TELKOMNIKA, Vol.13, No.2, June 2015, pp. 125~132
ISSN: 1693-6930, accredited **A** by DIKTI, Decree No: 58/DIKTI/Kep/2013
DOI: 10.12928/TELKOMNIKA.v13i2.xxxx

Graph-based Algorithm for Checking Wrong Indirect Relationships of Process Model Containing Non-Free Choice

Agung Wiratmo, Kelly Rossa Sungkono, Riyanarto Sarno
Department of Informatics, Faculty of Information Technology and Communication
Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia
e-mail: riyanarto@if.its.ac.id

Abstract

Detecting wrong indirect relationship in the non-free choice is a challenge in process mining. The existing decision mining without parameter can only detect the direction, but not the correctness. Because of that, this paper proposes a combination of parameterized decision mining and relation sequences to detect the direction and correctness. Firstly, discovering a graph process model based on the event log. Then, analyzing the graph process model to obtain decision points. After that, processing each decision point by parameterized decision mining, so there is a formation of decision rules. The derived decision rules are used as parameters of checking wrong indirect relationship in the non-free choice. The evaluation shows that the checking wrong indirect relationships in non-free choice with parameterized decision mining have 100% accuracy, whereas the existing decision mining has 90.7% accuracy.

Keywords*: Decision mining, Parameterize decision mining, Process model, Wrong indirect relationships*

1. Introduction

Each company records events carried out in the event log. Information from the event log can be analyzed to obtain knowledge [1]. The obtaining process is called the process mining, which aims to find out, monitor and improve the occurring processes [2]. In the process mining, there are two most prominent processes, namely: 1) conformance checking and 2) discovery process [2]. The process mining can found the wrong processes in the event log.

This paper discovers a graph process model based on the event log. Firstly, discovering the graph process to obtain decision points. Then, processing each decision point by parameterized decision mining, so that there is a formation of decision rules. The derived decision rules are used as parameters of checking wrong indirect relationship in the non-free choice.

Several previous studies discuss decision mining in recent years. Rozinat [3] introduced decision mining on business processes. Horita [4] improved the decision mining by interpreting that in linear temporal logic. Unfortunately, both of them had not been implemented as a decision rule for checking errors in event logs.

The existing methods in checking indirect relationships [5,6] in non-free choice only used direction so that the error can be detected only from directional error, not correctness. The proposed method, namely parameterized decision mining, uses the decision rule in checking event logs with considering not only from the direction but also from the parameters in the event log. In this research, the decision rule is used to find errors in the event log so the result is more accurate in terms of the direction and correctness of the choice.

2. Research Method

In this section, parameterized decision mining will be presented to find a wrong indirect relationship in non-free choice using the graph database. The non-free choice is a condition not free to make choices which depend on the results of the previous election [7]. Checking process models on the non-free choice part of the event log can be done correctly must consider all dependencies [8]. There are two types of dependence on the process of the model,

Received February 23, 2014; Revised May 29, 2014; Accepted June 12, 2014

direct dependence or referred to direct relationship and indirect dependence or referred to as indirect relationship [5,9].

Direct relationship is a relationship or dependency that is directly between tasks. Conversely, indirect relationship is a relationship or dependence that is indirectly between tasks [8]. A graph database is a NoSQL database that is represented in the form of graph [9]. Graph databases form data as nodes and relations between nodes [10,11]. The process mining is used to extract information from the event log to see business processes [8,12]. The process mining can be used to build a process model [18]. Decision mining is used to study parameters that can influence the selection of grooves [13].

Decision mining is used to find rules for branching from each decision point. By using a graph database, a decision point is known from a node that has a xorsplit relation. The used algorithm in decision mining research is the C4.5 decision tree algorithm [3]. The decision tree is used to predict an activity seen from the parameters of a data. In the decision tree, there are several terms such as root as the initial node, leaf node as the child of a node, and the depth of a node is the length of the path between the nodes to the leaf node [14,15].

The first step is to discover the graph process model of the event log based on the graph database. Then, graph process model be analyzed to find the decision point. The second step is discovering decision rule using decision mining from each decision point with notice parameter in the event log. Then, using decision rule as a parameter in determining the wrong decision in non-free choice. The last step searches each case in the event log with the parameters stated earlier.

2.1. Discovery process model based on graph database

Table 1. Input: event log

Case_ID	amount	stackType	Status	Time	Activity
PP10	3	nonreefer	complete	3/11/2016 0:52	A
PP10	3	nonreefer	complete	3/11/2016 2:00	C
PP10	3	nonreefer	complete	3/11/2016 3:08	D
PP10	3	nonreefer	complete	3/11/2016 4:16	E
PP10	3	nonreefer	complete	3/11/2016 5:24	F
PP10	3	nonreefer	complete	3/11/2016 6:32	H
.					
.					
.					
PP412	17	reefer	incomplete	7/2/2016 22:28	A
PP412	17	reefer	incomplete	7/2/2016 23:36	C
PP412	17	reefer	incomplete	7/3/2016 0:44	D
PP412	17	reefer	incomplete	7/3/2016 1:52	E
PP412	17	reefer	incomplete	7/3/2016 3:00	G
PP412	17	reefer	incomplete	7/3/2016 4:08	H
.					
.					
.					
PP735	13	nonreefer	incomplete	10/2/2016 10:52	A
PP735	13	nonreefer	incomplete	10/2/2016 12:00	B
PP735	13	nonreefer	incomplete	10/2/2016 13:08	D
PP735	13	nonreefer	incomplete	10/2/2016 14:16	E
PP735	13	nonreefer	incomplete	10/2/2016 15:24	G
PP735	13	nonreefer	incomplete	10/2/2016 16:32	H
.					
.					
.					
PP1050	10	nonreefer	complete	12/30/2016 16:52	A
PP1050	10	nonreefer	complete	12/30/2016 18:00	B
PP1050	10	nonreefer	complete	12/30/2016 19:08	D
PP1050	10	nonreefer	complete	12/30/2016 20:16	E
PP1050	10	nonreefer	complete	12/30/2016 21:24	F
PP1050	10	nonreefer	complete	12/30/2016 22:32	H

Table 2. Queries in the graph database

No	Queries
1	def importActivityT(tx, fileName): tx.run("LOAD CSV with headers FROM 'file:///"+fileName+"' AS line " "Merge (:Activity {CaseId:line.Case_ID, Name:line.Activity, Amount:toInt(line.amount), StackType:line.stackType, Status:line.status, Time:line.Time})")
2	def importCaseActivity(tx, fileName): tx.run("LOAD CSV with headers FROM 'file:///"+fileName+"' AS line Merge (:CaseActivity {Name:line.Activity })")
3	def createRelationship(tx): # create sequence relation tx.run("MATCH (c:Activity) " "WITH COLLECT(c) AS Caselist " "UNWIND RANGE(0,Size(Caselist) - 2) as idx " "WITH Caselist[idx] AS s1, Caselist[idx+1] AS s2 " "MATCH (b:CaseActivity),(a:CaseActivity) " "WHERE s1.CaseId = s2.CaseId AND s1.Name = a.Name AND s2.Name = b.Name " "MERGE (a)-[r:SEQUENCE]->(b)") # create xorsplit relation tx.run("MATCH (bef)-[r]->(aft) " "WHERE size((bef)-->())>1 AND size((aft)<--())=1 AND (size((aft)-->())=1 OR size((aft)-->())>1) " "CREATE (bef)-[:XORSPLIT]->(aft) " "DELETE r") # create xorjoin relation tx.run("MATCH (bef)-[r]->(aft) " "WHERE (size((bef)-->())=1 OR size((bef)-->())>1) AND size((aft)<--())>1 " "CREATE (bef)-[:XORJOIN]->(aft) " "DELETE r") # create andsplit relation tx.run("MATCH (aft1)<-[r]-(bef)-[s]->(aft2)" "WHERE size((bef)-->())>1 " "AND size((aft2)-->())=size((bef)-->()) AND size((aft1)-->())=size((bef)-->()) " "AND not (aft1)-[:SEQUENCE]->(bef) AND not (aft2)-[:SEQUENCE]->(bef) " "MERGE (aft1)<-[:ANDSPLIT]-(bef)-[:ANDSPLIT]->(aft2) " "DELETE r,s") # create andjoin relation tx.run("MATCH (aft1)-[r]->(bef)<-[s]-(aft2) " "WHERE size((bef)<--())>1 " "AND size((aft2)-->())=size((bef)<--()) AND size((aft1)-->())=size((bef)<--()) " "AND not ()-[:ANDSPLIT]->(bef) " "MERGE (aft1)-[:ANDJOIN]->(bef)<-[:ANDJOIN]-(aft2) " "DELETE r,s") # create Non-Free Choice tx.run("match ()-[c:XORSPLIT]->(n) " "match (a)-[b:XORJOIN]->() " "match (k:Activity),(l:Activity) " "where a.Name<>n.Name and k.Name=a.Name and l.Name=n.Name and k.CaseId=l.CaseId and k.Time<l.Time " "merge (a)-[:NONFREECHOICE]->(n)")

4	def printStartingNodeNonFreeChoice(tx):
	nodes = []
	global nodeStartedNonFreeChoice
	for record in tx.run("MATCH (p)-[r:XORSPLIT]->() RETURN p.Name ORDER BY p.Name"):
	nodes.append(record["p.Name"])
	nodeStartedNonFreeChoice = np.unique(np.array(nodes))
	return nodeStartedNonFreeChoice

The first step for the discovery of the graph model process is to enter event logs like in Table 1 into the graph database a query for (1) Import all data in event log, (2) Import unique activity, (3) Create relation: sequence, xorsplit, xorjoin, andsplit, andjoin, non-free choice, and (4) Get the decision point like in Table 2. The parameters in the event log used in the graph database are Case_ID, Activity, and Time.

In the model process, there will be several relationships [16,17] such as xorsplit, xorjoin, andsplit, andjoin and non-free choice shown in Figure 1. Joint relations are a relation of the union from branching to split relations at the base of branching. The xor relation is a branching relation which means that the flow of the event log can only choose one of the entire branches of the event log. The and relation is a flow relationship that will do all events even though the different order.

The results of working on queries in Table 2 produce the model process shown in Figure 1. After each activity is represented in the node, the next step is to make relations between nodes such as sequence, xorsplit, xorjoin, andjoin, andsplit relations, and non-free choice.

After the graph process model is formed, the next step is to determine the decision point by using the query in Table 2. Decision point is the node where branching of the process begins. Figure 1 shows graph process model where containing the decision points in node A and node E. Node A is the base of member node B or node C. Node E is the base of branching node G and node F. Furthermore, decision rule would be discovered by decision mining to find parameters each branching of each decision point.

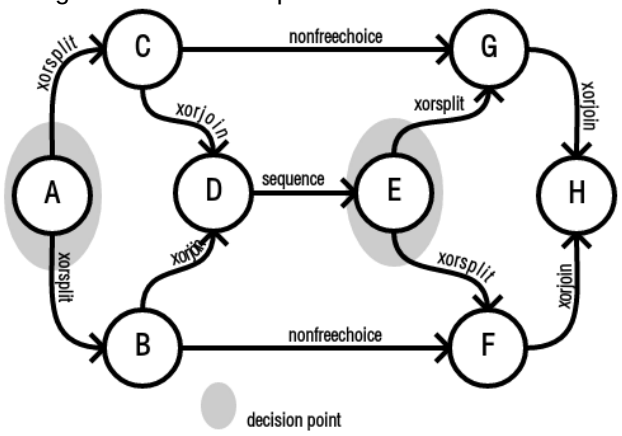

Figure 1. Examples of the process model

2.2. Extracting a parameter in decision points

The decision point shown in Figure 1, there are two decision point points. Each decision point will be analyzed by considering the parameters in the event log to get the decision rule. The process for extraction decision rule is called decision mining. The algorithm of decision mining is used C4.5 decision tree algorithm. The first step is getting leaf nodes from each decision point. Then, the next step is getting an event log that has activities such as leaf nodes with the algorithm in Figure 2.

The data needed in the algorithm in Figure 2 is $decisionPoint$, $leafNode$, and $eventLog$. Each decision point of leaf nodes obtained from algorithm in Figure 2 is used in the decision mining process. The decision mining algorithm is seen in Figure 3.

The algorithm in Figure 3 has five data variables: X is the event log on node leaf. Y is an attribute owned by X. The $splitingAtribute$ is an attribute used as a solving parameter. The $atributeSelectionMethod$ is the method used to find the best fraction value.

The method used to find the best splitting criterion is C4.5 with the Gini index parameter in (1) and (2). The function of (1) is to evaluate the separation in event log each attribute. The function of (2) is to assess the separation in event log each parameter.

Data: $decisionPoint, leafNode, eventLog$
Result: $Event\ log\ pada\ leaf\ node$
$DataLeaf$
for each x of $decisionPoint$ ***do***
  for each y of $leafNode$ of each x ***do***
    for z of $eventLog$ ***do***
      if activity in each z of $trueEventLog$ equals with activity in y $leafNode$ ***then***
        attach a event $Log\ z$ to $DataLeaf$
      end
    end
  end
end

Figure 2. Algorithm for getting event log of leaf nodes each decision point

Data: $X, Y, splitingAtribute, atributeSelectionMethod, splitingCriteria, N$
Result: $Parameter\ decision\ in\ decision\ point\ showed\ by\ decision\ tree$
$Create\ Node\ N$
if tuple in X are all of the same activity ***then***
  return N as a leaf node labeled with activity in X
end
if Y is empty ***then***
  return N as a leaf node labeled with majority activity in X
end
$atributeSelectionMethod(X, Y)$
if ($splitingAtribute$ is discrete − valued) ***then***
  $Y \leftarrow Y - splitingAtribute$
end
for each outcome i of $splitingCriteria$ ***do***
  let Xi be the set of data tuples in $X\ satisfying$ the outcome j
  if Xi is empty ***then***
    attach a leaf labeled with majority activity in X to node N
  else
    $atributeSelectionMethod(Xi, Y)$
  end
end
return ***N***

Figure 3. Algorithm for extracting a parameter each decision point using decision mining

$$Gini(t) = 1 - \sum[p(\frac{j}{t})]^2 \quad (1)$$

where $p(\frac{j}{t})$ represents the frequency of the j attribute in activity t.

$$Gini_{split} = \sum_{i=1}^{k} \frac{n_i}{n} Gini(i) \quad (2)$$

where k is the number of partitions, n_i is the amount of data in i partition, n is the amount of data in the $p$node. The smallest $Gini_{split}$ indicates the best split value. The $splitingCriteria$ is the value of the parameter that is used as a solver. N is a node. The algorithm in Figure 3 will continue to be repeated until the data is X empty. The results of algorithm in Figure 3 are a decision tree by showing the parameters and the leaf values can be seen in Figure 4.

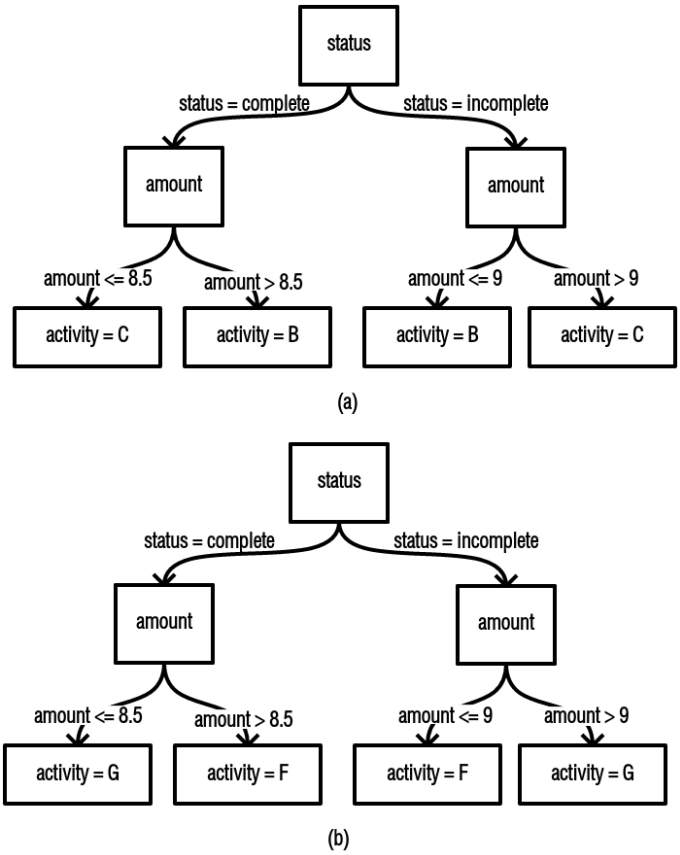

Figure 4. Decision tree in (a)node A and (b) node E

From Figure 4 (a), it can be concluded that the condition for doing activity B is ((status = complete AND amount > 8.5) OR (status = incomplete AND amount ≤ 9)). Whereas to do activity C, it must be conditioned ((status = complete AND amount ≤ 8.5) OR (status = incomplete AND amount > 9)). From Figure 4 (b) it can be concluded that the condition for doing activity F is ((status = complete AND amount > 8.5) OR (status = incomplete AND amount ≤ 9)). Whereas to do activity G, it must be conditioned ((status = complete AND amount ≤ 8.5) OR (status = incomplete AND amount t> 9)). The next step is to check the event log with parameters that have been obtained previously.

2.3. Checking wrong indirect relationships

After finding these parameters, then looking for a wrong indirect relationship in non-free choice. Each case is checking because it has different parameters. The goal is to find faults with precession far better than just using the direction of each case. The goal is to find faults with precession far better than just using the direction of each case — checking scheme for non-free choices containing indirect relations as in algorithm in Figure 5. The algorithm in Figure 5 needs several parameters like $decisionParameter$ and $EventLog$. The process will repeat as many cases as in $EventLog$ and checking in case i with $decisionParameter$. If the process is not the same with the rule, then the case i will be added in $IndirectRelationship$.

Figure 5. Algorithm for checking indirect relationship in non-free choice

Data: $decisionParameter, EventLog$
Result: $IndirectRelationship$
$IndirectRelationship$
for each case i of $EventLog$ ***do***
  if case i parameter not equal with $decisionParameter$ ***then***
    attach a case i to $node\ IndirectRelationship$
  end
end

3. Results and Analysis

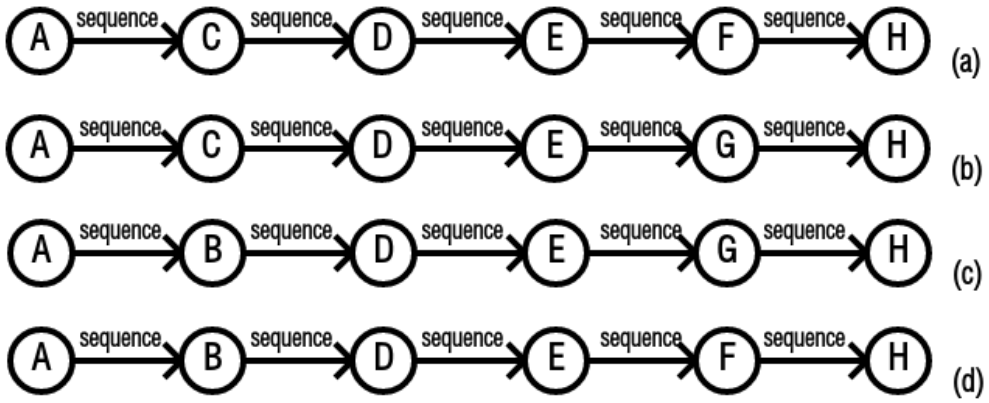

Figure 6. The result of the process which contains wrong indirect relationship in Case_ID: (a) PP10, (b) PP412, (c) PP735, and (d) PP1050

The proposed method implements 1199 cases in the event log. The event log has many variations of the attribute in parameters like amount, stackType, status, and times as in Table 1. From the 1199 case activity, the results of checking by using the proposed method are depicted in

Figure 6.

From

Figure 6 (a) having the sequence of events A → C → D → E → F → H is the sequence of events that are wrong based on the parameters and based on the order of non-free choice. Case_ID PP10 has several parameters, which value of parameter status is complete and in parameter amount has three.

Figure 6 (a) shows after activity A goes to activity C is a correct but wrong decision after activity E goes to activity F.

Figure 6 (b) has the order of events A → C → D → E → G → H is the sequence of events incorrectly based on parameters due to incomplete and amount seven status parameters. In decision parameters obtained from decision mining requires that it can pass activity C then conditions fulfilled ((status = complete AND amount 8.5) OR (status = incomplete AND amount> 9)) and conditions for passing activity G must meet the requirements ((status = complete AND amount 8.5) OR (status = incomplete AND amount> 9)). However, when viewed based on the order non-free choice is correct.

Figure 6 (c) has the sequence of events A → B → D → E → G → H is the sequence of events that are wrong based on the parameters and based on the order of non-free choice. Case_ID PP735 has several parameters, which value of parameter status is incomplete and and the parameter amount has thirteen.

Figure 6 (c) shows after activity A goes to activity B is a correct but wrong decision after activity E goes to activity G.

Figure 6 (d) has the order of events A → B → D → E → F → H is the sequence of events wrong based on parameters because of complete status parameters and amount 3. In the decision parameters obtained from decision mining requires that it can pass activity B then the conditions fulfilled ((status = complete AND amount> 8.5) OR (status = incomplete AND amount 9)) and the condition for passing activity F must meet the requirements ((status = complete AND amount> 8.5) OR (status = incomplete AND amount 9)) . However, when viewed based on the order non-free choice is correct.

Accuracy of the existing method shows 900 case activity in TP, 188 case activity in TN, 111 case activity in FP dan zero activity in FN.

$$Accuracy = \frac{900 + 188}{900 + 188 + 111 + 0}\ x\ 100\%$$

$$Accuracy = \ 0.907\ x\ 100\%$$

$$Accuracy = \ 90.7\%$$

The accuracy of the parameterize decision mining shows 900 case activity in TP, 299 case activity in TN, zero case activity in FP dan zero activity in FN.

$$Accuracy = \frac{900 + 299}{900 + 299 + 0 + 0}\ x\ 100\%$$

$$Accuracy = \ 1\ x\ 100\%$$

$$Accuracy = \ 100\%$$

4. Conclusion

The parameterized decision mining is to overcome the correct flow in a direction but not in a parameterize direction. The parameterized decision mining considers parameters in the selection of grooves.

Discovering a graph process model is based on the event log. Then, the graph process model is analyzed for obtaining decision points. Each decision point is processed by using parameterized decision mining, so that decision rules are formed. The obtained decision rules are used as parameters of checking wrong indirect relationship in the non-free choice.

The accuracy of the parameterized decision mining reaches 100%, which means that the proposed method can detect errors far more precisely than the existing method only get 90.7% accuracy, which means it can only detect errors that use direction but cannot detect parameter errors.

References

[1] Sarno R, Sungkono KR. Coupled Hidden Markov Model for Process Discovery of Non-Free Choice and Invisible Prime Tasks. Procedia Computer Science [Internet]. 2018;124:134–41. Available from: 10.1016/j.procs.2017.12.139

[2] der Aalst WMP. Decomposing Petri nets for process mining: A generic approach. Distributed and Parallel Databases. 2013;31(4):471–507.

[3] Rozinat A, van der Aalst WMP. Decision mining in ProM. In: International Conference on Business Process Management. 2006. p. 420–5.

[4] Horita H, Hirayama H, Hayase T, Tahara Y, Ohsuga A. Process Mining Approach Based on Partial Structures of Event Logs and Decision Tree Learning. In: 2016 5th IIAI International Congress on Advanced Applied Informatics (IIAI-AAI). 2016. p. 113–8.

[5] Wen L, Wang J, Sun J. Detecting implicit dependencies between tasks from event logs. In: Asia-Pacific Web Conference. 2006. p. 591–603.

[6] Van der Aalst WM., de Medeiros AKA. Process mining and security: Detecting anomalous process executions and checking process conformance. Electronic Notes in Theoretical Computer Science. 2005;121:3–21.

[7] Sungkono KR, Sarno R. Constructing Control-Flow Patterns Containing Invisible Task and Non-Free Choice Based on Declarative Model. International Journal of Innovative Computing, Information and Control (IJICIC). 2018;14(4).

[8] Wen L, van der Aalst WMP, Wang J, Sun J. Mining process models with non-free-choice constructs. Data Mining and Knowledge Discovery. 2007;15(2):145–80.

[9] Kalynychenko O, Chalyi S, Bodyanskiy Y, Golian V, Golian N. Implementation of search mechanism for implicit dependences in process mining. In: 2013 IEEE 7th International Conference on Intelligent Data Acquisition and Advanced Computing Systems (IDAACS). 2013. p. 138–42.

[10] Sarno R, Sungkono KR, Johanes R, Sunaryono D. Graph-Based Algorithms for Discovering A Process Model Containing Invisible Tasks. Intelligent Networks and

Systems Society (INASS).
[11] Iwasaki K, Kuriyama Y, Kondoh S, Shirayori A. Structuring engineers: implicit knowledge of forming process design by using a graph model. Procedia Cirp. 2018;67:563–8.
[12] Chapela-Campa D, Mucientes M, Lama M. Mining Frequent Patterns in Process Models. 2017 Oct 11 [cited 2019 Apr 19]; Available from: http://arxiv.org/abs/1710.05693
[13] Sarno R, Sari PLI, Sunaryono D, Amaliah B, Mukhlash I. Mining decision to discover the relation of rules among decision points in a non-free choice construct. In: Proceedings of International Conference on Information, Communication Technology and System (ICTS) 2014. 2014. p. 53–8.
[14] Bombara G, Vasile C, Penedo F, Yasuoka H, Belta C. A decision tree approach to data classification using signal temporal logic. In: Proceedings of the 19th International Conference on Hybrid Systems: Computation and Control. ACM; 2016. p. 1–10.
[15] Saettler A, Laber E, Pereira F de AM. Decision tree classification with bounded number of errors. Information Processing Letters. 2017;27–31.
[16] Sarno R, Effendi YA. Hierarchy Process Mining from Multi-source Logs. TELKOMNIKA. 2017;15(4):1960–75.
[17] Sarno R, Effendi YA, Haryadita F. Modified Time-Based Heuristics Miner for Parallel Business Processes. International Review on Computers and Software (IRECOS) [Internet]. 2016;11(3):249–60. Available from: https://doi.org/10.15866/irecos.v11i3.8717