\ Calculating_ROE 0.3
 /\ Calcuating_ROA 0.1) ) -> _O
( Displaying_Dashboard )
```

Figure.4 Result of LTL with weight

Table 3. Probability and Dependency Measure (DM)

No	Relations	Probability	DM
1	(A, B)	1	0.5
2	(B, C)	1	0.5
3	(C, D)	1	0.5
4	(D, E)	1	0.5
5	(E, F)	1	0.5
6	(F, G)	1	0.5
7	(G, H)	0.575	0.365079
8	(G, I)	0.425	0.298246
9	(H, I)	0.575	0.075
10	(H, J)	0.425	0.298246
12	(I, J)	0.575	0.365079
13	(J, K)	1	0.5
14	(K, L)	0.2	0.166667
15	(K, M)	0.05	0.047619
. . .	. . .	. . .	. . .
20	(V, W)	0.1	0.090909

Dependency Threshold (DT) is determined based on Dependency Measure. Dependency Threshold is determined using Eq. (3). The result of Dependency Threshold based on Table 3 is 0.0898.

3.3 Eliminate the relation below dependency threshold

In Heuristic LTL, the relation will be used if their value of Dependency Measure is more than Dependency Threshold. From Table 3, the relation that must be eliminated is relation of $K \rightarrow M$ and of $M \rightarrow P$. Their Dependency Measure value is below the Dependency Threshold, that is why it need to be eliminated. The result of Weight-LTL after elimination process is shown in the Fig. 5. The activities that is eliminated in using Heuristic miner is calculating EOQ.

```
Firstactivity (Initializing_Game)
Lastactivity (Displaying_Dashboard)

Initializing_Game -> _O
(Collecting_company_expenditure_data_every_month
1 )

Collecting_company_expenditure_data_every_month
 -> _O ( Updating_product_configuration_data 1 )

Updating_product_configuration_data -> _O
(Choosing_Supplier 1)

Choosing_Supplier -> _O
(Doing_restock_automatically 1)
Doing_restock_automatically -> _O
(Receiving_items_automatically 1)

Receiving_items_automatically -> _O
(Adding_Items_Automatically 1)

Adding_Items_Automatically -> _O
( ( Recording_items_purchase_journal 0.425
 \/ Calculating_Market_Share 0.575 ) )

_O ( ( Recording_items_purchase_journal 0. 575 \/
Calculating_Market_Share 0.425) ) ->
 _O ( Selling_Items_based_on_Market_Share )

Selling_Items_based_on_Market_Share -> _O
( Delivering_Items_automatically 1)

Delivering_Items_automatically -> <>
( ( Calculating_Supplier_Selection 0.2
/\ Calculating_Optimal_Price 0.2
/\ Calculating_ROP 0.55) )

<> ( ( Calculating_Supplier_Selection 0.4
 /\ Calculating_Optimal_Price 0.4
 /\ Calculating_ROP 0.15) ) -> _O
(Recording_items_sales_journal )

Recording items_sales_journal -> _O
(Making_maximum_round_checks 1)

Making_maximum_round_checks -> <>
((Calculating_the_highest_profit_item 0.1
 /\ Calculating_the_highest_income_item 0.3
 /\ Calculating_ROI 0.1
 /\ Calculating_ROE 0.4 /\ Calcuating_ROA 0.1))

<> ( ( Calculating_the_highest_profit_item 0.1
 /\ Calculating_the_highest_income_item 0.2
 /\ Calculating_ROI 0.3 /\ Calculating_ROE 0.3
 /\ Calcuating_ROA 0.1) ) -> _O
( Displaying_Dashboard )
```

Figure.5 Result of HLTL

The result of HLTL will be calculated by the four quality aspects. HLTL increases the threshold if the lowest value is Generalization and Simplicity, and decreases the threshold if the lowest value is Fitness or Precision. Increase and decrease the threshold are by using Standard Deviation. The iteration of finding threshold is stopped if the next quality is lower or the difference is 0.1.

4. Experimental result

This research is aimed to increase the performance value of process model using retail business process. Table 4 shows the activities and

Table 4 Activities of retail business process

Activities	**Aliases**
Initializing Game	A
Collecting company expenditure data every month	B
Updating product configuration data	C
Choosing Supplier	D
Doing restock automatically	E
Receiving items automatically	F
Adding Items Automatically	G
Calculating Market Share	H
Recording items purchase journal	I
Selling Items based on Market Share	J
Delivering Items automatically	K
Calculating Supplier Selection	L
Calculating EOQ	M
Calculating ROP	N
Calculating Optimal Price	O
Recording items sales journal	P
Making maximum round checks	Q
Calcuating ROA	R
Calculating ROE	S
Calculating ROI	T
Calculating the highest income item	U
Calculating the highest profit item	V
Displaying Dashboard	W

aliases of retail business process. The retail business process consist of 23 activities.

To evaluate each aspect of performance, the tree model of weight LTL is used. The tree model is shown in the Fig. 6. The tree model contains 22 activities that have AND and XOR relationship. From the Eqs. (7), (8), (9), and (10), we can determine the Fitness, Precision, Generalization and Simplicity. In the first experiment, the obtained Dependency Threshold is 0.0898. This is will cause elimination for the relation that has Dependency Measure value below the Dependency Threshold. From the Table 3, $K \rightarrow M$ and $M \rightarrow P$ will be eliminated because the value of Dependency Measure is lower than Dependency Threshold. This elimination process will cause the activity M disappeared from the process model. if the activity M disappeared, the Precision value will be 0.

To avoid this problem, Heuristic LTL needs to increase or decrease the Dependency Threshold value. In this research, we decrease the Dependency Threshold by reducing it using standard deviation.

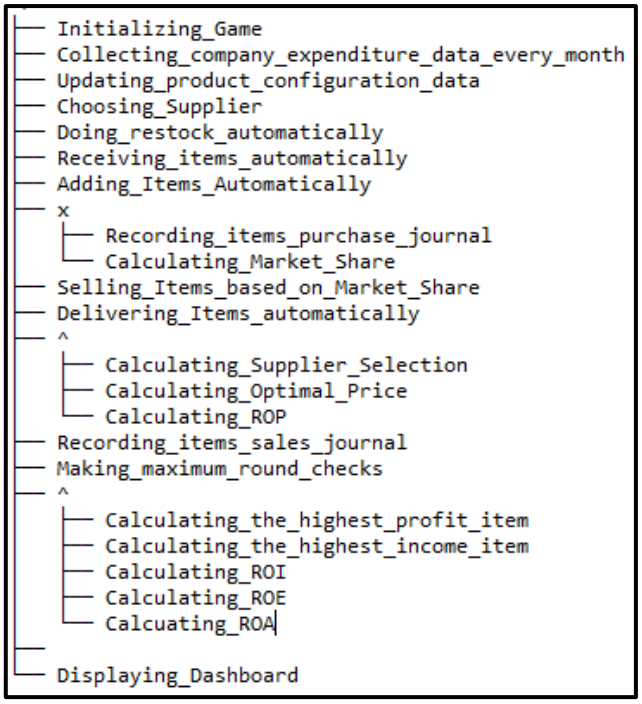

Figure.6 The result of Tree Model

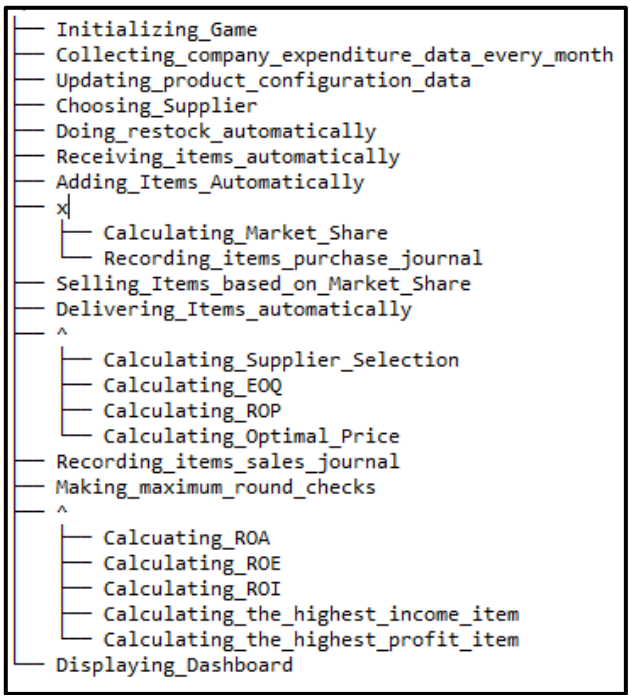

Figure. 7 The result of Tree Model after Dependency Threshold is increased

Fig. 7 shows the tree model after the value of Dependency Threshold is increased. The tree model contains 23 activities that have AND and XOR relationship.

The result shows that the new process model have a better quality in term of Generalization compare to previous method, such as Modified Heuristic Miner. The result of Heuristic LTL

Table 5. Experimental Result

No	Algorithm	F	P	G	S
1	Heuristic LTL	1	1	0.8422	1
2	Modified Heuristic Miner	1	1	0.8421	1

which are :F = Fitness (the range is 0.0 – 1.0)
P = Precision (the range is 0.0 – 1.0)
G = Generalization (the range is 0.0 – 1.0)
S = Simplicity (the range is 0.0 – 1.0)

compared to Modified Heuristic Miner is shown in Table 5. Table 5 compares the performance result of Heuristic LTL and Modified Heuristic Miner. The performance result of both methods does not have a significant different. However, HLTL has a better quality in term of Generalization value.

5. Conclusion

This paper proposes an algorithm of process discovery, namely HLTL. This new algorithm provides an automatic threshold of forming a model by utilizing quality aspects, i.e. Generalization, Simplicity, Fitness, and Precision. Besides that, HLTL gives a formal representation of process model by utilizing Linear Temporal Logic.

There are several steps of HLTL. Firstly, the activity records are modeled by using Linear Temporal Logic with the weight of probability of relations. Then, The Dependency Measure and Dependency Threshold will be calculated based on the weight. All the relation that has weight less than DT will be eliminated. Afterwards, the model will be measured by the quality aspects. HLTL increases the threshold if the lowest value is Generalization and Simplicity, and decreases the threshold if the lowest value is Fitness or Precision. The result of HLTL is a model having highest average of quality aspects.

The result of experiment shows that HLTL has better performance quality in term of Generalization than the modified Heuristic Miner. The generalization value of HLTL is 0.8422 and the generalization value of Modified Heuristic Miner is 0.8421. It happened because the value of Dependency Threshold in the proposed method is lower than value of Dependency Threshold in the Modified Heuristic Miner method.

Acknowledgments

Authors give a deep thank to *Institut Teknologi Sepuluh Nopember*, The Ministry of Research, Technology and Higher Education of Indonesia, *Direktorat Riset dan Pengabdian Masyarakat, and Direktorat Jenderal Penguatan Riset dan Pengembangan Kementerian Riset, Teknologi, dan Pendidikan Tinggi Republik Indonesia* for supporting the research.

References

[1] F. Mannhardt, M. de Leoni, H. A. Reijers, W. M. P. van der Aalst, and P. J. Toussaint, "Guided Process Discovery – A pattern-based approach", *Information Systems*, Vol. 76, pp. 1–18, 2018.

[2] F. M. Maggi, C. Di Ciccio, C. Di Francescomarino, and T. Kala, "Parallel algorithms for the automated discovery of declarative process models", *Information Systems*, Vol. 74, pp. 136–152, 2018.

[3] H. Zhang, W. Liu, H. Xiong, and X. Dong, "Analyzing data flow diagrams by combination of formal methods and visualization techniques", *Journal of Visual Languages and Computing*, Vol. 48, No. August, pp. 41–51, 2018.

[4] A. Peres Penteado, F. Molina Cohrs, A. Diniz Hummel, J. Erbs, R. F. Maciel, C. L. Feijó Ortolani, B. De Aguiar Roza, and I. Torres Pisa, "Kidney transplantation process in Brazil represented in business process modeling notation", *Transplantation Proceedings*, Vol. 47, No. 4, pp. 963–966, 2015.

[5] F. Durán, C. Rocha, and G. Salaün, "Stochastic Analysis of BPMN with Time in Rewriting Logic", *Science of Computer Programming*, No. August, 2018.

[6] G. Karami and J. Tian, "Maintaining accurate web usage models using updates from activity diagrams", *Information and Software Technology*, Vol. 96, No. November 2017, pp. 68–77, 2018.

[7] Z. Daw and R. Cleaveland, "Science of Computer Programming Comparing model checkers for timed UML activity diagrams ☆," *Science of Computer Programming*, Vol. 1, pp. 1–23, 2015.

[8] S. Babenyshev and V. Rybakov, "Unification in linear temporal logic LTL", *Annals of Pure and Applied Logic*, Vol. 162, No. 12, pp. 991–1000, 2011.

[9] V. Rybakov, "Linear temporal logic with until and next, logical consecutions", *Annals of Pure and Applied Logic*, Vol. 155, No. 1, pp. 32–45, 2008.

[10] A. J. M. M. Weijters, W. M. P. van der Aalst, and A. K. A. De Medeiros, "Process Mining with the Heuristics Miner Algorithm",

Technische Universiteit Eindhoven, Tech. Rep. WP, Vol. 166, No. July 2017, pp. 1–34, 2006.
[11] W. M. P. Van Der Aalst and A. J. M. M. Ton Weijters, *Process Mining*. 2005.
[12] R. Sarno, F. Haryadita, D. Sunaryono, and A. Munif, "Model Discovery of Parallel Business Processes using Modified Heuristic Miner", No. 1, pp. 30–35, 2015.
[13] Y. Caesarita, R. Sarno, and K. R. Sungkono, "Identifying bottlenecks and fraud of business process using alpha ++ and heuristic miner algorithms (Case study: CV. Wicaksana Artha)", In: *Proc. of the 11th International Conference on Information and Communication Technology and System*, Vol. 2018–Janua, pp. 143–148, 2018.
[14] R. Sarno, Y. A. Effendi, and F. Haryadita, "Modified Time-Based Heuristics Miner for Parallel Business Processes", *International Review on Computers and Software*, Vol. 11, No. 3, pp. 249–260, 2016.
[15] S. K. L. M. vanden Broucke and J. De Weerdt, "Fodina: A robust and flexible heuristic process discovery technique", *Decision Support Systems*, Vol. 100, pp. 109–118, 2017.
[16] K. R. Sungkono and R. Sarno, "Constructing Control-Flow Patterns Containing Invisible Task and Non-Free Choice Based on Declarative Model", *International Journal of Innovative Computing, Information and Control*, Vol. 14, No. 4, 2018.
[17] J. De Smedt, J. De Weerdt, E. Serral, and J. Vanthienen, "Discovering hidden dependencies in constraint-based declarative process models for improving understandability", *Information Systems*, Vol. 74, pp. 40–52, 2018.
[18] S. Goedertier, J. De Weerdt, D. Martens, J. Vanthienen, and B. Baesens, "Process discovery in event logs: An application in the telecom industry", *Applied Soft Computing Journal*, Vol. 11, No. 2, pp. 1697–1710, 2011.
[19] S. De Cnudde, J. Claes, and G. Poels, "Improving the quality of the Heuristics Miner in ProM 6.2", *Expert Systems with Applications*, Vol. 41, No. 17, pp. 7678–7690, 2014.