TELKOMNIKA, Vol.16, No.6, December 2018, pp.2809~2816
ISSN: 1693-6930, accredited First Grade by Kemenristekdikti, Decree No: 21/E/KPT/2018
DOI: 10.12928/TELKOMNIKA.v16i6.10568

Anomaly Detection based on Control-flow Pattern of Parallel Business Processes

Hendra Darmawan*, Riyanarto Sarno, Adhatus Solichah Ahmadiyah, Kelly Rossa Sungkono, Cahyaningtyas Sekar Wahyuni
Department of Informatics, Institut Teknologi Sepuluh Nopember
Kampus ITS Sukolilo, Jalan Raya ITS, Surabaya, Jawa Timur 60111, (031) 5994251, Indonesia
*Corresponding author, e-mail: hendradn96@gmail.com

Abstract

The purpose of this paper was to discover an anomalous-free business process model from event logs. The process discovery was conducted using a graph database, specifically using Neo4J tool involving trace clustering and data filtering processes. We also developed a control-flow pattern to address, AND relation between activities named parallel business process. The result showed that the proposed method improved the precision value of the generated business process model from 0.64 to 0.81 compared to the existing algorithm. The better outcome is constructed by applying trace clustering and data filtering to remove the anomaly on the event log as well as discovering parallel (AND) relation between activities.

Keywords: *anomaly data filtering, control-flow pattern, graph database, process discovery*

1. Introduction

Process discovery has been intensively used to generate business process model [1] as it is beneficial in supporting the companies to understand the business process, identify possible anomalies, monitor and control the executed operation [2]. The generated business process model represents the behavior of performed activities mined from event logs [3].

There are several algorithms introduced to mine the event logs, such as Alpha Miner [4], Heuristic Miner [5, 6], Alpha-T [7], Coupled Hidden Markov Model of Non-free Choice and Invisible Prime task (CHMM-NCIT) [8], Hidden Markov Model of Parallel Business Process (HMM-Parallel) [9], and Coupled Hidden Markov Model of Invisible Task (CHMM-Invisible) [1].

The recent researches on business process discovery are on anomaly detection. An Alpha algorithm is a basic algorithm to generate a business process model from the event logs [10], but it is limited to mining short loops [11]. Alpha+ algorithm was then introduced to solve Alpha algorithm problem, but it failed to mine non-free choice or implicit dependencies anomaly [12]. Alpha++ algorithm answered short loop mining and non-free choice problems, however, it was unable to mine invisible task anomaly which then addressed by Alpha# algorithm [13]. Alpha$ algorithm [14] was able to discover both invisible task and non-free choice anomalies simultaneously. However, Alpha miner algorithms as the above-mentioned processed event logs directly without data filtering, anomaly business process due to the existence of low-frequency values on several trace clusters.

Therefore, this paper proposed a method to tackle this problem by using a graph database incorporating trace clustering and data filtering. Graph database was used since it is flexible to store interconnected activities of business [15], to show the relations among activities [16], and to adjust business needs [17]. Neo4J was employed along with Cypher Query Language (CQL) [18].

2. Research Method

Figure 1 shows our proposed method to generate an anomalous-free business process model. It consists of two main stages. The first one is the anomaly deletion stage, while the second stage deals with business process discovery.

Received July 15, 2018; Revised September 7, 2018; Accepted October 8, 2018

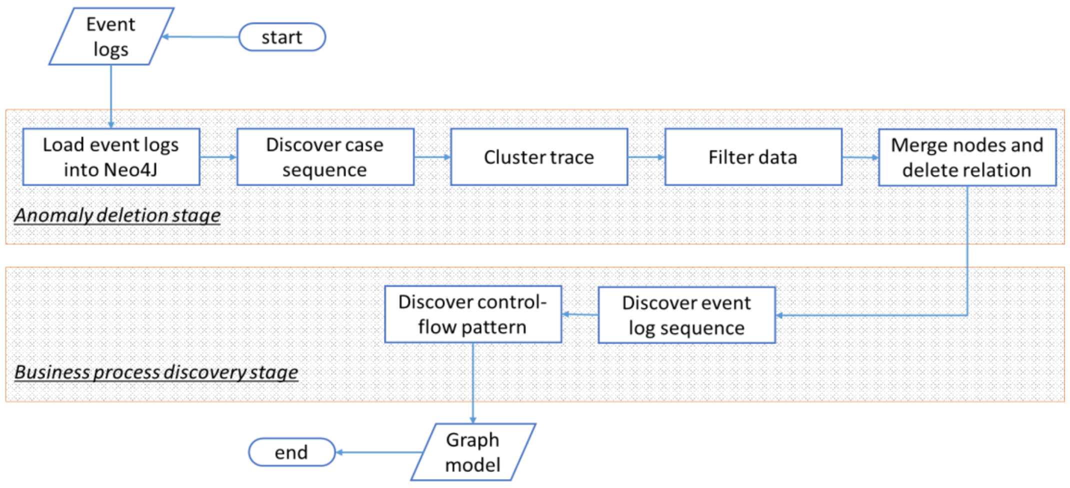

Event logs
start
Load event logs into Neo4J
Discover case sequence
Cluster trace
Filter data
Merge nodes and delete relation
Anomaly deletion stage
Discover control-flow pattern
Discover event log sequence
Business process discovery stage
end
Graph model

Figure 1. From event logs to generated anomalous-free business process model in the form of a graph model

2.1. Anomaly Deletion Stage

This stage is dedicated to removing all possible business process anomalies in the event log. First, a graph model is constructed from the event logs by loading the event logs into NEO4J. There are two functions to load the event logs, i.e., create and merge functions. The create function is used since it creates nodes based on all existing data in the event logs. The event logs are loaded into two labels, i.e., Activity and CaseActivity. Activity label contains all columns inside of event logs while CaseActivity label only contains CaseId and activity name. The Activity label is used for data filtering step, while CaseActivity label is used for case sequence discovery and trace clustering steps. Figure 2 shows the query structure of loading event logs into Neo4J.

```
LOAD CSV with headers FROM "file:///*.csv" AS line
create (:Activity {CaseId:line.Case_ID, Name:line.Activity, StartTime:line.Start_Timestamp, EndTime:line.End_Timestamp })
LOAD CSV with headers FROM "file:///*.csv" AS line
create (:CaseActivity { CaseId:line.Case_ID, Name:line.Activity })
```

Figure 2. Load event log into neo4j using create function query

Having loaded the event logs into nodes, they are connected by case ID and the order of activity in the associated event logs instead of attached by case. Therefore, in the second step, the case sequence is discovered. If a node is the successor of the other node, a NEXT label is put on the relation between them. Figure 3 depicts the case sequence construction query.

```
//CASE SEQUENCE DISCOVERY
match (c:Activity) with collect(c) AS Caselist unwind range(0,Size(Caselist) - 2) AS array with
Caselist[array] AS list1, Caselist[array+1] AS list2
match (b:CaseActivity),(a:CaseActivity) where list1.CaseId = list2.CaseId AND list1.Name = a.Name
AND list2.Name = b.Name AND list1.CaseId = a.CaseId AND list2.CaseId = b.CaseId
merge (a)-[:NEXT {relation:"NEXT"}]->(b)
```

Figure 3. Case sequence discovery query

After connecting the nodes, the next step is performing trace clustering. The purpose of trace clustering is to group node sequences having the same trace and calculate its occurrences within each cluster. The query of trace clustering in Neo4J is shown in Figure 4.

```
//TRACE CLUSTERING
match(a:Activity) with a.CaseId AS id, collect (a.Name) AS Trace_Type, count (a.Name) AS
Trace_Lenght
match(b:CaseActivity) where id = b.CaseId
return count (distinct b.CaseId) AS Frequencies, Trace_Type, Collect (distinct b.CaseId) AS CaseId,
Trace_Lenght
order by Frequencies desc
```

Figure 4. Trace clustering query

After traces are clustered, data filtering is conducted. The objective of this step is to remove the anomaly data. In this case, it is the two lowest frequencies of occurrence of the trace. Because the Neo4J cannot save the result of a query, automatic deletion cannot be done using Neo4J. To overcome this problem data deletion is performed by reading the trace length in each case ID and then the two lowest frequencies are deleted. The query removes the data based on the results of the trace clustering by changing Frequencies variable on where clause as depicted in Figure 5.

```
//DELETING DATA BASED ON TRACE LENGTH
match (m:CaseActivity)
where m.CaseId as id, COLLECT (m.Name) as Trace_Type, count (m.Name) as Frequencies
where Frequencies<=6
match (n:CaseActivity{CaseId:id})
detach delete n
```

Figure 5. Data filtering cipher query

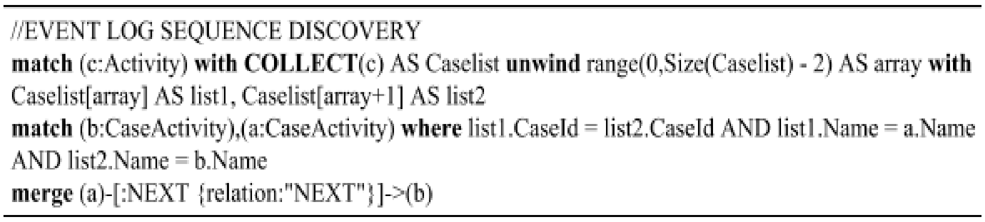
To obtain a correct business process model in the next stage, all nodes having the same activity name must be merged, and the existing case sequence relations must be removed. After both processes as mentioned above, it follows with deleting Case ID inside CaseActivity label since it leads to producing wrong event log sequence relations. Figure 6 displays a Neo4J query to conduct the merging and deletion processes.

```
//NODE MERGING
match (n:CaseActivity) with n.Name AS name, collect(n) AS nodelist, count(*) AS count
where count > 1 call apoc.refactor.merge Nodes(nodelist) YIELD node return node
//DELETE SEQUENCE RELATION
match ()-[n:NEXT]->() detach delete n
//REMOVE CaseId PROPERTIES
match (n:CaseActivity) remove n.CaseId return n
```

Figure 6. Node merging and relation delete query

2.2. Business Process Discovery Stage

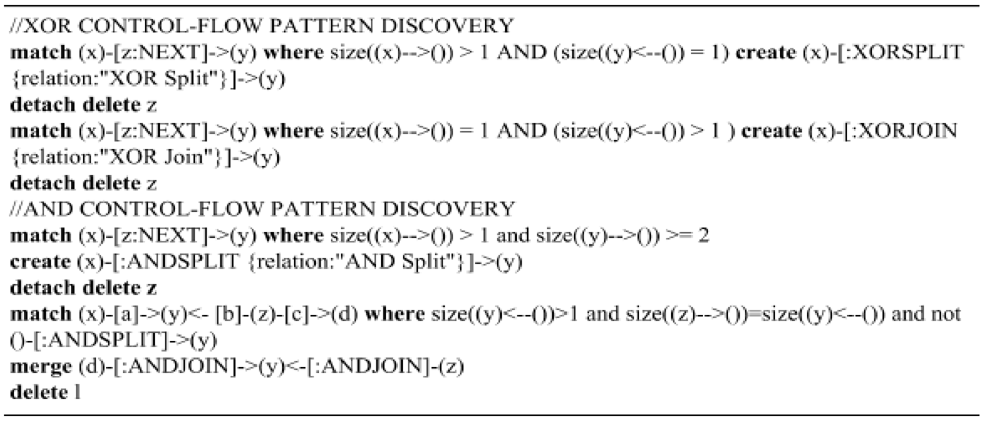
The second stage of the proposed method is discovering the business process as the continuation of the previous stage. First, an event log sequence needs to be created by connecting all nodes based on the order of activities. The Neo4J query of event log sequence discovery presented in Figure 7 is slightly similar to the one used for case sequence discovery used in the previous stage.



```
//EVENT LOG SEQUENCE DISCOVERY
match (c:Activity) with COLLECT(c) AS Caselist unwind range(0,Size(Caselist) - 2) AS array with
Caselist[array] AS list1, Caselist[array+1] AS list2
match (b:CaseActivity),(a:CaseActivity) where list1.CaseId = list2.CaseId AND list1.Name = a.Name
AND list2.Name = b.Name
merge (a)-[:NEXT {relation:"NEXT"}]->(b)
```

Figure 7. Event log discovery sequence query

The second step of this stage is discovering the control-flow pattern. It consists of AND, XOR, OR SPLIT, and OR JOIN relations. The incoming and outgoing edges determine these relations. Figure 8 shows the added AND and XOR connection queries along with the existing OR relation queries. This step produces the final graph model.



```
//XOR CONTROL-FLOW PATTERN DISCOVERY
match (x)-[z:NEXT]->(y) where size((x)-->()) > 1 AND (size((y)<--()) = 1) create (x)-[:XORSPLIT
{relation:"XOR Split"}]->(y)
detach delete z
match (x)-[z:NEXT]->(y) where size((x)-->()) = 1 AND (size((y)<--()) > 1 ) create (x)-[:XORJOIN
{relation:"XOR Join"}]->(y)
detach delete z
//AND CONTROL-FLOW PATTERN DISCOVERY
match (x)-[z:NEXT]->(y) where size((x)-->()) > 1 and size((y)-->()) >= 2
create (x)-[:ANDSPLIT {relation:"AND Split"}]->(y)
detach delete z
match (x)-[a]->(y)<- [b]-(z)-[c]->(d) where size((y)<--())>1 and size((z)-->())=size((y)<--()) and not
()-[:ANDSPLIT]->(y)
merge (d)-[:ANDJOIN]->(y)<-[:ANDJOIN]-(z)
delete l
```

Figure 8. Control-flow pattern discovery query

3. Results and Analysis

The event log must contain Case_ID as a process instance, activities name, start timestamp and end timestamp. Table 1 describes the content of event logs of interlibrary loan application used in this research. In total, it contains 74 cases and 1293 activities which are divided into 31 types of activities.

Table 1. The Fragment of Interlibrary Loan Event Log

Case_ID	Activity	Start_Timestamp	End_Timestamp
PP1	Request material	02-05-17 07:05	02-05-17 07:10
PP1	Check availability	02-05-17 07:10	02-05-17 10:15
PP1	Confirm material availability	02-05-17 10:15	02-05-17 10:20
PP1	Send material	02-05-17 10:20	02-05-17 14:15
PP1	Confirm material arrival	02-05-17 14:15	02-05-17 14:20
PP1	Take material	02-05-17 14:20	03-05-17 14:35
PP1	Use material	03-05-17 14:35	15-05-17 07:15
PP1	Return material before due time	15-05-17 07:15	15-05-17 07:30
PP1	Send back material	15-05-17 07:30	15-05-17 10:07
PP1	Confirm material send back the arrival	15-05-17 10:07	15-05-17 10:12
PP1	Check material condition	15-05-17 10:12	15-05-17 13:12
PP1	Confirm material in good condition	15-05-17 13:12	15-05-17 13:17
PP1	Confirm loan finish	15-05-17 13:17	15-05-17 3:25

Figure 9 depicts the implementation result of load data and case sequence discovery. This research limits the case sequence discovery on two Case IDs, i.e., PP1 and PP2. The result of trace clustering which is the is the frequency of occurrences of traces is shown in Table 2. In Figure 9, the names of activities are aliases of the real names.

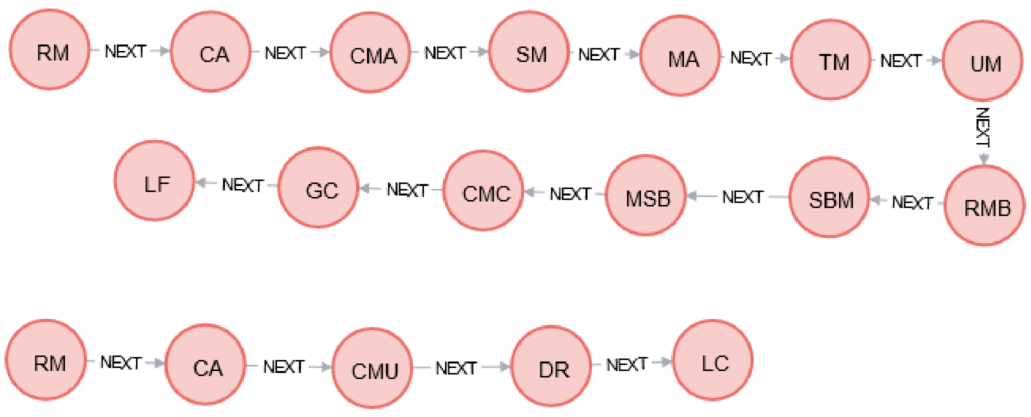


Figure 9. Data loading and case sequence discovery limited on PP1 and PP2 case IDs

Table 2. Trace Clustering Implementation using Neo4J

Frequencies	Trace_Type	CaseId	Trace_Length
12	RM, CA, CMA, SM, MA, TM, UM, UM, SBM, MSB, CMC, GC, LF	PP34, PP16, PP1, PP48, PP15, PP18, PP68, PP50, PP47, PP20, PP67, PP52	13
8	RM, CA, CMA, SM, MA, TM, UM, RMA, SBM, MSB, CMC, GC, DF, SFD, APF, PF, PAY, LF	PP49, PP58, PP7, PP19, PP39, PP51, PP26, PP17	18
6	RM, CA, CMA, SM, MA, TM, UM, RLE, LEA, ALE, UM, RLE, LEA, DLE, UM, SBM, MSB, CMC, GC, LF	PP25, PP57, PP69, PP6, PP38, PP70	20
6	RM, CA, CMA, SM, MA, TM, UM, RLE, LEA, DLE, UM, SBM, MSB, CMC, BC, DF, SFD, APF, PF, PAY, LF	PP31, PP13, PP45, PP12, PP44, PP32	21
5	RM, CA, CMA, SM, MA, TM, UM, RLE, LEA, ALE, UM, RMA, SBM, MSB, CMC, GC, DF, SFD, APF, PF, PAY, LF	PP10, PP63, PP66, PP42, PP29	22
5	RM, CA, CMA, SM, MA, TM, UM, RLE, LEA, ALE, UM, RLE, LEA, ALE, UM, UM, SBM, MSB, CMC, GC, LF	PP46, PP33, PP62, PP65, PP14	21
5	RM, CA, CMA, SM, MA, TM, UM, RLE, LEA, ALE, UM, UM, SBM, MSB, CMC, BC, DF, SFD, APF, PF, PAY, LF	PP61, PP43, PP64, PP30, PP11	22
4	RM, CA, CMA, SM, MA, EXT, SBM, MSB, LC	PP37, PP24, PP56, PP5	16
4	RM, CA, CMA, SM, MA, TM, UM, RLE, LEA, DLE, UM, SBM, MSB, CMC, GC, LF	PP55, PP4, PP36, PP23	18
4	RM, CA, CMA, SM, MA, TM, UM, RLE, LEA, ALE, UM, UM, SBM, MSB, CMC, GC, LF	PP22, PP3, PP54, PP35	17
4	RM, CA, CMA, SM, MA, TM, UM, RMA, SBM, MSB, CMC, BC, DF, SFD, APF, PF, PAY, LF	PP28, PP9, PP60, PP41	9
4	RM, CA, CMA, SM, MA, TM, UM, UM, SBM, MSB, CMC, BC, DF, SFD, APF, PF, PAY, LF	PP40, PP27, PP59, PP8	18
3	RM, CA, CMU, DR, LC	PP21, PP53, PP2	5
2	RM, PMR, BS, UBL, DR, LC	PP72, PP71	6
2	RM, BS, PMR, UBL, DR, LC	PP73, PP74	6

Mapping of aliases and real names can be seen on Table 3. Table 3 contains activity names as the real names of activities in the event log, and aliases are used to simplify those names of activities.

Based on the results of trace clustering query in Table 2, the two smallest values are two and three which appears in ((PP72, PP71), (PP73, PP74)) and (PP2, PP21, PP53), respectively. By doing Node Merging, the relationship between nodes is stacked by the number of occurrences of the relation. In this paper, the number of NEXT relation between activity

Request material and Check availability activity is 70 relations. Therefore, it is necessary to delete NEXT relation and then redo sequence discovery to form the correct business process model. After deleting the NEXT relation, Case ID on the CaseActivity label must be removed to generate the correct business process model in the event log sequence discovery step.

Table 3. Alias of Interlibrary Loan Event Log Activities

Alias	Activity Name	Alias	Activity Name
RM	Request material	DR	Deny request
CA	Check availability	LC	Confirm loan cancel
CMA	Confirm material availability	RLE	Request lending extension
SM	Send material	LEA	Ask lending extension availability
MA	Confirm material arrival	ALE	Allow lending extension
TM	Take material	DLE	Deny lending extension
UM	Use material	EXT	Requested material pickup exceed the time limit
RMB	Return material before due time	RMA	Return material after due time
SBM	Send back material	DF	Determine the amounts of fines to be paid
MSB	Confirm material send back the arrival	SFD	Send detail of the fines
CMC	Check material condition	APF	Ask the user to pay the fine
GC	Confirm material in good condition	PF	Pay the fines and confirm payment
LF	Confirm loan finish	PAY	Confirm payment
CMU	Confirm material unavailable	BC	Confirm material in a bad condition
BS	Check user blacklist status	PMR	Check user past material request
UBL	A user is blacklisted, or user has not returned material after the time limit		

The result of event log sequence discovery is shown in Figure 10. The event log sequence discovery connects entire nodes to an event log in the order of events. The relation created at this stage describes the overall activities from start to finish and branching activity if there are branches in the event log. In Figure 10, the activities that only exist on PP2, PP21, and PP53 are Deny Request and Confirm Unavailable Material Not Found. This indicates that anomaly data deletion has been successfully performed. The nodes formed in Figure 11 are 26 nodes, in contrast to before the deletion of anomalous data the number of nodes is 31.

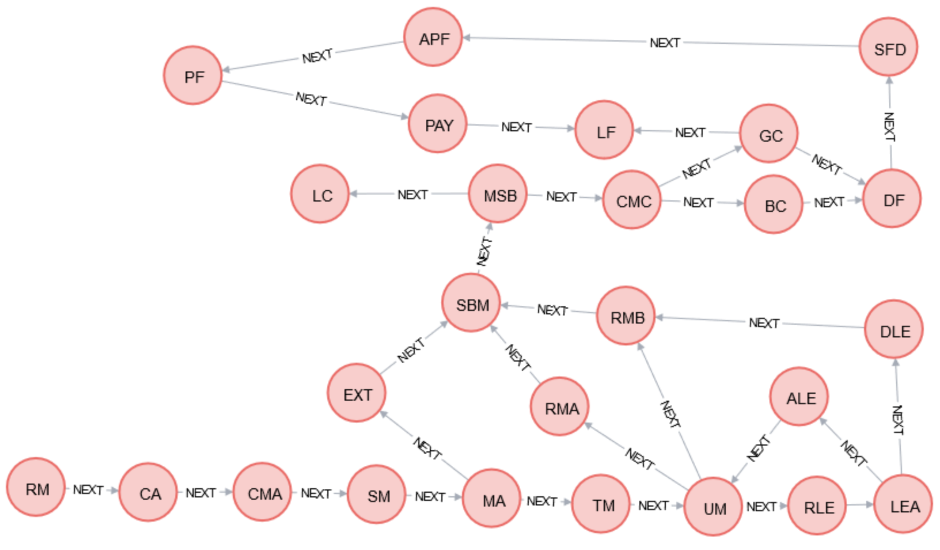


Figure 1. Filtered graph model with sequence relation

The result of discovering control-flow patterns is shown in Figure 11. In the filtered graph model of an interlibrary loan, only XOR pattern exists. Figure 11 is a modified Figure 10. Some sequence relations is deleted and replaced by XORSPLIT and XORJOIN as the results of control-flow pattern discovery.

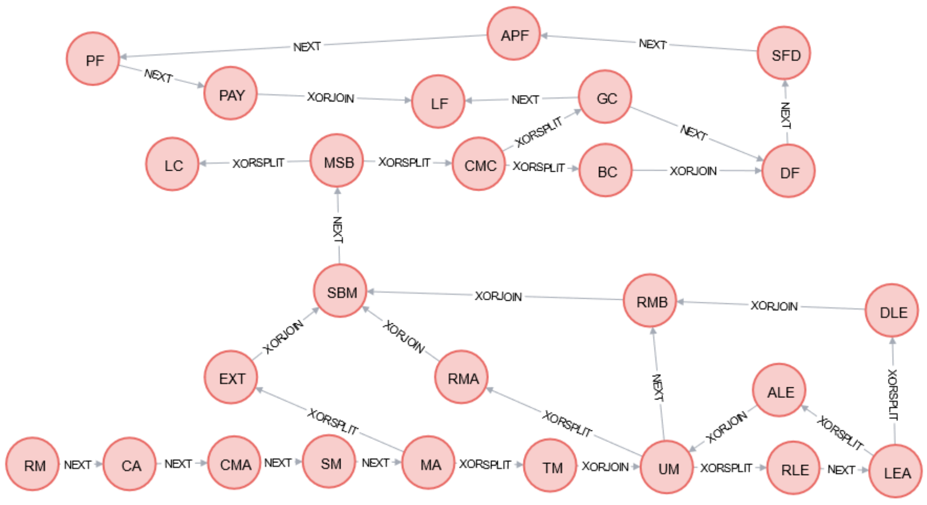


Figure 2. Filtered graph model with control-flow pattern

The result of modeling the event log without data filtering is shown in Figure 12. There are five more nodes or activities in Figure 12 compared to Figure 11. Some of the deleted nodes contain ANDSPLIT and ANDJOIN relation, and some nodes contained XORSPLIT and XORJOIN.

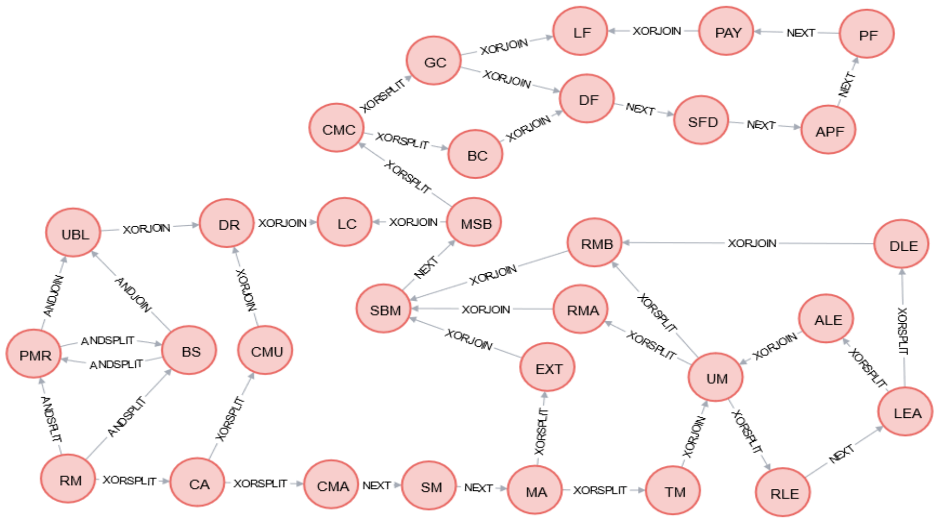


Figure 3. Unfiltered graph model with a control-flow pattern

After the graph model has been formed, the last step is to compare both graph fitness and precision. Fitness and precision are one of the measurement aspects to measure the quality of the business process model. Fitness measures how many processes that can be depicted in the model. Meanwhile, precision estimates how many traces in the event log can be depicted in the model. In (1), the number of captured cases in the model is stored in variable $n\ (Captured\_Cases)$, while the number of cases in the event log is stored in variable $n(Cases\_in\_Event\_Log)$. In (2), the number of captured traces in the model is stored in variable $n(Captured_{Traces})$, while the number of traces in the event log is stored in variable $n(Traces\_in\_Event\_Log)$.

$$Fitness(x) = {}^{n(Captured\_Cases)}\Big/_{n(Cases\_in\_Event\_Log)} \qquad (1)$$

$$Precision(x) = {n(Captured\_Traces)}\big/{n(Traces\_in\_Event\_Log)} \quad (2)$$

Both filtered and unfiltered model have the same fitness values of 1. But the precision value of the filtered model is higher than the precision value of the unfiltered model. Precision value of the filtered model is 0.81. Meanwhile, the precision value of the unfiltered model is 0.64.

4. Conclusion

This research proposed a methodology to discover a better business process model by conducting trace clustering. The event log is imported and modeled to form a business process model. There are several traces captured in the model. Some of the traces appeared with a low-frequency value. The low-frequency value is considered as anomalous. Then, data filtering is done to remove the anomalous traces. After that, process discovery is made to form a new business process model. The result found improvement of the precision value of the proposed method. Therefore, trace clustering and data filtering can form a better business process model.

References

[1] R Sarno, KR Sungkono. Coupled Hidden Markov Model for Process Mining of Invisible Prime Tasks. *International Review Computers and Software (IRECOS)*. 2016; 11(6): 539-547.

[2] NY Setiawan, R Sarno. Multi-criteria Decision Making for Selecting Semantic Web Service Considering Variability and Complexity Trade-off. *Journal of Theoretical and Applied Information Technology*. 2016; 86(2): 316-326.

[3] Mekhala. A Review Paper on Process Mining. *International Journal of Engineering and Techniques*. 2015; 1(4): 11-17.

[4] WMP van der Aalst, T Weijters, L Maruster. Workflow Mining: Discovering Process Models from Event Logs. *IEEE Transactions on Knowledge and Data Engineering*. 2004; 16(9): 1128-1142.

[5] AJMM Weijters, WMP van der Aalst, AAK Medeiros. Process Mining with the Heuristics Miner-Algorithm. Eindhoven: Eindhoven University of Technology. 2006.

[6] R Sarno, F Haryadita, D Sunaryono, A Munif. *Model Discovery of Parallel Business Processes using Modified Heuristic Miner*. International Conference on Science in Information Technology (ICSITech). Yogyakarta. 2015: 30-35.

[7] Hermawan, R Sarno. A More Efficient Deterministic Algorithm for Process Model Discovery. *International Journal of Innovative Computing, Information, and Control (IJICIC)*. 2018; 14(3): 971-995.

[8] R Sarno, KR Sungkono. Coupled Hidden Markov Model for Process Mining of Non-free Choice and Invisible Prime Tasks. *Procedia Computer Science*. 2018; 124: 134-141.

[9] R Sarno, KR Sungkono. Hidden Markov Model for Process Mining of Parallel Business Processes. *International Review on Computer and Software (IRECOS)*. 2016; 11(4): 290-300.

[10] AAK Medeiros, WMP van der Aalst, AJMM Weijters. *Workflow Mining: Current Status and Future Directions*. On The Move to Meaningful Internet Systems 2003: CoopIS, DOA, and ODBASE. OTM 2003. Catania. 2003: 389-406.

[11] AAK Medeiros, BF van Dongen, WMP van der Aalst, AJMM Weijters. Process Mining: Extending the Alpha-algorithm to Mine Short Loops. Technische Universiteit Eindhoven. Eindhoven. 2004.

[12] L Wen, WMP van der Aalst, J Wang, J Sun. Mining Process Models with Non-Free-Choice Constructs. *Data Mining and Knowledge Discovery*. 2007; 15(2): 145-180.

[13] L Wen, WMP van der Aalst, B Huang, J Sun. Mining Process Models with Prime Invisible Tasks. *Data & Knowledge Engineering*. 2010; 69(10): 999-1021.

[14] Q Guo, L Wen, J Wang, Z Yan, PS Yu. *Mining Invisible Tasks in Non-free-choice Constructs*. Business Process Management. BPM 2016. Rio de Janeiro. 2015: 109-125.

[15] I Robinson, J Webber, E Eifrem. Graph Database: New Opportunities for Connected Data. Amerika Serikat: O'Reilly Media, Inc. 2015.

[16] R Sarno, KR Sungkono, R Septiarakhman. Graph-Based Approach for Modeling and Matching Parallel Business Processes. *International Information Institute*. 2018; 21(5): 1603-1614.

[17] I Robinson, J Webber, E Eifrem. Graph Databases: New Opportunities for Connected Data, 2nd ed., O'Reilly Media, Inc. 2015.

[18] N Francis, A Taylor, A Green, P Guagliardo, L Libkin, T Lindaaker, V Marsault, S Plantikow, M Rydberg, P Selmer. *Cypher: An Evolving Query Language for Property Graphs*. SIGMOD '18 Proceedings of the 2018 International Conference on Management of Data. Houston. 2018: 1433-1445.