Received: December 10, 2018

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

Modified Alpha++ Algorithm for Discovering the Hybrid of Non-Free Choice and Invisible Task of Business Processes

Lukman Hakim[1] **Riyanarto Sarno[1]*** **Kelly R. Sungkono[1]**

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

Abstract: Process discovery is a concern in analyzing business processes. Process discovery methods handle many aspects, e.g. a hybrid of non-free choice and invisible task. A hybrid of non-free choice and an invisible task is a condition that needs additions of invisible tasks in non-free choice relationships. Process discovery methods, e.g. Alpha++, Alpha#, Graphical method, Time-based Alpha Miner, cannot discover the hybrid of non-free choice and invisible task, especially invisible tasks in the overlapping pattern. To modeling a hybrid of non-free choice and invisible tasks, this research proposes an algorithm of adding invisible tasks, including invisible tasks of overlapping pattern, automatically in the event log and processing the event log with the rules of Alpha++ algorithm. To test the proposed algorithm, namely Invisible Task Log with Alpha++, the experiment compares it with Alpha++, Alpha#, Graphical Method, Time-based Alpha Miner, Hidden Markov Model-Parallel, and Coupled Hidden Markov Model-Invisible Task. The experiment results proved that Invisible Task Log with Alpha++ algorithm obtains higher precision than the obtained precision by comparison methods.

Keywords: Event log, Invisible task, Non-free choice, Process discovery

1. Introduction

Process models have been widely utilized for measuring the performance of processes in the company and pointing issues related with those processes [1]. Several areas of issues are deceptions [2-3], communications [4], and environments [5]. Process models can be drawn manually by analysts or can be generated based on the event logs. Many kinds of research, such as Ontology Invisible Task [6], Hidden Markov Model-Parallel (HMM-Parallel) [1], Coupled Hidden Markov Model-Invisible Task (CHMM-Invisible Task) [7], Alpha# [8], Alpha++ [9], Graphical method [10], and Time-based Alpha Miner [11] already suggested methods for discovering process models.

Most activities in a business process are run in sequence. Thereafter, some activities are run in a parallel way [12]. During activities execution, there are several occurring problems, which are invisible tasks and a hybrid of non-free choice and invisible tasks. The primary task of invisible prime tasks is the main task that does not appear in the activity log, but it is added to illustrate the process clearly [12]. A hybrid of non-free choice and invisible tasks is a condition that needs invisible tasks to detect a non-free choice relationship.

Alpha++ algorithm [9] detects non-free choice relation by describing activities of selected relation that depend on other activities. However, in the case of a hybrid non-free choice and invisible tasks, Alpha++ cannot discover a non-free choice relationship because this algorithm cannot detect invisible tasks. There is an algorithm that detects invisible tasks to describe the process completely, i.e. Alpha# algorithm [8]. The kinds of invisible task that can be detected by this algorithm are SKIP, REDO, and SWITCH. However, Alpha# cannot detect invisible tasks in overlapping patterns and also cannot detect non-free choice construct.

Besides Alpha++ and Alpha#, other research proposes several approaches to modeling business processes. The first approach is Graphical Method [10]. Graphical Method handles parallel cases of

processes. The second approach is Time-Based Alpha Miner [11]. Time-Based Alpha Miner is a modification of the Alpha Miner [13] algorithm that considers the sequence of activities and time interval information from the event log to find parallel business process models. Afterward, HMM-Parallel [1] is another approach that uses the Baum-Welch method and the double time-stamped event log to discover parallel relationships. The development of HMM-Parallel [14] has been able to find models of processes related to the invisible task, but have not been able to handle non-free choices. To sum up, these four approaches cannot model the hybrid of non-free choice and invisible tasks, especially invisible tasks in the overlapping pattern.

To overcome the inability of those algorithms, this study proposes an algorithm, namely Invisible Task Log with Alpha++. Invisible Task Log with Alpha++ algorithm adds invisible tasks, including invisible tasks in overlapping patterns by adding it automatically to the existing event log. This algorithm proposes rules to find the location of additional invisible tasks in the event log. Then, the invisible tasks are inserted into the event log based on the discovered location. The obtained event log is called an event log with additional invisible tasks. Then, the event log with additional invisible tasks is processed by Alpha++ to discover a process model which has a hybrid non-free choice and invisible tasks.

To test Invisible Task Log with Alpha++ algorithm, this experiment compares this algorithm with Graphical Method [10], Time-Based Alpha Miner [11], HMM-Parallel [1], and CHMM-Invisible [14]. The experiment uses an event log that has a condition of a hybrid non-free choice and invisible tasks and has a condition of invisible tasks in overlapping patterns. SKIP invisible task and overlapping patterns are the pattern of invisible tasks which is used in this study.

To reaffirm, this study does the following things, which are:

1) Proposing an algorithm of discovering a hybrid non-free choice and invisible task that creates rules to adds invisible tasks in an event log automatically and processes the event log with Alpha++;
2) Comparing the algorithm, namely Invisible Task Log with Alpha++, with other comparison methods to test the ability of Invisible Task Log with Alpha++.

2. Research method

The research method explains the studies that supports the proposed algorithm, namely Invisible Task Log with Alpha++.

2.1 Event log

An event log contains a set of information concerning the processes, such as the performer of tasks, the name of the tasks, and the starting period or ending period of tasks [2]. That information can be used to build an improved system. The techniques of process mining are helpful to analysing the information since they collect information about the actual events in accordance with the event log of an organization [12].

The event log examples are shown in Table 1 and Table 2. There are three cases and three traces on Table 1 while there are seven cases and seven traces on Table 2. Cases are activity flows that are recorded in the log and traces are variants of cases. Because each case in those tables has different sequence of activities, the number of traces is the same with the number of cases. Then, in the log, CaseId describes the identity of the case and TaskName describes the running tasks.

Table 1. Log of Fig. 2

CaseId	TaskName	CaseId	TaskName
P01	ActA	P02	ActB
P01	ActB	P02	ActE
P01	ActC	P03	ActA
P01	ActE	P03	ActD
P02	ActA	P03	ActE
P02	ActC		

Table 2. Log of Fig. 3

CaseId	TaskName	CaseId	TaskName
P01	ActA	P04	ActD
P01	ActB	P04	ActF
P01	ActC	P05	ActA
P01	ActF	P05	ActD
P02	ActA	P05	ActB
P02	ActB	P05	ActF
P02	ActD	P06	ActA
P02	ActF	P06	ActD
P03	ActA	P06	ActC
P03	ActC	P06	ActF
P03	ActB	P07	ActA
P03	ActF	P07	ActE
P04	ActA	P07	ActF
P04	ActC		

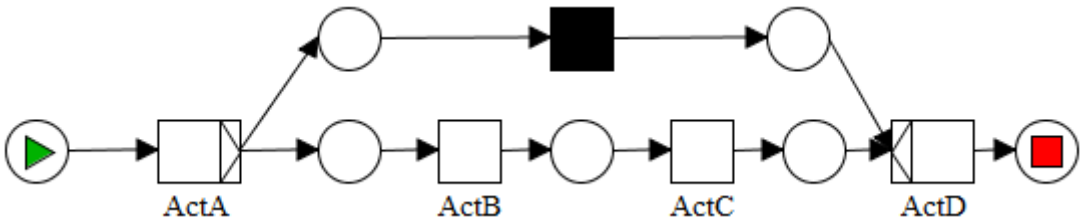

Figure.1 A model of invisible task in skip condition by YAWL

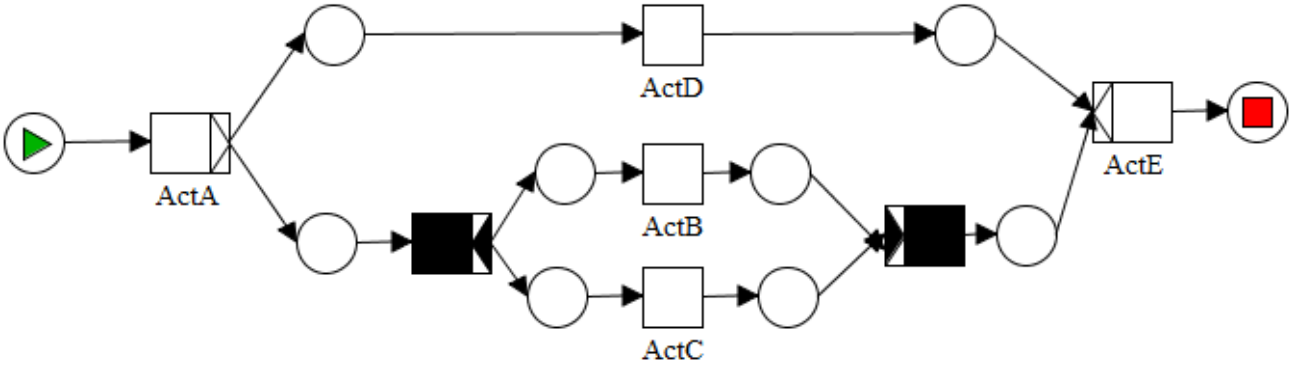

Figure.2 A model of invisible tasks in overlapping pattern XOR gates and AND gates

2.2 Non-free choice

Non-free choice contains a mixture of synchronization and choice. The non-free choice is marked by additional arrows that show an activity appointed by the arrowhead will be executed if an activity appointed by the nock of the arrow is executed. Alpha++ algorithm [11] is able to model non-free choice. However, if there are specific conditions, such as a skip condition, Alpha ++ have difficulty to model non-free choice. Those specific conditions can be handled by adding invisible tasks. Therefore, this study provides a technique of adding invisible tasks in the event log for handling those specific conditions.

2.3 Invisible task in the case of skip and overlapping pattern

Semantic measurements according to Gomaa and Fahmy [15] are classified into three different groups, namely String-based similarity measure, Corpus-based similarity measure, and Knowledge-based similarity measure. Semantic measurements related to the utilization of information contained in wordnet are included in the Knowledge-based similarity measure group. The Knowledge-based similarity measure group is divided into two different parts, namely the semantic similarity measure and the semantic relatedness measure [15].

An invisible task is a task that exists in the model process for discovering the real case. It is difficult to be detected because the task have never been recorded in the event log. In the real case, an invisible task is justified, because a condition enables to skip several activities. Therefore, a process model is required to model invisible tasks.

Alpha++ algorithm and Alpha# algorithm has the same problem in modeling cases that have skip condition or need parallel overlapping. The result of Alpha++ algorithm and Alpha# algorithm are different from the initial model when modeling those cases. To fit the resulting model with the initial model, it is necessary to add invisible tasks when processing those cases.

Skip condition is a condition wherein some activities are skipped deliberately. Fig. 1 illustrates the condition through the activity in the trace. Skip condition in Fig. 1 occurs when activity ActA goes directly to activity ActD without going through activities ActB and ActC. The black box in Fig. 1 denotes the invisible task for showing skip condition.

The overlapping pattern is a condition when two patterns are modeled without any activities between them. The patterns are XOR gates with AND gates and XOR gates with OR gates. The XOR gate is needed in a relationship that only chooses one path. The OR gate describes a condition when the executed path can be chosen flexibly while the AND gate signifies all paths must be executed simultaneously [1].

Fig. 2 illustrates the usage of invisible tasks in the overlapping pattern XOR gates with AND gates. This figure is modeled using YAWL. Triangles in

box ActA and box ActE denote XOR gates and triangles in invisible tasks (black boxes) denote AND gates. In Fig. 2, invisible tasks of parallel overlapping is needed because only two chosen paths, ActA → ActD → ActE and ActA → ActB, ActC → ActE. The chosen paths are based on the log in Table 1. For the trace with first CaseId, the subsequent activity of ActA is activity ActB and then ActC. While on a trace with the second caseId, the sequence of activities is activity ActA, then activity ActC, and then activity ActB. In the last trace, activity ActC and ActB are not executed, but activity ActD is executed after activity ActA. This condition needs invisible task in parallel overlapping XOR gates with AND gates.

Fig. 3 illustrates the usage of invisible tasks in the overlapping pattern XOR gates with OR gates. In contrast to Fig. 2, triangles in invisible taks denote OR gates. Fig. 3 is discovered by following the log in Table 3. All cases on the table declare ActB, ActC, and ActD can be chosen flexibly while ActE can be chosen if those three activities are not executed. OR gates illustrate the condition of ActB, ActC, and ActD while XOR gates illustrate the condition of ActE. In order for both kinds of gates can be drawn in the model, invisible tasks are added. Those invisible tasks are black boxes in Fig. 3.

2.4 Hybrid on non-free choice and invisible task

In Fig. 4, alpha++ algorithm cannot discover a non-free choice relationship because this algorithm cannot detect invisible tasks. This algorithm can detect invisible tasks when invisible tasks are added to overlapping patterns automatically in the event log. Fig. 5, is the result of a hybrid non-free choice and overlapping pattern with the addition of invisible tasks.

2.5 Fitness and precision

Fitness measures how many cases that are captured in a model, meanwhile precision measures how many traces from the model that are captured in the log. Fitness is obtained from a number of cases discovered on the model divided by a number of cases contained in the log while precision is obtained from a number of traces recorded in the log divided by a number of traces contained in the model. For example, Fig. 2 is a model of a case with an overlapping pattern of XOR gates and AND gates with the inserted invisible task. In Fig. 2, the traces discovered from the model are [ActA, ActB, ActC, ActE], [ActA, ActC, ActB, ActE] and [ActA, ActD, ActE] so that the number of traces of the model is three. While the number of traces contained in the model is also three, so the fitness value of the model is 3/3=1.0 with a precision value of 3/3 =1.0. The equation used for fitness is shown in Eq. (1), whereas precision is shown in Eq. (2).

$$Fitness = \frac{n(Case\_Captured\_in\_Model}{n(Case\_of\_Log)} \quad (1)$$

$$Precision = \frac{n(Trace\_Captured\_in\_Log}{n(Trace\_of\_Model)} \quad (2)$$

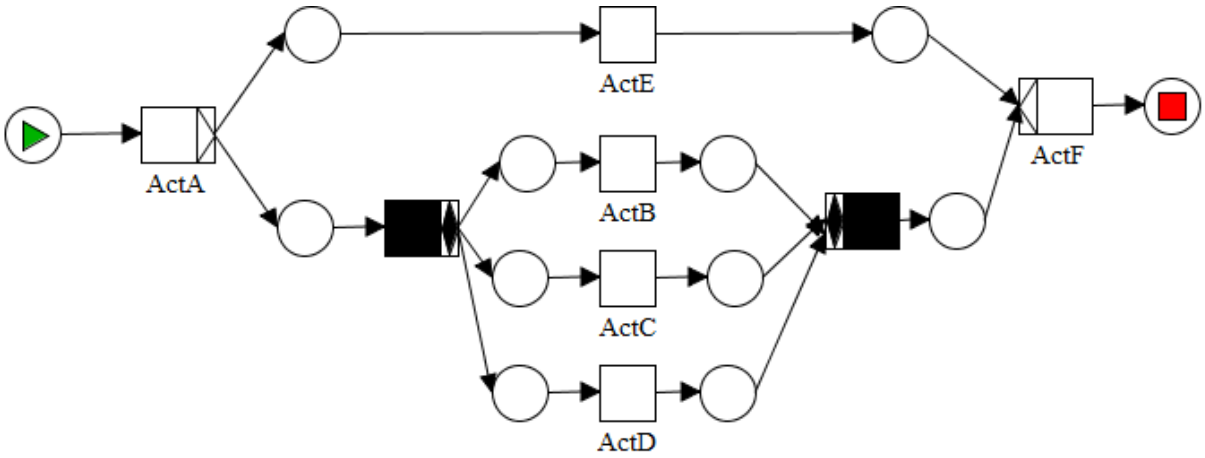

Figure.3 A model of invisible tasks in overlapping pattern XOR gates and OR gates

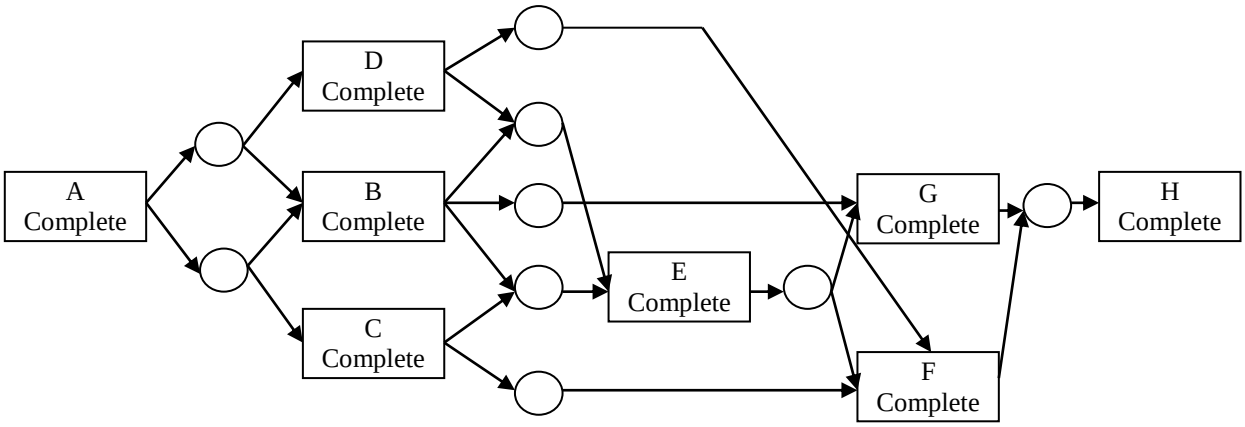

Figure.4 The model of case hybrid non-free choice and overlapping invisible task by Alpha++ or Alpha# Algorithm

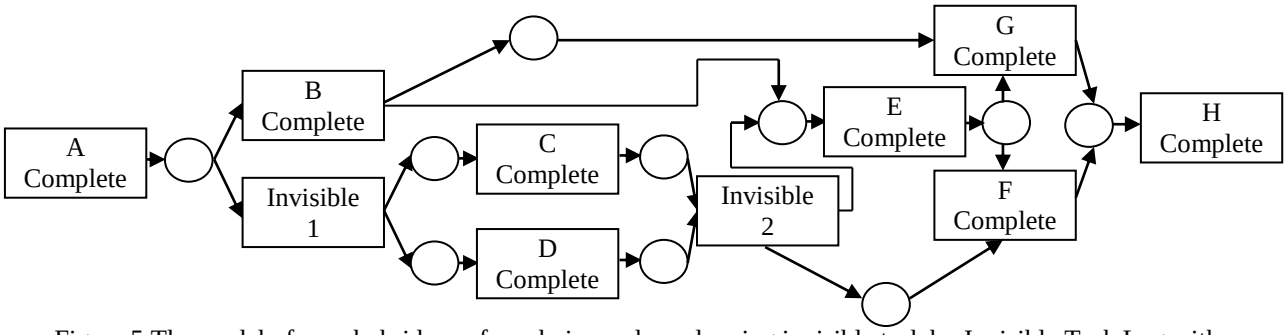

Figure.5 The model of case hybrid non-free choice and overlapping invisible task by Invisible Task Log with Alpha++

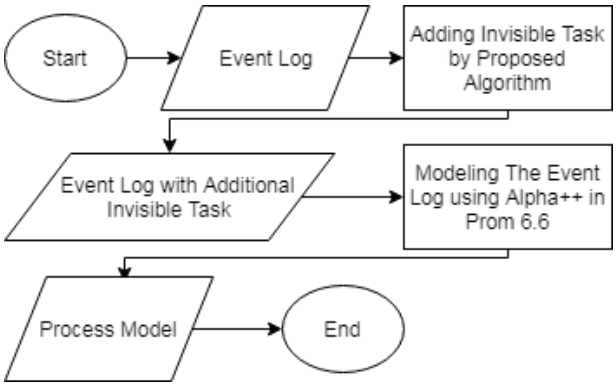

Figure.6 Flowchart proposed algorithm (Invisible Task Log with Alpha++)

2.6 Flowchart of the proposed algorithm

A workflow of Invisible Task Log with Alpha++ algorithm refers to Fig. 6. Firstly, original event logs are processed using the algorithm. The algorithm detects and adds additional invisible tasks in the event log. After that, the new event log is modeled by Alpha++ algorithm that is already existed in Prom 6.6. The new event log is a generated event log by Invisible Task Log with Alpha++ algorithm. The event log is in excel format. Thus, it must be converted to MXML format for Alpha++ algorithm in Prom 6.6 to be able to model the log.

2.7 Pseudocode of the proposed algorithm

A pseudocode of Invisible Task Log with Alpha++ refers to Table 3. The algorithm detects the locations for the added invisible tasks in the log. There are 16 lines in the pseudocode to determine invisible tasks, especially invisible tasks of case skip activity and invisible tasks of case overlapping pattern. First, Invisible Task Log with Alpha++ algorithm counts a number of subsequent activities namely quantity actAfter (QAF), a number of foregoing activities namely quantity actBefore (QBF), a number of activity appearances namely quantity actAppear (QAP), a number of occurrences of subsequent activity for each trace namely quantity actAppearAfterwards (QAPA), and a number of traces namely quantity Trace (QT) for every activity. Then, the algorithm checks QAF. If the QAF is more than one, the second line until the seventh line will be executed. Based on those lines, there are three conditions of adding an invisible task between an activity and its subsequent activity. The

first condition is both of QAP and QAPA having similar values to the value of QT. This condition is for case skip activity. A second condition is QAP has similar value to QT and QAPA has similar value to QAP. The third condition detects case overlapping pattern. Because there are two invisible tasks that are added in the case overlapping pattern, the eight until twelfth line determine the addition of second invisible task for case overlapping pattern.

Table 3. Pseudocode for case study skip and overlapping pattern

	A Method of Obtaining Invisible Task for Skip and Overlapping
1	if QAF > 1 then
2	if ((QAP == QT) && (QAPA == QT)) then
3	Add Invisible_Task
4	else if ((QAP == QT) && (QAPA == Q1)) then
5	Add Invisible_Task
6	Endif
7	Endif
8	if QBF > 1 then
9	if ((QAP == Q1) && (QAPA == QT)) then
10	Add Invisible_Task
11	Endif
12	Endif

where :

actAfter : subsequent activity
actBefore : foregoing activity
actFirst : first activity in a relation
actLast : last activity in a relation
QAF : quantity actAfter
QAP : quantity actAppear
QBF : quantity actBefore
QAPA : quantity actAppearAfterwards
QT : quantity Trace
Q1 : Number of Second Highest from QAP

Table 4. The union of activity after, before, appear, and appear afterwards based on Table 1

Traces	**Activity**	**QAF**	**QBF**	**QAP**	**QAPA**
T01	ActA	3	0	3	2
T01	ActB	2	2	2	2
T01	ActC	2	2	2	3
T01	ActE	0	3	3	3
T02	ActA	3	0	3	2
T02	ActC	2	2	2	2
T02	ActB	2	2	2	3
T02	ActE	0	3	3	3
T03	ActA	3	0	3	1
T03	ActD	1	1	1	3
T03	ActE	0	3	3	0

Table 5. Results of addition of invisible task based on Table 1

CaseId	**TaskName**	**CaseId**	**TaskName**
P01	ActA	P02	ActC
P01	InvisibleTask	P02	ActB
P01	ActB	P02	InvisibleTask
P01	ActC	P02	ActE
P01	InvisibleTask	P03	ActA
P01	ActE	P03	ActD
P02	ActA	P03	ActE
P02	InvisibleTask		

Table 4 is an example of QAF, QBF, QAP, QAPA for every activity in each trace. Table 4 is obtained based on the log illustrated in Table 1. QAF is obtained by calculating the number of subsequent activities. A subsequent activity is an activity that is executed exactly after another activity based on the log. ActA has ActB as its subsequent activity based on the first trace, has ActC as its subsequent activity based on the second trace, and ActD as its subsequent activity based on the third trace. Because of that, QAF of ActA is 3. QBF is obtained by calculating the number of foregoing activities. A foregoing activity is an activity that is executed exactly before another activity. ActB has two foregoing activities, ActA and ActC. Because of that, QBF of ActB is two. QAP is the number of activity appearance based on traces of the log. Because ActA appears on the first, second, and third traces, QAP of ActA is 3. QAPA is obtained by calculating the occurrences of subsequent activity in each trace. In the first trace, QAPA of ActA is two because the number of appearances of ActB is two. Then, in the last trace, QAPA of ActA is one because the number of appearances of ActD is one.

Using Invisible Task Log with Alpha++ algorithm, the results of the addition of invisible tasks can be seen in Table 5. Invisible tasks are added between ActA and ActB and between ActA and ActC because ActA fulfills a statement in the fourth line of the pseudocode. It is proven by the number of QAP of ActA is three, same as the number of traces and the number of QAPA of ActA is two, same as the second highest score of all QAPs. For other invisible tasks, there are added because ActC in the first case and ActB in the second case fulfill the ninth line of the pseudocode.

3. Result and analysis

The experiments in this research are performed using 1) the processes of Port Container Handling,

and 2) simulation processes that describe a hybrid of non-free choice relation and overlapping pattern. The model of Port Container Handling processes is illustrated in Fig. 7. Port Container Handling processes contain the hybrid of non-free choice relation and skip invisible tasks.

Fig. 4, Fig. 5, Fig. 8 until Fig. 10 show models of a hybrid non-free choice and overlapping pattern case XOR gates with AND gates. Then, Fig. 11 until Fig. 15 show models of a hybrid non-free choice and skip invisible tasks. Fig. 5 and Fig. 12 are depicted by Invisible Task Log with Alpha++ algorithm. In Fig. 4, the non-free choice relations are not well illustrated by Alpha++ or Alpha# because of their inability in discovering an overlapping pattern. On the contrary, Fig. 8 until Fig. 10 show the other comparison methods cannot depict non-free choice relations.

For the case of Port Container Handling, Alpha++ cannot depict non-free choice related to Determine Dry because Alpha++ cannot discover invisible tasks. It can be seen in Fig. 11. Based on Fig. 13 until Fig. 15, other comparison methods, i.e. Graphical method, Time-Based Alpha++, HMM-Parallel, and CHMM-Invisible cannot depict non-free choice relationships.

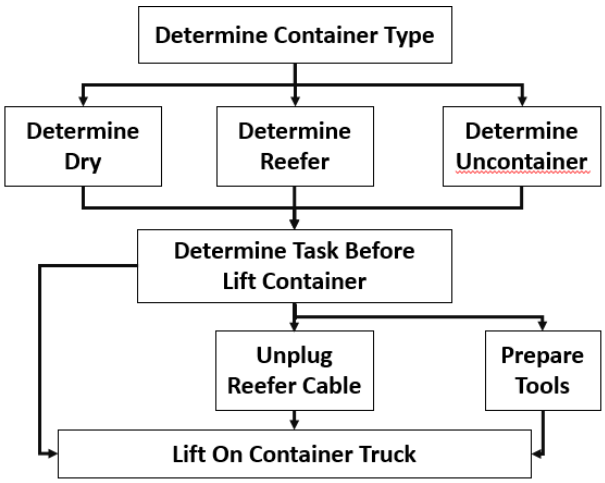

Figure.7 Port Container Handling event log modeling by Disco

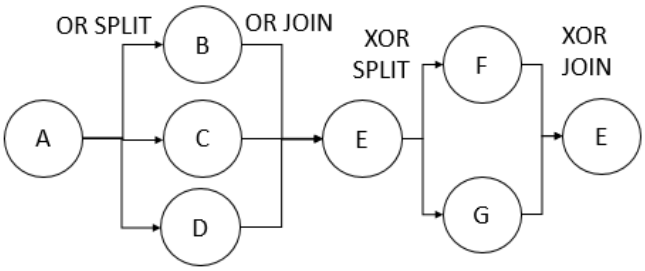

Figure.8 The model of case hybrid non-free choice and overlapping invisible task by Graphical Method or HMM-Parallel

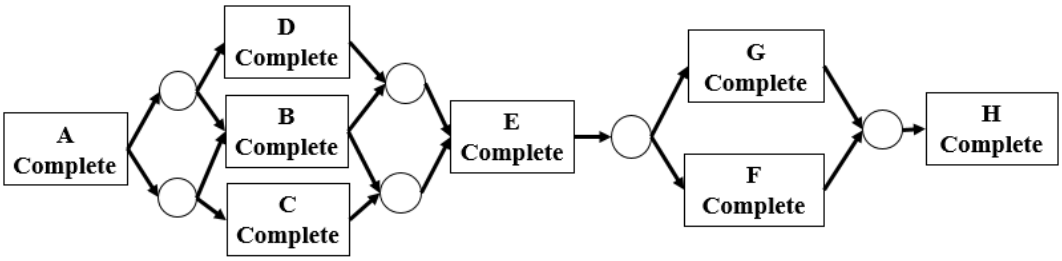

Figure.9 The model of case hybrid non-free choice and overlapping invisible task by Time-Based Alpha Miner

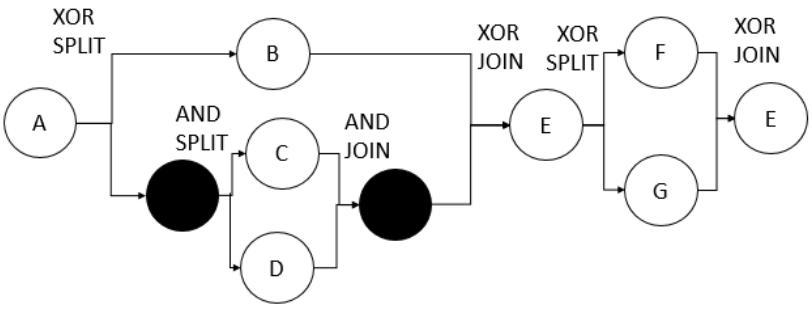

Figure.10 The model of case hybrid non-free choice and overlapping invisible task by CHMM-Invisible Task

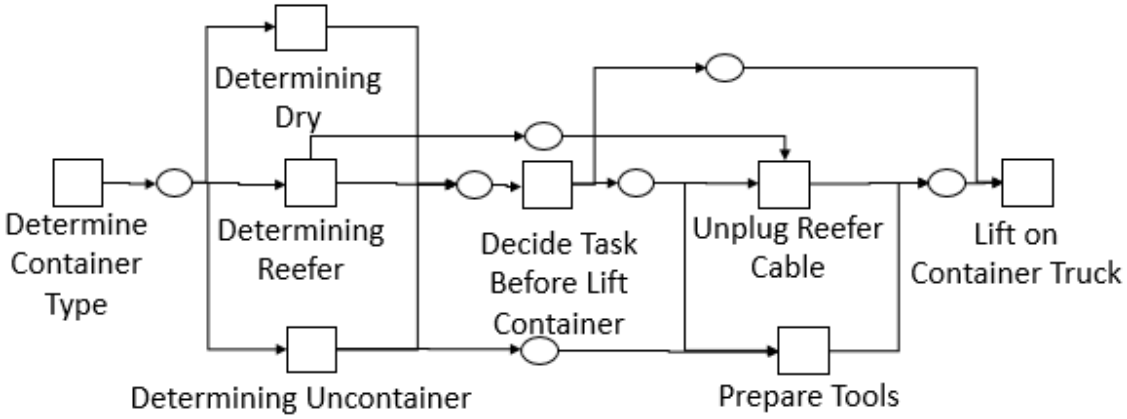

Figure.11 The model by Alpha++ of Port Container Handling

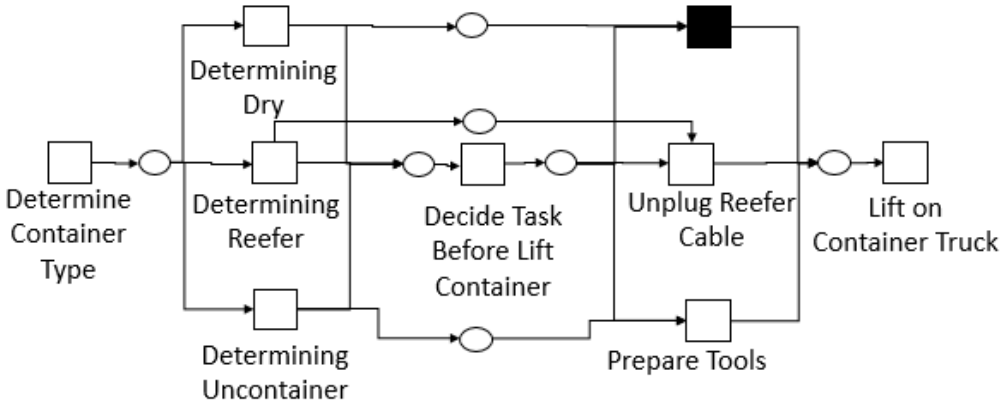

Figure.12 The model by Invisible Task Log with Alpha++ of Port Container Handling

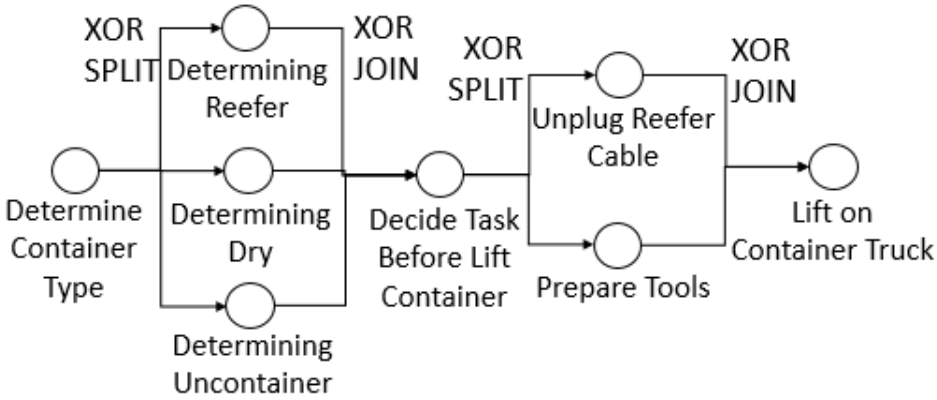

Figure.13 The model by Graphical Method or HMM-Parallel of Port Container Handling

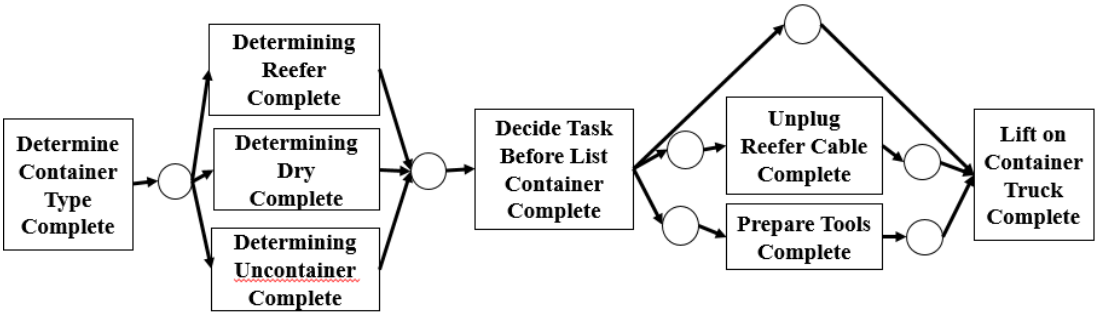

Figure.14 The model by Time-Based Alpha++ of Port Container Handling

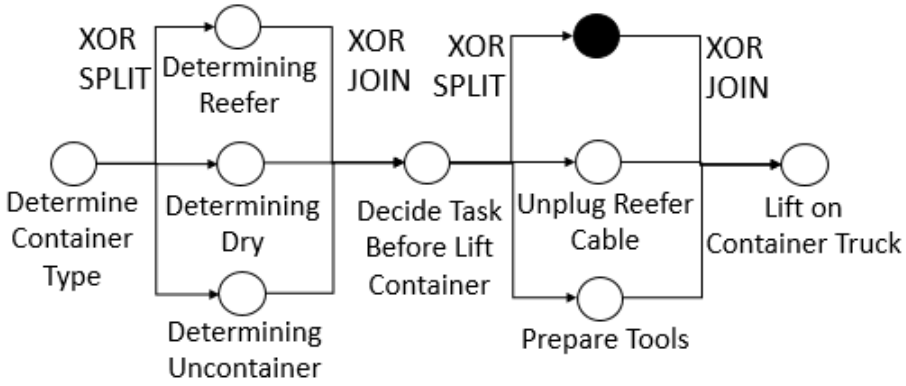

Figure.15 The model by CHMM-Invisible of Port Container Handling

Table 6. Result of evaluation on Fitness and Precision for all Cases

Cases	Model	Fitness	Precision
Hybrid non-free choice and overlapping invisible task	Proposed Algorithm (Invisible Task Log with Alpha++)	1	1
	Original Event Log with Alpha++ [9] or Alpha#[8]	1	0.20
	Graphical Method[10]	0.67	0.17
	Time-Based Alpha Miner[11]	0.67	0.17
	HMM-Parallel [1]	0.67	0.17
	CHMM-Invisible [14]	1	0.50
Port Container Handling Processes	Proposed Algorithm (Invisible Task Log with Alpha++)	1	1
	Original Event Log with Alpha++[9]	1	0.31
	Graphical Method[10]	0.67	0.33
	Time-Based Alpha Miner[11]	1	0.33
	HMM-Parallel [1]	0.67	0.33
	CHMM-Invisible [14]	1	0.33

4. Conclusion

This research proposes Invisible Task Log with Alpha++ algorithm to discover the hybrid of invisible tasks, including invisible tasks in an overlapping pattern, and non-free choices. This algorithm can discover overlapping patterns, which are not described by Alpha#, the existing algorithm of invisible tasks.

There are several steps in this research. Firstly, Invisible Task Log with Alpha++ algorithm detects the invisible tasks in an event log and adds invisible tasks automatically in the event log. Then, the inserted event log is processed by Alpha++ to discover a process model containing the hybrid of invisible tasks and non-free choices.

Invisible Task Log with Alpha++ will be compared with Alpha++, Alpha#, Graphical Method, Time-based Alpha Miner, HMM-Parallel, and CHMM-Invisible in the evaluation. The results of the evaluation show the model by Invisible Task Log with Alpha++ algorithm gives better models than those by Alpha++, Alpha#, Graphical Method, Time-based Alpha Miner, HMM-Parallel, and CHMM-Invisible. In the hybrid non-free choice and overlapping pattern, Invisible Task Log with Alpha++ algorithm has the highest precision because other comparison methods have inabilities, i.e. Graphical method, CHMM, HMM, Time-based Alpha Miner cannot discover non-free choice, subsequently Graphical method, HMM, Time-based Alpha Miner, Alpha++ and Alpha# cannot discover overlapping pattern. Afterward Invisible Task Log with Alpha++ also has the highest precision among those comparison methods. To sum up, the evaluation proves that the precision of the model by Invisible Task Log with Alpha++ is the highest precision among the precisions by the others. The concerns of the research are invisible tasks of skip processes and overlapping patterns. Handling other kinds of invisible tasks are addressed in future research.

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] R. Sarno and K. R. Sungkono, "Hidden Markov Model for Process Mining of Parallel Business Processes", *International Review on Computers and Software*, Vol. 11, No. 4, pp. 290–300, 2016.

[2] D. Rahmawati, M. A. Yaqin, and R. Sarno, "Fraud detection on event logs of goods and services procurement business process using Heuristics Miner algorithm", In: *Proc. of the 2016 International Conference on Information Communication Technology and Systems*, pp. 249–254, 2016.

[3] G. Baader and H. Krcmar, "Reducing false positives in fraud detection: Combining the red flag approach with process mining", *International Journal of Accounting Information Systems*, Vol. 31, No. March, pp. 1–16, 2018.

[4] V. R. L. Shen, H.-Y. Lai, and A.-F. Lai, "The implementation of a smartphone-based fall detection system using a high-level fuzzy Petri net", *Applied Soft Computing*, Vol. 26, pp. 390–400, 2015.

[5] A. Sanaa, S. Ben Abid, A. Boulila, C. Messaoud, M. Boussaid, and N. Ben Fadhel, "Modeling hydrochory effects on the Tunisian island populations of Pancratium maritimum L. using colored Petri nets", *Biosystems*, Vol. 129, pp. 19–24, 2015.

[6] K. R. Sungkono, R. Sarno, and N. F. Ariyani, "Refining business process ontology model with invisible prime tasks using SWRL rules", In: *Proc. of the 2017 11th International Conference on Information Communication Technology and System*, pp. 215–220, 2017.

[7] R. Sarno and K. R. Sungkono, "Coupled Hidden Markov Model for Process Discovery of Non-Free Choice and Invisible Prime Tasks", *Procedia Computer Science*, Vol. 124, pp. 134–141, 2018.

[8] L. Wen, J. Wang, W. M. P. Van Der Aalst, B. Huang, and J. Sun, "Mining process models with prime invisible tasks", *Data and Knowledge Engineering*, Vol. 69, No. 10, pp. 999–1021, 2010.

[9] L. Wen, W. M. P. Van Der Aalst, J. Wang, and J. Sun, "Mining process models with non-free-choice constructs", *Data Mining and Knowledge Discovery*, Vol. 15, No. 2, pp. 145–180, 2007.

[10] R. Sarno, K. R. Sungkono, and R. Septiarakhman, "Graph-Based Approach for Modeling and Matching Parallel Business Processes", Information, Vol. 21, No. 5, pp. 1603–1614, 2018.

[11] Y. A. Effendi and R. Sarno, "Modeling parallel business process using modified time-based alpha miner", *International Journal of Innovative Computing, Information and Control*, Vol. 14, No. 5, pp. 1565–1582, 2018.

[12] R. Sarno and K. R. Sungkono, "Coupled hidden Markov model for process mining of invisible prime tasks", *International Review on Computers and Software*, Vol. 11, No. 6, pp. 539–547, 2016.

[13] A. J. M. M. Weijters, W. M. P. Van Der Aalst, and A. K. A. de Medeiros, "Process Mining with the HeuristicsMiner Algorithm", *Cirp Annals-manufacturing Technology*, Vol. 166, pp. 1–34, 2006.

[14] K. R. Sungkono and R. Sarno, "CHMM for discovering intentional process model from event logs by considering sequence of activities", In: *Proc. of the 2017 4th International Conference on Electrical Engineering, Computer Science and Informatics*, pp. 1–6, 2017.

[15] W. H. Gomaa, "A Survey of Text Similarity Approaches", *International Journal of Computer Applications*, Vol. 68, No. 13, pp. 13–18, 2013.