Journal of King Saud University - Computer and Information Sciences

Enhancing Model Quality and Scalability for Mining Business Processes with Invisible Tasks in Non-Free Choice

--Manuscript Draft--

Manuscript Number:	JKSUCIS-D-23-01194R2
Article Type:	Full Length Article
Keywords:	Business process management; Graph Database; invisible tasks; Process Mining; process modelling
Corresponding Author:	Riyanarto Sarno, Ph.D. Institut Teknologi Sepuluh Nopember Surabaya, INDONESIA
First Author:	Kelly Rossa Sungkono, magister
Order of Authors:	Kelly Rossa Sungkono, magister
	Riyanarto Sarno
	Bhakti Stephan Onggo
	Muhammad Farhan Haykal
Abstract:	At present, business processes are growing rapidly, resulting in various types of activity relationships and big event logs. Discovering invisible tasks and invisible tasks in non-free choice is challenging. alpha$ mines invisible prime tasks in non-free choice based on pairs of events, so it consumes considerable processing time. In addition, the invisible tasks formation by alpha$ is limited to skip, switch, and redo conditions. This study proposes a graph-based algorithm named Graph Advanced Invisible Task in Non-free choice (GAITN) to form invisible tasks in non-free choice for stacked branching relationships condition and handle large event logs. GAITN partitions the event log and creates rules for merging the partitions to scale up the volume of discoverable events. Then, GAITN utilises rules of previous graph-based process mining algorithm to visualises branching relationships (XOR, OR, AND) and creates rules of mining invisible tasks in non-free choice based on obtained branching relationships. This study compared the performance of GAITN with that of Graph Invisible Task (GIT), alpha$, and Fodina and found that GAITN produces process models with better fitness, precision, generalisation, and simplicity measure based on higher number of events. GAITN significantly improves the quality of process model and scalability of process mining algorithm.
Suggested Reviewers:	Hyerim Bae Professor, Pusan National University hrbae@pusan.ac.kr the research is about process mining, which is related with the topic of the proposed article
	Corallo Angelo Professor, University of Salento angelo.corallo@unisalento.it His research about implementation process modelling, so it is suitable with the topic of our article
	Daniel Amyot Professor, University of Ottawa School of Electrical Engineering and Computer Science damyot@uottawa.ca His research interest is business process modelling, which is suitable with the topic of our research
Response to Reviewers:	Reviewer #1 : Congratulations. This is a very good paper. Comment #1: Thank you very much for your review and your compliment.

	Reviewer #3 : -. Comment #3: Thank you very much for your time to review my paper. Reviewer #4 : The raised issues in a previous round of reviewing have been addressed. However, there are still some key points that need further clarifications, and the clarity of presentation could be enhanced by clarifying these issues and attending to some reasoning issues. Specifically, the authors need to address the clarity of expressions. 1.The expression "Business processes are currently growing rapidly causing various types of activity relationships to emerge" needs clarifications. 2."but only a few algorithms have capability to discover invisible tasks and invisible tasks in non-free choice constructs." It is still unknown how the related work differs from theirs with the drawbacks of existing ones. Comment #4 : Thank you very much for your review. 1.I changed the expression with two new sentences (the first sentence and the second sentence of the second paragraph). I argue that the emergence of various business process relationships is due to the diversity of user requirements to achieve business objectives, and mining those business process relationships is a process discovery algorithm challenge. 2.I added a list of process discovery algorithms that can form invisible tasks and invisible tasks in free choice to emphasize the algorithm referred to in the sentence.

[Prof. Riyanarto Sarno
Institut Teknologi Sepuluh Nopember
Jl. Raya ITS, Keputih,
Kecamatan Sukolilo
Surabaya
Jawa Timur
Indonesia 60111]

[Professor Nasser-Eddine Rikli
Editors-in-Chief
Journal of King Saud University - Computer and Information Sciences]

[June 11, 2023]

Dear Professor Nasser-Eddine Rikli,

I am pleased to submit an original research article entitled [“Enhancing Model Quality and Scalability for Mining Business Processes with Invisible Tasks in Non-Free Choice” by Kelly Rossa Sungkono, Riyanarto Sarno, Bhakti Stephan Onggo, and Muhammad Farhan Haykal] for consideration for publication in ***Journal of King Saud University - Computer and Information Sciences***. We propose a graph-based process mining method to handle large event logs and discover advanced relationships, i.e., invisible tasks of stacked branching relationship in non-free choice. The proposed graph-based method is an improvement of our previous graph-based process mining algorithm named GIT (https://doi.org/10.1186/s40537-021-00487-x)

$\alpha^{\$}$, the sophisticated α-based algorithm, discovers business process containing invisible prime tasks in non-free choice. The drawback of $\alpha^{\$}$ is (1) $\alpha^{\$}$ mines invisible prime tasks in non-free choice based on pairs of events, so it consumes considerable processing time, and (2) the invisible tasks formation by $\alpha^{\$}$ is limited to skip, switch, and redo conditions. Based on our experiment data, we found that invisible tasks are needed for stacked branching relationships. GIT algorithm (https://doi.org/10.1186/s40537-021-00487-x) is proven to produce less complexity time than $\alpha^{\$}$; however GIT cannot handle large data of the event log at once due to the data limit of the graph database and cannot mine invisible tasks of invisible task of stacked branching relationships. We propose new graph-based algorithm to extends GIT rules for handling large event logs and mining invisible task of stacked branching relationships. First, the proposed graph-based algorithm partitions the event log and creates rules for merging the partitions to scale up the volume of discoverable events. The proposed graph-based algorithm visualises branching relationships in graph model and creates rules of mining invisible tasks in non-free choice based on obtained branching relationships. We compared our proposed graph-based algorithm with GIT, $\alpha^{\$}$ and Fodina and found our proposed graph-based algorithm produces better fitness, precision, generalisation, and simplicity measure with higher events than those three algorithms.

This manuscript has not been published and is not under consideration for publication elsewhere. We have no conflicts of interest to disclose.

Thank you for your consideration.

Sincerely,
[Prof. Riyanarto Sarno, Ph.D.
Professor in Institut Teknologi Sepuluh Nopember]

Response to Reviewer Comments

Paper ID	JKSUCIS-D-23-01194
Paper Title	Enhancing Model Quality and Scalability for Mining Business Processes with Invisible Tasks in Non-Free Choice

Reviewer #1 : Congratulations. This is a very good paper.
Comment #1: *Thank you very much for your review and your compliment.*

Reviewer #3 : -.
Comment #3: *Thank you very much for your time to review my paper.*

Reviewer #4 :
The raised issues in a previous round of reviewing have been addressed. However, there are still some key points that need further clarifications, and the clarity of presentation could be enhanced by clarifying these issues and attending to some reasoning issues. Specifically, the authors need to address the clarity of expressions.

1. The expression "Business processes are currently growing rapidly causing various types of activity relationships to emerge" needs clarifications.
2. "but only a few algorithms have capability to discover invisible tasks and invisible tasks in non-free choice constructs." It is still unknown how the related work differs from theirs with the drawbacks of existing ones.

Comment #4 :
Thank you very much for your review.

1. *I changed the expression with two new sentences (the first sentence and the second sentence of the second paragraph). I argue that the emergence of various business process relationships is due to the diversity of user requirements to achieve business objectives, and mining those business process relationships is a process discovery algorithm challenge.*
2. *I added a list of process discovery algorithms that can form invisible tasks and invisible tasks in free choice to emphasize the algorithm referred to in the sentence.*

Enhancing Model Quality and Scalability for Mining Business Processes with Invisible Tasks in Non-Free Choice

Kelly R. Sungkono[a], Riyanarto Sarno[a], Bhakti S. Onggo[b], Muhammad F. Haykal[a]

[a]Department of Informatics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia

[b]CORMSIS, University of Southampton, Southampton, UK

Abstract

At present, business processes are growing rapidly, resulting in various types of activity relationships and big event logs. Discovering invisible tasks and invisible tasks in non-free choice is challenging. $\alpha^{\$}$ mines invisible prime tasks in non-free choice based on pairs of events, so it consumes considerable processing time. In addition, the invisible tasks formation by $\alpha^{\$}$ is limited to skip, switch, and redo conditions. This study proposes a graph-based algorithm named Graph Advanced Invisible Task in Non-free choice (GAITN) to form invisible tasks in non-free choice for stacked branching relationships condition and handle large event logs. GAITN partitions the event log and creates rules for merging the partitions to scale up the volume of discoverable events. Then, GAITN utilises rules of previous graph-based process mining algorithm to visualises branching relationships (XOR, OR, AND) and creates rules of mining invisible tasks in non-free choice based on obtained branching relationships. This study compared the performance of GAITN with that of Graph Invisible Task (GIT), $\alpha^{\$}$, and Fodina and found that GAITN produces process models with better fitness, precision, generalisation, and simplicity measure based on higher number of events. GAITN significantly improves the quality of process model and scalability of process mining algorithm.

Keywords: business process management; graph database; invisible tasks; process mining; process modelling

Click here to view linked References

Enhancing Model Quality and Scalability for Mining Business Processes with Invisible Tasks in Non-Free Choice

Abstract

At present, business processes are growing rapidly, resulting in various types of activity relationships and big event logs. Discovering invisible tasks and invisible tasks in non-free choice is challenging. $\alpha^\$$ mines invisible prime tasks in non-free choice based on pairs of events, so it consumes considerable processing time. In addition, the invisible tasks formation by $\alpha^\$$ is limited to skip, switch, and redo conditions. This study proposes a graph-based algorithm named Graph Advanced Invisible Task in Non-free choice (GAITN) to form invisible tasks in non-free choice for stacked branching relationships condition and handle large event logs. GAITN partitions the event log and creates rules for merging the partitions to scale up the volume of discoverable events. Then, GAITN utilises rules of previous graph-based process mining algorithm to visualises branching relationships (XOR, OR, AND) and creates rules of mining invisible tasks in non-free choice based on obtained branching relationships. This study compared the performance of GAITN with that of Graph Invisible Task (GIT), $\alpha^\$$, and Fodina and found that GAITN produces process models with better fitness, precision, generalisation, and simplicity measure based on higher number of events. GAITN significantly improves the quality of process model and scalability of process mining algorithm.

Keywords: business process management; graph database; invisible tasks; process mining; process modelling

1. Introduction

In recent years, the availability of event logs for analysing business processes in many domains, such as healthcare (De Roock & Martin, 2022; Pika et al., 2020) and manufacturing (Choueiri et al., 2020; Choueiri & Portela Santos, 2021), has increased significantly. Process mining is a study that uses event logs to gain insight into real-life processes, identify process-related issues, and improve process performance (van der Aalst, 2011). Process mining consists of three aspects: process discovery, conformance checking, and enhancement. Process discovery is a process mining study (Erdogan & Tarhan, 2018; Hamdani & Abdelli, 2020; Kim et al., 2022; van der Aalst, 2016) that automatically visualises the flow of activities recorded in an event log (Beeson et al., 2002; Berger et al., 2022).

There are many kinds of business process relationships to describe the diversity of user requirements for achieving business objectives. Therefore, the challenge of the process discovery algorithm is to mine various business process relationships in order to visualise the right model of the running business process. Various studies have proposed process discovery techniques to form variations in business process relationships. The pioneering process discovery algorithm, α (Back et al., 2020; van der Aalst, 2016) implemented several rules based on footprints of activity pairs to form sequence, XOR, and AND in a Petri net-based process model. α algorithm has several variants, such as α^{++} (Wen et al., 2007; Zheng et al., 2019) to discover non-free choice constructs (NFC), $\alpha^{\#}$ algorithm (Wen et al., 2010) to mine invisible tasks, and $\alpha^{\$}$ algorithm (Guo et al., 2015) to form invisible tasks in non-free choice constructs. The inductive miner (Battineni et al., 2020; Pika et al., 2020) and the refined process structure tree (RPST) (Yan et al., 2019) divide the event log into smaller sub-logs and implement rules to define activity relationships based on the pairs of activities. Inductive miner (Battineni et al., 2020; Pika et al., 2020), refined process structure tree (Anugrah et al., 2015) and β (Zayoud et al., 2019) can only discover sequence, XOR, and AND relationships. Furthermore, Heuristic Miner (Kurniati et al., 2016; Namaki et al., 2022; Weber et al., 2018) can only discover sequences, XOR, and AND relationships, while Fodina (vanden Broucke & De Weerdt, 2017) improves the rules of Heuristic Miner to discover invisible tasks. Graph-based algorithms have undergone various developments to discover activity relationships of a process model, such as sequence relationships, branching (XOR, OR, AND) relationships (Waspada et al., 2020), invisible tasks (Sarno et al., 2019) and invisible tasks in non-free choice (Sarno et al., 2021). Most process discovery algorithms can mine sequence and branching relationships, but only a few algorithms, which are $\alpha^{\#}$ algorithm (Wen et al., 2010), $\alpha^{\$}$ algorithm (Guo et al., 2015), a graph-based algorithm (Sarno et al., 2019), GIT (Sarno et al.,

2021) and Fodina (vanden Broucke & De Weerdt, 2017), can form invisible tasks and/or invisible tasks in non-free choice constructs in the process model.

Creating invisible tasks to delineate specific process flows is challenging because those tasks are not stored in the event log. Wen (Wen et al., 2010), the inventor of $\alpha^{\#}$ algorithm, states that invisible tasks are formed to visualise skip, switch, and redo conditions. These invisible tasks are called invisible prime tasks. Other algorithms, $\alpha^{\$}$ algorithm (Guo et al., 2015), GIT (Sarno et al., 2021) and Fodina (vanden Broucke & De Weerdt, 2017) mine invisible tasks with reference to the rules of $\alpha^{\#}$ algorithm.

In addition to the branching relationship, some processes need stacked branching relationships to model their conditions. Invisible tasks are added to clarify the position of branching relationships when they are stacked (can be seen in the actual process model of Fig 1). This study also presents bakery productions processes and a solid hazardous waste management process that need invisible tasks to model stacked branching relationships in their process models (detailed explanation in Section 2.3). The ability of a process discovery algorithm to discover invisible tasks of stacked branching relationships is important, especially when the result is used for further analysis (e.g., conformance analysis) or modelling (e.g., simulation).

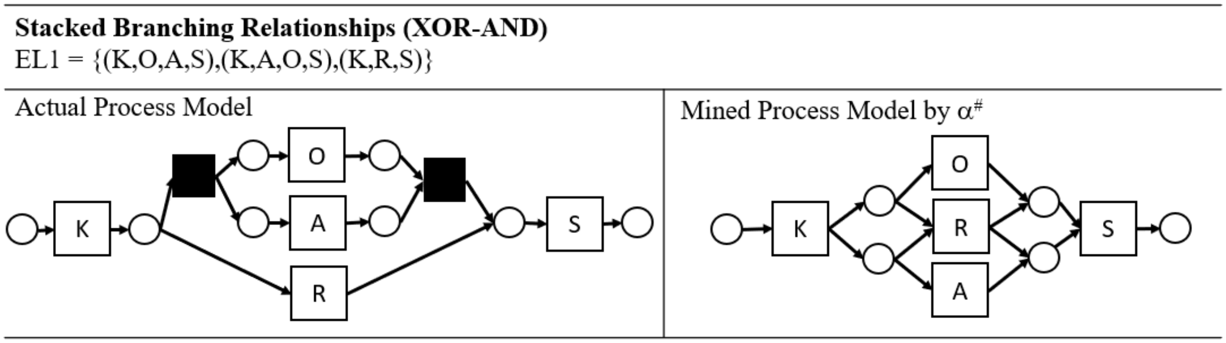

Stacked Branching Relationships (XOR-AND)
EL1 = {(K,O,A,S),(K,A,O,S),(K,R,S)}
Actual Process Model
K
O
A
R
S
Mined Process Model by α#
K
O
R
A
S

Fig 1. The problem encountered when mining stacked branching relationships by $\alpha^{\#}$ algorithm

$\alpha^{\#}$ algorithm can mine one type of stacked branching relationships (an AND relationship is followed by a XOR relationship) without define an invisible task, so $\alpha^{\#}$ algorithm does not create rules to mine invisible tasks for stacked branching relationships. However, this study detects a problem if the stack of branching relationships is switched (a XOR relationship is followed by an AND relationship) by using $\alpha^{\#}$ algorithm. In Fig 1, although a mined process model of $\alpha^{\#}$ based on an event log EL1 is sound, R cannot be parallelised with A and O. The process model based on EL1 shows that R and A or R and O can be carried out in one process flow (case), whereas they are not allowed based on the actual model. $\alpha^{\$}$ algorithm (Guo et al., 2015) extends rules of $\alpha^{\#}$ algorithm to mine invisible prime tasks in non-free choice (IT-SBR-NFC). Nevertheless, $\alpha^{\$}$ has not considered IT-SBR-NFC because it adopts the invisible task mining rules of the $\alpha^{\#}$ algorithm.

In addition to accurately detecting relational variations, the computing time of process discovery is an important concern. Existing algorithms which are α (van der Aalst, 2016) and its developments, Inductive miner (Battineni et al., 2020), refined process structure tree (RPST) (Yan et al., 2019), Heuristic Miner (Namaki et al., 2022), and Fodina (vanden Broucke & De Weerdt, 2017) create rules of each type of relationships based on pairs of activities. Creating rules of each type of relationships produces high time complexity to mine advanced relationships, including invisible tasks and invisible tasks in non-free choice construct. Graph-based algorithms (Sarno et al., 2019, 2021; Waspada et al., 2020) store activities and obtained basic relationships in the form of graphs and mines advanced relationships based on obtained basic relationships to reduce time complexity. Another advantage of graph-based algorithm is that the event log and the process model are formed on one platform (graph database), so there is no conversion process.

The current graph-based algorithms have several drawbacks. First, those algorithms could not identify invisible tasks of stacked branching relationships (IT-SBR) and invisible tasks in non-free choice for stacked branching relationships (IT-SBR-NFC) because their rules to constructing invisible tasks adopt rules of $\alpha^{\#}$ algorithm. Moreover, those graph-based algorithms are not able to handle large data from the event log at once due to the data limit of the graph database. Based on this background, we have several motivations to perform this study as follows:

1. This paper presents invisible tasks are needed to form stacked branching relationships condition (a XOR relationship is followed by AND/OR relationships) in the process model. A few process

discovery algorithms (Guo et al., 2015; Sarno et al., 2021; vanden Broucke & De Weerdt, 2017; Wen et al., 2010) can mine invisible tasks and $\alpha^{\#}$ algorithm (Wen et al., 2010) is the pioneer algorithm of discovering invisible tasks. Unfortunately, $\alpha^{\#}$ algorithm (Wen et al., 2010) cannot form invisible tasks of stacked branching relationships. Therefore, the appropriate method for mining invisible tasks of stacked branching relationships (IT-SBR) is needed to produce better process models and enhance the process model quality.

2. Another advanced activity relationship is invisible tasks in non-free choice for stacked branching relationships (IT-SBR-NFC), i.e., a condition when IT-SBR meets non-free choice constructs. $\alpha^{\$}$ algorithm (Guo et al., 2015), the first algorithm that introduces invisible tasks in non-free choice, cannot mine IT-SBR-NFC because $\alpha^{\$}$ algorithm adopts rules of discover invisible tasks by $\alpha^{\#}$ algorithm. Inability to form IT-SBR-NFC can decrease the quality of process model.

3. Existing algorithms (Battineni et al., 2020; Guo et al., 2015; van der Aalst, 2016; vanden Broucke & De Weerdt, 2017; Wen et al., 2007, 2010; Yan et al., 2019) create rules of every type of activity relationships, so they produce high time complexity to mine advanced activity relationships, including IT-SBR and IT-SBR-NFC. Graph-based algorithms (Sarno et al., 2019, 2021; Waspada et al., 2020) reduce time complexity by utilising obtained basic relationships to form advanced relationships. However, the data limit of graph database causes the graph-based algorithms unable to model processes from a massive event log. Data pre-processing is needed to overcome the inadequacies of graph-based algorithms.

According to those motivations, we propose a new graph-based algorithm—Graph Advanced Invisible Task in Non-free choice (GAITN)—to discover IT-SBR and IT-SBR-NFC from an event log. Specifically, we extend Graph Invisible Task (GIT) (Sarno et al., 2021) by adding new rules to identify IT-SBR and IT-SBR-NFC. To handle a large volume of data, GAITN does data pre-processing by partitioning event logs into event log snippets. This is because the computing time required to process multiple smaller event log snippets is shorter than that required to process a single large event log. The use of partitions requires us to modify the rule that discovers a sequence relationship, such that the rule can discover a sequence relationship from activities stored in multiple event log snippets. Identifying IT-SBR and IT-SBR-NFC improves the quality of the process model, whereas partitioning event logs aims to increase scalability.

We conducted two experiments to evaluate the GAITN. The objective of the first experiment is to compare the quality of the process model discovered by GAITN, GIT (Sarno et al., 2021), $\alpha^{\$}$ (Guo et al., 2015) and Fodina (vanden Broucke & De Weerdt, 2017) using fitness, precision, simplicity, and generalisation. For this experiment, we will use two synthetic event logs of processes containing IT-SBR-NFC, an event log generated by the simulation model of a medical waste management system, and an event log of medical records (Sarno et al., 2021). The objective of the second experiment is to compare the computing times and number of events handled by those algorithms. For this experiment, we use two synthetic event logs: two event logs from medical records (Sarno et al., 2021), and three real-life event logs: BPIC 2011 Hospital (B. F. van Dongen, 2012), Domestic Declarations (B. van Dongen, 2020), BPI Challenge 2012 (B. van Dongen, 2012a).

The remainder of this paper is organized as follows. Section 2 describes the preliminaries of this research, such as the definition of the event log, graph query language, IT-SBR-NFC, types of constructs discovered by previous process discovery algorithms, and quality measures. A detailed explanation of the proposed GAITN algorithm is presented in Section 3. The materials, experimental results, and discussion are presented in Section 4. Finally, we conclude the study in Section 5.

2. Preliminaries

2.1. Event Log, Traces, and Cases

The main input of PDAs is the event log. The event log is a collection of traces. A trace is a collection of unique cases. A case is formed by a sequence of executed activities. For example, an event log $L3 = \{(GA, GB, GC, GD),\ (GA, GB, GC, GD)\}$ contains two cases, each with four activities. However, these two cases were not unique. Therefore, *L3* contained only one trace.

2.2. Graph Query Language (Cypher)

Our proposed algorithm, GAITN, uses Cypher Query Language (Francis et al., 2018; Saad et al., 2023; Šestak & Turkanovic, 2023) to implement discovery rules and construct a process model. Cypher Query Language is a declarative graph query language that can be used to create nodes and their relationships. In a GAITN, nodes represent activities and relationships denote links between activities in a process model. Cypher Query Language has several notations: rounded brackets denote nodes **(nodes)** and square brackets represent the relationship **[:relation]**. Every node contains information, and every piece of information is stored in a node as its attribute. An attribute contains two types of data: a label and a description. For example, Fig 2 shows a representation of the process model at the top of Cypher Query Language. To represent the GA activity in the process model, we used Cypher Query Language syntax **(:Activity: {name:"GA"})**. For the sequence relationship in the process model, the equivalent Cypher Query Language syntax is **–[:relationship]→.** The other Cypher syntaxes used by GAITN are listed in Table 1.

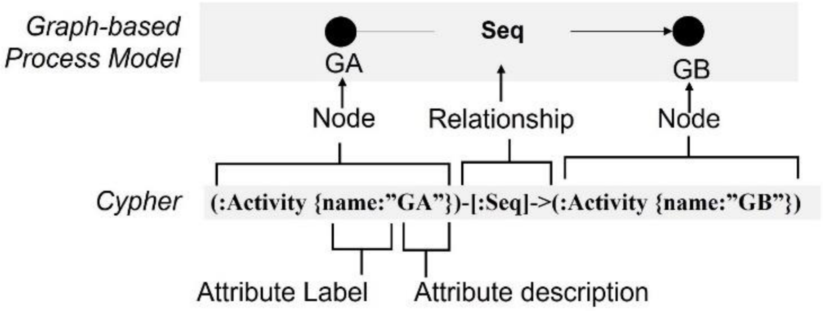
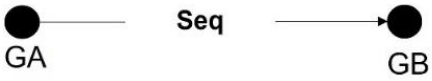
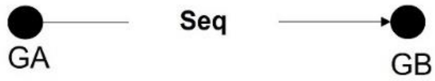


Fig 2. Graph Model and Cypher Query Language

Table 1. Cypher Syntaxes

Syntaxes	Description	Examples
MERGE	Depict nodes and/or relationships and merge nodes with same names	**Objective:** Construct an activity GA that has a sequence relationship with GB, where GA and GB are the first and last activities, respectively **Cypher**: MERGE (:activity {name:"GB"}) – [:Seq] → (:activity {name:"GA"}) **Before Process:** Empty graph database **After Process:** 

DELETE	Remove nodes and/or relationships	**Objective:** Delete all nodes and their relationship **Cypher:** DELETE p=()–[]–() **Before Process:**  **After Process:** Empty graph database

2.3 Invisible Tasks in Non-Free Choice for Stacked Branching Relationships (IT-SBR-NFC)

This paper adds the stacked branching relationships condition as a new condition that needs invisible tasks. The first example of a business process having stacked branching relationships is the production processes of a bakery. The bakery gets an order to factory (OTF) from the distributor and checks the OTF data. If the data are correct, the OTF will be forwarded to the Production Planning and Inventory Control (PPIC) Department and the Finished Goods Warehouse (FGW). However, if the data is incorrect, the bakery will return the OTF request to the distributor. The production process has stacked branching relationships after activity Checking OTF because there is a choice relationship (XOR) between activity returning OTF to Distributor and a parallel relationship (AND) of passing OTF to PPIC and passing OTF to FGW.

Stacked branching relationships also occurred in the solid hazardous waste management processes. The solid waste is sorted into seven types: recyclable waste, pathological waste, infectious waste, sharp waste, pharmaceutical waste, cytotoxic waste, and radioactive waste. The activity of putting waste in the bin has choice relationships (XOR relationships) based on the type of waste. However, there are solid wastes classified as pathological infectious waste, so the activity of put pathological waste and put infectious waste has multi-choice relationship (OR relationship). The process of putting waste portrays stacking XOR-OR relationships, so the process model of solid hazardous waste management has invisible tasks of stacking branching relationships (denoted by gray boxes in Fig 5). Furthermore, there is non-free choice constructs between put waste and shed waste in the solid waste management process, so there is a condition that the invisible tasks meet non-free choice constructs. The solid waste management process needs invisible tasks in non-free choice constructs for stacked branching relationship condition (IT-SBR-NFC).

Fig 3 illustrates two examples of IT-SBR-NFC in a real-world process. The first flow pattern, the IT-SBR represents a combination of a choice relationship (XOR) and a multi-choice relationship (OR) or parallel relationship (AND). In a process model, this is achieved by adding a dummy activity called invisible tasks (e.g., gray nodes in Fig 3). The second flow pattern, the non-free choice (NFC) construct, represents a situation in which a choice depends on a previous choice (Guo et al., 2015; Sarno et al., 2021; Zheng et al., 2019); for example, the choices are connected by dotted lines in Fig 3. Hence, the IT-SBR-NFC occurs when the invisible tasks of stacked branching relationships (IT-SBR) meet the non-free choice (NFC) construct in the process model.

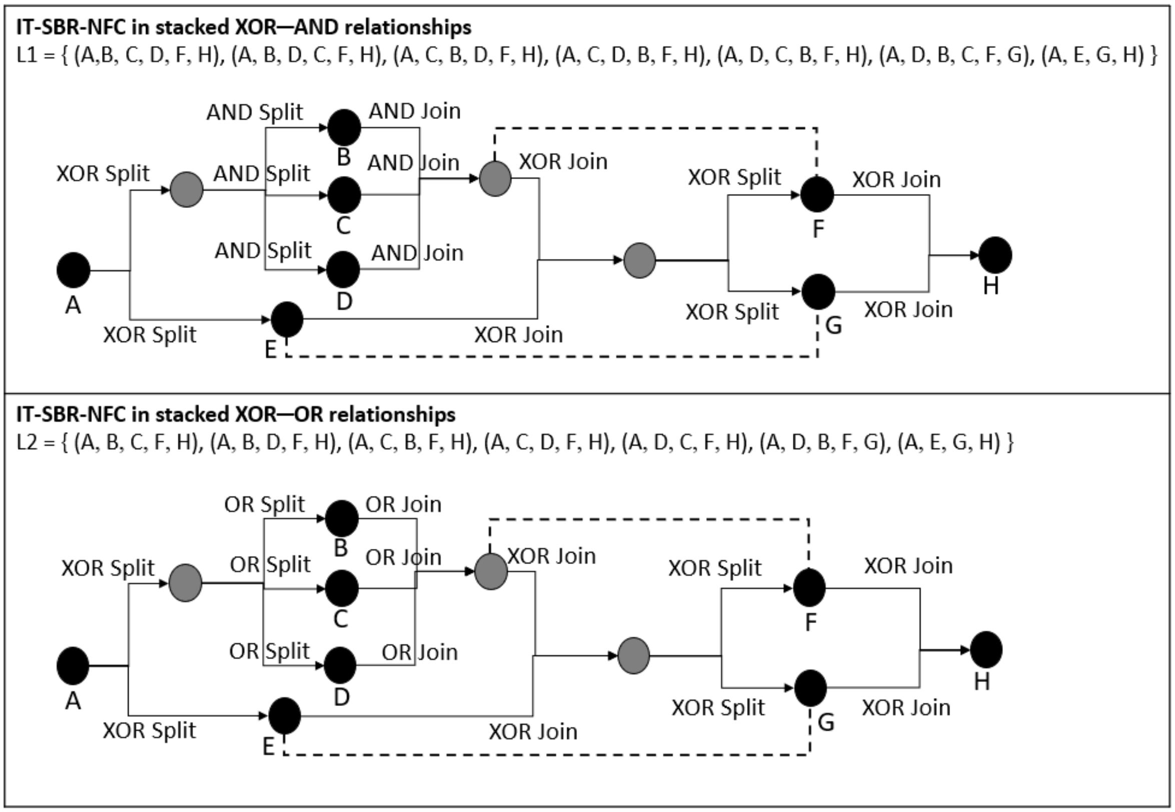

IT-SBR-NFC in stacked XOR—AND relationships
L1 = { (A,B, C, D, F, H), (A, B, D, C, F, H), (A, C, B, D, F, H), (A, C, D, B, F, H), (A, D, C, B, F, H), (A, D, B, C, F, G), (A, E, G, H) }
AND Split
AND Join
XOR Split
AND Split
B
AND Join
XOR Join
XOR Split
XOR Join
C
F
AND Split
AND Join
A
D
H
XOR Split
E
XOR Join
XOR Split
G
XOR Join
IT-SBR-NFC in stacked XOR—OR relationships
L2 = { (A, B, C, F, H), (A, B, D, F, H), (A, C, B, F, H), (A, C, D, F, H), (A, D, C, F, H), (A, D, B, F, G), (A, E, G, H) }
OR Split
OR Join
XOR Split
OR Split
B
OR Join
XOR Join
XOR Split
XOR Join
C
F
OR Split
OR Join
A
D
H
XOR Split
E
XOR Join
XOR Split
G
XOR Join

Fig 3. Invisible Tasks in Non-Free Choice for Stacked Branching Relationships

1.3 GIT Algorithm

GIT (Sarno et al., 2021) defines eight relationships: $\xrightarrow{seq}$, $\xrightarrow{xorsplit}$, $\xrightarrow{xorjoin}$, $\xrightarrow{andsplit}$, $\xrightarrow{andjoin}$, $\xrightarrow{orsplit}$, $\xrightarrow{orjoin}$, $\xrightarrow{NFC}$, and depicts an additional task: Invisible Prime Task (IPT). $\xrightarrow{seq}$ expresses two activities that can be executed consecutively. $\xrightarrow{xorsplit}$ and $\xrightarrow{xorjoin}$ symbolize the XOR relationship, whereas $\xrightarrow{andsplit}$ and $\xrightarrow{andjoin}$

denote the AND relationship. $\xrightarrow{orsplit}$ and $\xrightarrow{orjoin}$ represent OR relationships. The NFC relationship is denoted as $\xrightarrow{NFC}$. Definition 1 is formal description of GIT algorithm and Table 2 shows the graph-based process models depicted by the GIT.

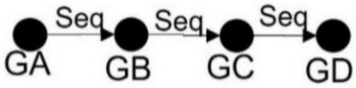
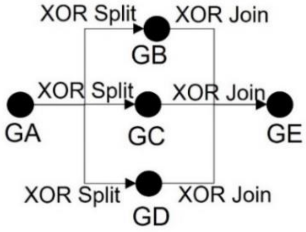
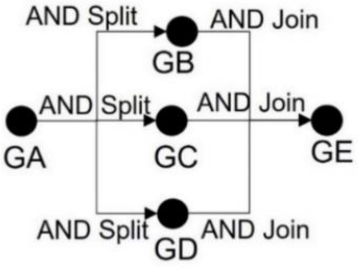
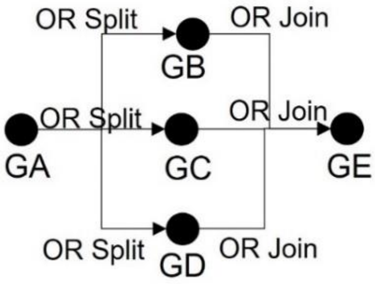
Definition 1. *(Relationships defined in GIT algorithm) Let T be a set of activities $(t_1, t_2, \dots, t_n)$, L be an event log over T, $k, l,$ and y are three tasks depicted in a graph model, the relationships defined in GIT algorithm are defined as follows:*

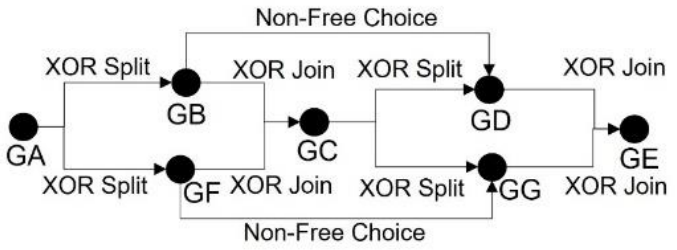
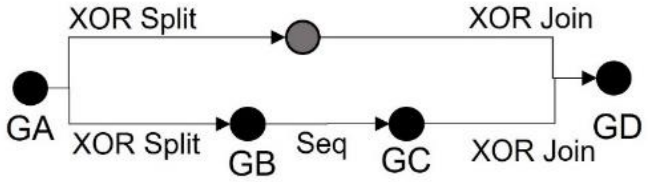
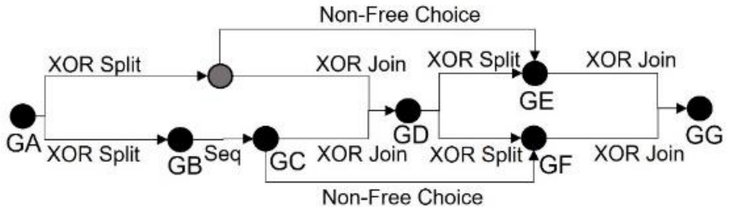
- $k \xrightarrow{seq} l \Leftrightarrow t_1, t_2, \dots, t_n \in L, i \in 1, \dots, n: t_i = k \wedge t_{i+1} = l \wedge t_{i_{CaseID}} = t_{i+1_{CaseID}}$
- $outgoing(k) \Leftrightarrow k \xrightarrow{seq} t_i$
- $incoming(k) \Leftrightarrow t_i \xrightarrow{seq} k$
- $k \xrightarrow{xorsplit} l \Leftrightarrow (n_{outgoing(k)} > 1) \wedge (n_{incoming(l)} = 1) \wedge (n_{outgoing(l)} = 1)$
- $l \xrightarrow{xorjoin} k \Leftrightarrow (n_{incoming(k)} > 1) \wedge (n_{outgoing(l)} = 1) \wedge (n_{incoming(l)} = 1)$
- $k \xrightarrow{andsplit} l \Leftrightarrow (n_{outgoing(k)} > 1) \wedge (n_{incoming(l)} = n_{outgoing(k)}) \wedge (\neg(l \xrightarrow{seq} k))$
- $l \xrightarrow{andjoin} k \Leftrightarrow (n_{ingoing(k)} > 1) \wedge (n_{outgoing(l)} = n_{incoming(k)}) \wedge (\neg(k \xrightarrow{seq} l))$
- $k \xrightarrow{orsplit} l \Leftrightarrow (n_{outgoing(k)} > 1) \wedge (n_{incoming(l)} > 1) \wedge (n_{incoming(l)} < n_{outgoing(k)}) \wedge$

  $(\neg(l \xrightarrow{seq} k))$
- $l \xrightarrow{orjoin} k \Leftrightarrow (n_{incoming(k)} > 1) \wedge (n_{outgoing(l)} > 1) \wedge (n_{outgoing(l)} < n_{incoming(k)}) \wedge$

  $(\neg(k \xrightarrow{seq} l))$
- $k \xrightarrow{xorsplit} IPT \xrightarrow{xorjoin} l \Leftrightarrow \left(k \xrightarrow{xorjoin} l\right) \wedge \left(k \xrightarrow{xorspit} \neg l\right)$
- $l \xrightarrow{NFC} y \Leftrightarrow \left(l \xrightarrow{xorjoin} k\right) \wedge \left(k \xrightarrow{xorsplit} y\right) \wedge (CaseID_l = CaseID_y)$
- $l \xrightarrow{NFC} IPT \Leftrightarrow \left(l \xrightarrow{xorjoin} k\right) \wedge \left(k \xrightarrow{xorsplit} IPT\right) \wedge \left(IPT \xrightarrow{xorjoin} y\right) \wedge (CaseID_l = CaseID_y)$

Table 2. Relationships of Activities

Name of Relationship	**Traces of an Event Log / Process Models of GIT**
Sequence	Trace: (GA, GB, GC, GD) Process Model: 
XOR	Traces: (GA, GB, GE), (GA, GC, GE), (GA, GD, GE) Process Model:  XOR Split XOR Join GB XOR Split XOR Join GA GC GE XOR Split XOR Join GD
AND	Traces: (GA, GB, GC, GD, GE), (GA, GB, GD, GC, GE), (GA, GC, GB, GD, GE), (GA, GC, GD, GB, GE), (GA, GD, GB, GC, GE), (GA, GD, GC, GB, GE) Process Model:  AND Split AND Join GB AND Split AND Join GA GC GE AND Split AND Join GD
OR	Traces: (GA, GB, GC, GE), (GA, GB, GD, GE), (GA, GC, GB, GE), (GA, GC, GD, GE), (GA, GD, GB, GE), (GA, GD, GC, GE) Process Model:  OR Split OR Join GB OR Split OR Join GA GC GE OR Split OR Join GD

NFC	Traces: (GA, GB, GC, GD, GE), (GA, GF, GC, GG, GE) Process Model:  Non-Free Choice XOR Split XOR Join XOR Split XOR Join GB GA GC GD GE XOR Split GF XOR Join XOR Split GG XOR Join Non-Free Choice
IPT	Traces: (GA, GB, GC, GD), (GA, GD) Process Model:  XOR Split XOR Join GA XOR Split GB Seq GC XOR Join GD
IPT-NFC	Traces: (GA, GB, GC, GD, GF, GG), (GA, GD, GE, GG) Process Model:  Non-Free Choice XOR Split XOR Join XOR Split XOR Join GE GA GD GG XOR Split GB Seq GC XOR Join XOR Split GF XOR Join Non-Free Choice

The GIT determines XOR, AND, and OR based on the number of outgoing and incoming arrows. The total number of outgoing arrows from k ($n_{outgoing(k)}$) represents the number of relationships in the graph-based process model, where all these relationships have k denotes the initial activity. By contrast, the total number of incoming arrows to k ($n_{incoming(k)}$) is the number of relationships appearing in a graph-based process model, where all of these relations have k as the final activity. An XOR relationship over a set of activities is discovered when each activity in the set has an incoming or an outgoing arrow. An AND relationship occurs over a set of activities when the number of incoming and outgoing arrows is equal to the total number of activities in the set. For example, in a graph-based process model, activities k, l and y

are included in the AND relationship if each activity has three incoming and outgoing arrows. The OR relationship over a set of activities is depicted if the number of incoming arrows or outgoing arrows is more than one but less than the number of activities in the set. An IPT relationship between two activities is formed in the process model if those activities have $\xrightarrow{xorjoin}$ and if one of the activities has $\xrightarrow{xorsplit}$ to another activity. For example, if activity k has $\xrightarrow{xorjoin}$ with activity l and activity k has $\xrightarrow{xorsplit}$ with activity y, an IPT relationship exists between activities k and activity l. An NFC exists if the execution of an activity in an XOR relationship depends on the earlier activity in an XOR relationship. For example, if there is an event log $L4 = \{(GA, GB, GC, GD, GE), (GA, GF, GC, GG, GE)\}$, there are two NFC: $GF \xrightarrow{NFC} GG$ and $GB \xrightarrow{NFC} GD$. Meanwhile, if there is an event log $L5 = \{(GA, GB, GC, GD, GE)$, (GA, GF, GC, GG, GE), (GA, GB, GC, GG, GE), $(GA, GF, GC, GD, GE)\}$, $GF \xrightarrow{NFC} GG$ and $GB \xrightarrow{NFC} GD$ are not depicted by GIT (Sarno et al., 2021) because GF and GG are not always executed in the same trace. This is also true for GB and GD. IPT-NFC is a combination of IPT and NFC.

2.5. Quality Measurements

The PDA quality can be measured using four dimensions: fitness, precision, simplicity, and generalization (Imran et al., 2022; Sarno et al., 2021; Syring et al., 2019). Fitness measures the ability of PDAs to portray all traces of an event log in the process model obtained. The more traces of an event log formed in a process model, the higher the fitness value. Precision measures the fit of a discovered process model. The fitness value emphasizes the traces of an event log, whereas the precision value focuses on traces formed in the obtained process model. Generalization shows how a process model can accommodate the emergence of new traces. Simplicity measures the simplicity of a process model. The ‘simple process model’ depicts precise relationships and nonredundant activities.

3. Methodology

Fig 4 shows the main flow of the GAITN algorithm. The GAITN algorithm solves the scalability problem of the Graph-based Invisible Task (GIT) algorithm (Sarno et al., 2021) by partitioning the event log into event log snippets (Step 1). As shown in the experiment, the rule to discover sequence performs better in processing multiple smaller event log snippets than a big event log (even though the total size of the event log snippets is the same as that of the large event log). In the second step, we applied the rule to form a sequence relationship to all event log snippets and combined the discovered sequences into one graph model. In the final step, GAITN applies rules to form other relationships and adds them to the graph model. The GAITN output is a graph-based process model.

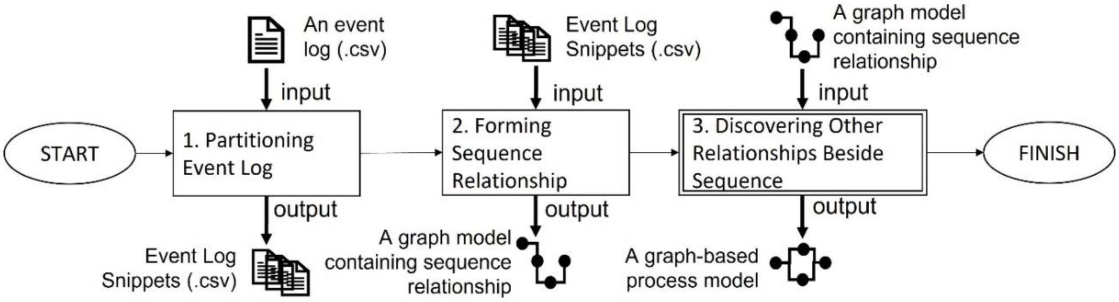

An event log (.csv)
input
Event Log Snippets (.csv)
input
A graph model containing sequence relationship
input
START
1. Partitioning Event Log
2. Forming Sequence Relationship
3. Discovering Other Relationships Beside Sequence
FINISH
output
Event Log Snippets (.csv)
output
A graph model containing sequence relationship
output
A graph-based process model

Fig 4. Flowchart of GAITN

3.1. Partitioning Event Log

A graph database requires high computing time when processing a large volume of data (Sarno et al., 2021). Therefore, GAITN partitions the event log into smaller event log snippets to reduce computing time. The partitioning process is performed using the Hadoop Distributed File System (HDFS) (Hua et al., 2018; Oussous et al., 2018). In our experiment, we set the size of each snippet to 1 MB because the graph database employed in GAITN performs better with smaller data. Based on the five event logs used in this study, a block size of 1 MB stores approximately 500 recorded cases.

3.2. Forming Sequence Relationship

The rules for discovering sequence relationships in GAITN are listed in Algorithm 1. This rule is formed by the CQL syntax of MERGE. This rule uses the timestamp and case ID to determine the sequence relationship between two activities. The rule discovers a sequence relationship between two activities if both activities have the same case ID and the timestamp of the initial activity ($T_{a_{time}}$) is earlier than the timestamp of the end activity ($T_{a+1_{time}}$).

An additional rule was implemented in the first activity of the event log snippet to merge two snippets (lines 2-8). This additional rule is not applicable for the first snippet ($G \neq \emptyset$). First, the rule searches for activities (ga) that have the same case id as the case id of the first activity (gc). If no activity after ga $\left((ga) \rightarrow \emptyset\right)$, then the rule creates a sequence relationship from ga to gc. Another condition for creating a sequence relationship from ga to gc if the next activity of ga (gb) has different case ID with gc. The purpose of the second condition is to connect the first activity of the new snippet with the last activity of the previous snippet, where the last activity already has a sequence relationship with another activity.

Algorithm 1. Algorithm to Discover Sequence Relationship

```
Input: $a\ list\ of\ activities\ in\ an\ event\ \log\ snippet\ (T), number\ of\ activities\ in\ list\ (n_T),$
       $and\ a\ list\ of\ activities\ in\ a\ graph\ process\ model\ (G)$
Output: $an\ updated\ graph\ process\ model$
1:  $\boldsymbol{for}\ a = 0\ \boldsymbol{to}\ n_T - 1\ \boldsymbol{do}$
2:    $\boldsymbol{if} (a = 0) \wedge (G \neq \emptyset)\ \boldsymbol{then}$
3:      $\boldsymbol{for}\ (ga, gb \in G) \wedge ((ga) - [:\text{CASEID}] \rightarrow T_{a_{CaseID}})\ \boldsymbol{do}$
4:        $\boldsymbol{if}\ ((ga) \rightarrow \emptyset) \vee (((ga) - [] \rightarrow (gb)) \wedge \neg((gb) - [:\text{CASEID}] \rightarrow T_{a_{CaseID}}))\boldsymbol{then}$
5:          $gc \leftarrow T_{a_{Name}}$
6:          MERGE $(ga) - [:\text{SEQUENCE}] \rightarrow (gc) - [:\text{CASEID}] \rightarrow (T_{a_{CaseID}})$
7:        $\boldsymbol{endif}$
8:      $\boldsymbol{endfor}$
9:    $\boldsymbol{else}$
10:     $\boldsymbol{if}\ \left(T_{a_{CaseID}} = T_{a+1_{CaseID}}\right) \wedge \left(T_{a_{time}} < T_{a+1_{time}}\right)\ \boldsymbol{then}$
11:       $ga \leftarrow T_{a_{Name}}, gb \leftarrow T_{a+1_{Name}}$
12:       MERGE $(T_{a_{CaseID}}) \leftarrow [:\text{CASEID}] \leftarrow (ga) - [:\text{SEQUENCE}] \rightarrow (gb) - [:\text{CASEID}] \rightarrow \left(T_{a+1_{CaseID}}\right)$
13:     $\boldsymbol{endif}$
14:   $\boldsymbol{endif}$
15: $\boldsymbol{endfor}$
```

3.3. Discovering Other Relationships Beside Sequence

After discovering the sequence relationships, the GAITN applies the rules implemented in the GIT to discover the XOR, OR, AND, invisible prime tasks (IPT), and NFC relationships. Details of this step are reported in (Sarno et al., 2021). The next step was to discover invisible tasks of stacked branching relationships and invisible tasks in non-free choice.

3.3.1 Constructing Invisible Tasks of Stacked Branching Relationships

The GAITN algorithm uses the outputs of the GIT algorithm to form invisible tasks of stacked branching relationship (IT-SBR). GIT detected stacked branching XOR-AND relationships and stacked branching XOR-OR relationships as OR relationships, so GIT could not mine stacked branching relationships. However, we can create a rule to discover IT-SBR by utilising the number of recorded activities of cases in the event log. The rules for discarding the IT-SBR are shown in Algorithm 2. Lines 5-20 are rules for constructing the IT-SBR in SPLIT relationships and lines 21-30 for forming the IT-SBR in JOIN relationships.

The GAITN algorithm applies the rules if there is an activity (ga) that has XOR relationship with an activity (gb) and OR relationships with other activities ($gc_1, gc_2, \ldots, gc_n$) in the graph process model. Based on lines 6-12, $n_{sameCaseID}$ is a number of cases containing all activities ($gc_1, gc_2, \ldots, gc_n$) and $n_{differentCaseID}$ is a number of cases containing one or several activities of ($gc_1, gc_2, \ldots, gc_n$). GAITN constructs invisible tasks of stacked branching XOR-AND relationships if $n_{differentCaseID}$ is 0 and forms invisible tasks of stacked branching XOR-OR relationships if $n_{differentCaseID}$ is greater than 0.

Algorithm 2. Algorithm to Discover Invisible Tasks of Stacked Branching Relationships (IT-SBR)

Input: $a\ list\ of\ activities\ (G)\ and\ a\ list\ of\ case\ ID\ (\mathrm{kID})\ of\ a\ graph\ process\ model$

Output: $an\ updated\ graph\ process\ model$

```
1:   IT $\leftarrow invisible\ task$
2:   $n_{differentCaseID} \leftarrow 0$
3:   $n_{sameCaseID} \leftarrow 0$
4:   $\boldsymbol{for}\ (ga \in G)\ \boldsymbol{do}$
5:     $\boldsymbol{if}\ \big((ga) - [:\mathrm{XORSPLIT}] \rightarrow (gb)\big) \wedge \big((ga) - [:\mathrm{ORSPLIT}] \rightarrow (gc_1, gc_2, \ldots, gc_n)\big)\ \boldsymbol{then}$
6:       $\boldsymbol{for}(k \in \mathrm{kID})\ \boldsymbol{do}$
7:         $\boldsymbol{if}\ (\forall(gc_1, gc_2, \ldots, gc_n) - [:\mathrm{CASEID}] \rightarrow k)\ \boldsymbol{then}$
8:           $n_{sameCaseID} \leftarrow n_{sameCaseID} + 1$
9:         $\boldsymbol{else\ if}\ (\exists(gc_1, gc_2, \ldots, gc_n) - [:\mathrm{CASEID}] \rightarrow k)\ \boldsymbol{then}$
10:          $n_{differentCaseID} \leftarrow n_{differentCaseID} + 1$
11:        $\boldsymbol{endif}$
12:      $\boldsymbol{endfor}$
13:      $\boldsymbol{if}\ n_{differentCaseID} = 0\ \boldsymbol{then}$
14:        MERGE $(ga) - [:\mathrm{XORSPLIT}] \rightarrow (i{:}\mathrm{IT})$
15:        MERGE $(i{:}\mathrm{IT}) - [:\mathrm{ANDSPLIT}] \rightarrow (gc_1, gc_2, \ldots, gc_n)$
16:      $\boldsymbol{else\ then}$
17:        MERGE $(ga) - [:\mathrm{XORSPLIT}] \rightarrow (i{:}\mathrm{IT})$
18         MERGE $(i{:}\mathrm{IT}) - [:\mathrm{ORSPLIT}] \rightarrow (gc_1, gc_2, \ldots, gc_n)$
19:      $\boldsymbol{endif}$
20:    $\boldsymbol{endif}$
21:    $\boldsymbol{if}\ \big((ga) \leftarrow [:\mathrm{XORJOIN}] - (gb)\big) \wedge \big((ga) \leftarrow [:\mathrm{ORJOIN}] - (gc_1, gc_2, \ldots, gc_n)\big)\ \boldsymbol{then}$
22:      $do\ lines\ 6 - 12$
23:      $\boldsymbol{if}\ n_{differentCaseID} = 0\ \boldsymbol{then}$
```

Algorithm 2. Algorithm to Discover Invisible Tasks of Stacked Branching Relationships (IT-SBR)

```
            $\textbf{MERGE}\ (i{:}\,\mathrm{IT}) - [:\mathrm{XORJOIN}] \rightarrow (ga)$
            $\textbf{MERGE}\ (gc_1, gc_2, \dots, gc_n) - [:\mathrm{ANDJOIN}] \rightarrow (i{:}\,\mathrm{IT})$
          $\boldsymbol{else\ then}$
            $\textbf{MERGE}\ (i{:}\,\mathrm{IT}) - [:\mathrm{XORJOIN}] \rightarrow (ga)$
            $\textbf{MERGE}\ (gc_1, gc_2, \dots, gc_n) - [:\mathrm{ORJOIN}] \rightarrow (i{:}\,\mathrm{IT})$
          $\boldsymbol{endif}$
        $\boldsymbol{endif}$
    $\boldsymbol{endfor}$
```

3.3.2 Constructing Invisible Tasks in Non-Free Choice for Stacked Branching Relationships

Algorithm 3 describes the rules for discovering Invisible Tasks in Non-Free Choice for Stacked Branching Relationships (IT-SBR-NFC). There were three steps: discovering an NFC from an invisible task to another activity (lines 3-7); forming an NFC from one activity to an invisible task (lines 8-12); and describing the NFC relationship from one invisible task to another invisible task (lines 13-19). GAITN searches for an activity (say gc) that has a XORJOIN relationship and a XORSPLIT relationship with other activities. An NFC from an invisible task to an activity (say ga) is formed when the invisible task has a XORJOIN relationship with gc, gc has a XORSPLIT relationship with ga and the set of activities related to the invisible task has the same Case ID as ga. An NFC from an activity (ga) to an invisible task is formed when gc has a XORSPLIT to the invisible task and a XORJOIN from ga, and the set of activities related to the invisible task has the same Case ID as ga. GAITN discovers an NFC from one invisible task to another invisible task if the invisible task has an XORSPLIT relationship to gc, another invisible task has an XORJOIN relationship from gc, and the activities related to both invisible tasks have the same Case ID.

Algorithm 3. Algorithm to Discover IT-SBR in Non-Free Choice (IT-SBR-NFC)

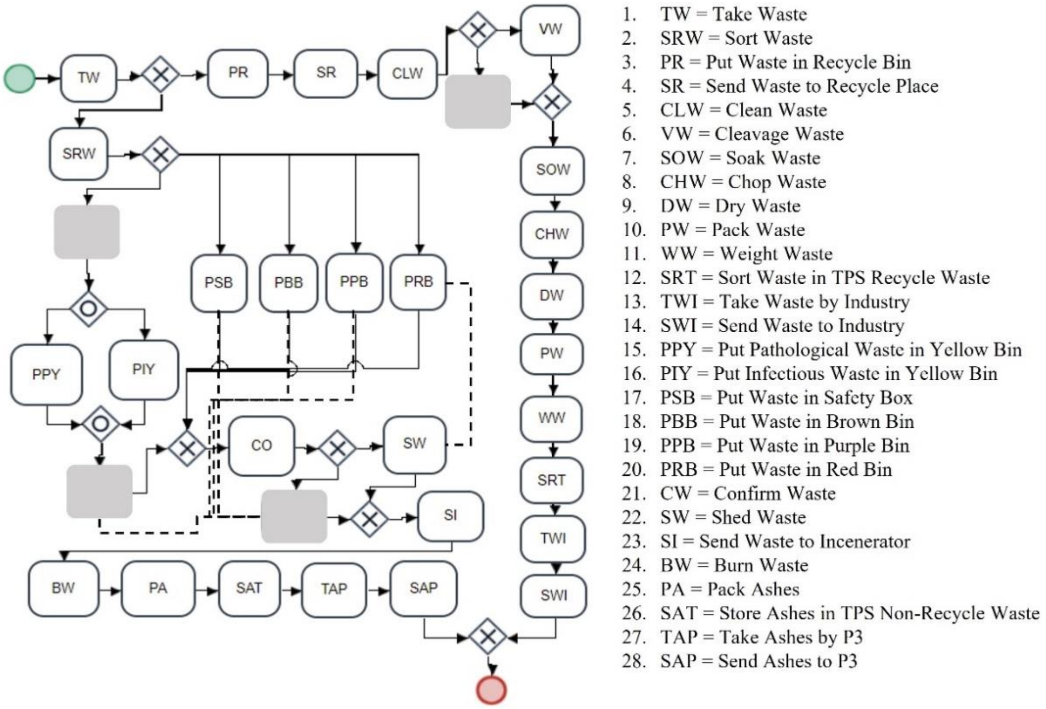
```
Input: $a\ list\ of\ activities\ (G)\ of\ a\ graph\ process\ model$
Output: $an\ updated\ graph\ process\ model$
1:    $k \leftarrow name\ of\ case\ ID,\ \mathrm{IT} \leftarrow invisible\ task$
2:    $\boldsymbol{for}\ (ga, gb, gc \in G)\ \boldsymbol{do}$
3:      $\boldsymbol{if}\ ((i{:}\,\mathrm{IT}) - [:\mathrm{JOIN}] \rightarrow (gc) - [:\mathrm{SPLIT}\,] \rightarrow (ga))\ \boldsymbol{do}$
4:        $\boldsymbol{if}\ ((gb) - [\,] \rightarrow (i{:}\,\mathrm{IT})) \wedge \big((ga) - [:\mathrm{CASEID}] \rightarrow k \leftarrow [:\mathrm{CASEID}] - (gb)\big)\boldsymbol{then}$
5:          $\textbf{MERGE}\ (i) - [:\mathrm{NONFREECHOICE}] \rightarrow (ga)$
6:        $\boldsymbol{endif}$
7:      $\boldsymbol{endif}$
8:      $\boldsymbol{if}\ ((ga) - [:\mathrm{JOIN}] \rightarrow (gc) - [:\mathrm{SPLIT}\,] \rightarrow (i{:}\,\mathrm{IT}))\ \boldsymbol{do}$
9:        $\boldsymbol{if}\ ((i{:}\,\mathrm{IT}) - [\,] \rightarrow (gb)) \wedge \big((ga) - [:\mathrm{CASEID}] \rightarrow k \leftarrow [:\mathrm{CASEID}] - (gb)\big)\boldsymbol{then}$
10:         $\textbf{MERGE}\ (ga) - [:\mathrm{NONFREECHOICE}] \rightarrow (i)$
```

```
        $\boldsymbol{endif}$
      $\boldsymbol{endif}$
      $\boldsymbol{if}\ ((i\text{: IT}) - [\text{: JOIN}] \rightarrow (gc) - [\text{: SPLIT }] \rightarrow (j\text{: IT}))\ \boldsymbol{do}$
        $\boldsymbol{if}\ ((ga) - [\,] \rightarrow (i\text{: IT})) \wedge ((gb) \leftarrow [\,] - (j\text{: IT}))\ \boldsymbol{then}$
          $\boldsymbol{if}\ \big((ga) - [\text{: CASEID}] \rightarrow k \leftarrow [\text{: CASEID}] - (gb)\big)\ \boldsymbol{then}$
            $\textbf{MERGE}\ (i) - [\text{: NONFREECHOICE}] \rightarrow (j)$
          $\boldsymbol{endif}$
        $\boldsymbol{endif}$
      $\boldsymbol{endif}$
    $\boldsymbol{endfor}$
```

4 Experiments

In the first experiment, we compared the quality of the process model discovered by the GAITN, GIT, $\alpha^{\$}$, and Fodina algorithms, using the four measurements. In this experiment, we used four synthetic event logs: L1, L2, solid medical waste handling simulation and medical record (Sarno et al., 2021).

We show the standard operating procedure (SOP) of solid medical waste handling in our case using the Business Process Modeling Notation (BPMN) (Marin-Castro & Tello-Leal, 2021). In Fig 5, the activity names are shown in their initials and the corresponding full names are given in the list next to the diagram. The figure shows several relationships, such as sequence (denoted by →), XOR (denoted by a diamond with X icon), OR (denoted by a diamond with O icon), invisible tasks (denoted by gray boxes), and the NFC relationship (shown by a dotted line; for example, if activity PRB is executed in the following XOR choice, activity SW must be selected). Appendix A provides a detailed explanation of the solid medical waste handling simulation model.



1. TW = Take Waste
2. SRW = Sort Waste
3. PR = Put Waste in Recycle Bin
4. SR = Send Waste to Recycle Place
5. CLW = Clean Waste
6. VW = Cleavage Waste
7. SOW = Soak Waste
8. CHW = Chop Waste
9. DW = Dry Waste
10. PW = Pack Waste
11. WW = Weight Waste
12. SRT = Sort Waste in TPS Recycle Waste
13. TWI = Take Waste by Industry
14. SWI = Send Waste to Industry
15. PPY = Put Pathological Waste in Yellow Bin
16. PIY = Put Infectious Waste in Yellow Bin
17. PSB = Put Waste in Safety Box
18. PBB = Put Waste in Brown Bin
19. PPB = Put Waste in Purple Bin
20. PRB = Put Waste in Red Bin
21. CW = Confirm Waste
22. SW = Shed Waste
23. SI = Send Waste to Incenerator
24. BW = Burn Waste
25. PA = Pack Ashes
26. SAT = Store Ashes in TPS Non-Recycle Waste
27. TAP = Take Ashes by P3
28. SAP = Send Ashes to P3

Fig 5. Standard Operating Procedure for Solid Medical Waste Handling

To measure the scalability of algorithms, we added three real-life event logs: BPIC 2011 Hospital (B. F. van Dongen, 2012), DomesticDeclarations (B. van Dongen, 2020), BPI Challenge 2012 (B. van Dongen, 2012a). The detail information of event logs is described in Table 3.

Table 3. Event Logs for Experiment

Event Log	Total Events	Total Cases	Total Traces	Reference
L2 – IT-SBR-NFC of XOR-OR relationships	**1.7 x 10^2** (170)	35	7	This paper (Fig 3)
L1 – IT-SBR-NFC of XOR-AND relationships	**2.0 x 10^2** (200)	35	7	This paper (Fig 3)
Medical Record Small	**5.5 x 10^2** (552)	48	48	(Sarno et al., 2021)
Medical Record	**3.0 x 10^3** (3,084)	306	303	(Sarno et al., 2021)
Solid Waste Handling	**5.9 x 10^3** (5,941)	478	10	This paper (Fig 5)
DomesticDeclarations	**5.6 x 10^4** (56,437)	10,500	99	(B. van Dongen, 2020)
BPIC 2011 Hospital Log	**1.5 x 10^5** (150,291)	1,143	981	(B. F. van Dongen, 2012)

Event Log	Total Events	Total Cases	Total Traces	Reference
BPI Challenge 2012	**2.6 x 10^5** (262,200)	13,087	4,366	(B. van Dongen, 2012b)

A process model discovered by the GAITN from the solid-waste handling event log is shown in Fig 6, while other process models are presented in Appendix B. $\alpha^\$$ produces a timeout error from a log of a solid waste handling; hence, the process model cannot be identified. GAITN has successfully discovered invisible tasks, NFC, and IT-SBR-NFC, which are denoted by grey circles, red lines, and grey circles connected by red lines, respectively. GIT misinterprets the stacked branching XOR-AND relationships and the stacked branching XOR-OR relationships with OR relationships, while Fodina discovers the stacked branching XOR-OR relationships as AND relationships. Neither GIT nor Fodina discovered IT-SBR-NFC.

The results of the first experiment and the average scores of the results are listed in Table 4. The inability of forming a process model from solid waste-handling event log causes $\alpha^\$$ has lowest scores in fitness, generalisation, and simplicity. GIT and Fodina get low precision scores because those algorithms discover many traces that do not match the event log due to inability to form IT-SBR and IT-SBR-NFC. GAITN gets the highest average scores of fitness, precision, generalisation, and simplicity among other algorithms. This can be attributed to the fact that it can form OR, NFC, and IT-SBR-NFC relationships.

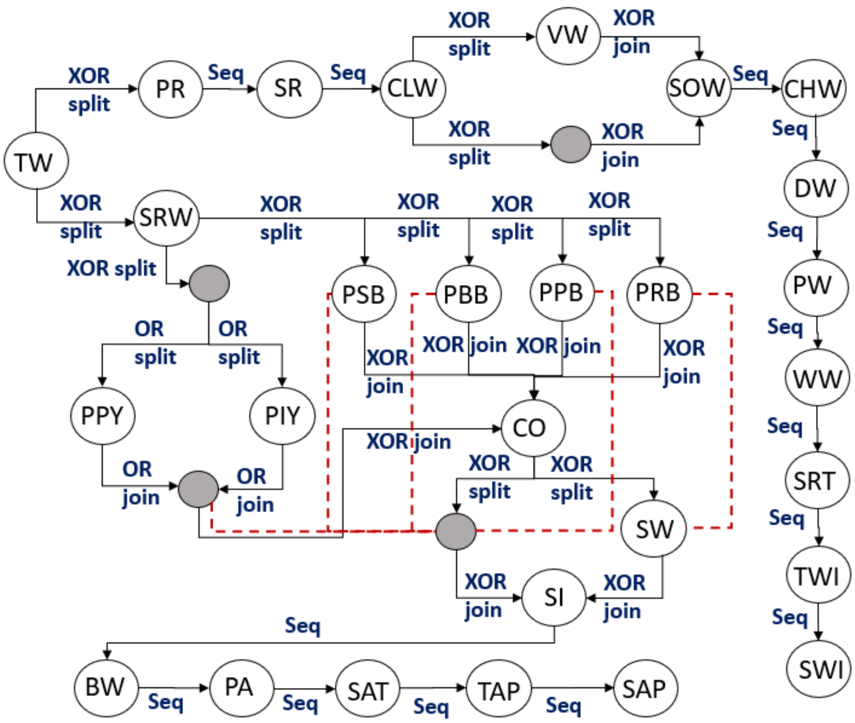


Fig 6. A solid medical waste handling process model obtained by the proposed method (GAITN)

Table 4. Quality Measurements of Obtained Process Models Discovered by Algorithms

Event Log	**Algorithm**	**Fitness (0.0 – 1.0)**	**Precision (0.0 – 1.0)**	**Generalisation (0.0 – 1.0)**	**Simplicity (0.0 – 1.0)**
L1 – IT-SBR-NFC of XOR-AND relationships	Proposed method (GAITN)	1.000	1.000	0.930	1.000
	GIT	0.857	0.083	0.929	0.909
	$\alpha^{\$}$	1.000	0.250	0.933	0.813
	Fodina	1.000	0.500	0.930	1.000
L2 – IT-SBR-NFC of XOR-OR relationships	Proposed method (GAITN)	1.000	1.000	0.930	1.000
	GIT	0.857	0.083	0.933	0.909
	$\alpha^{\$}$	0.143	0.036	0.920	0.750
	Fodina	0.140	0.070	0.929	0.917
Solid Medical Waste Handling	Proposed method (GAITN)	1.000	1.000	0.808	1.000
	GIT	0.400	0.002	0.804	0.972
	$\alpha^{\$}$	-	-	-	-
	Fodina	0.800	0.444	0.806	0.944
Medical Record	Proposed method (GAITN)	1.000	1.000	0.970	1.000
	GIT	1.000	1.000	0.970	1.000
	$\alpha^{\$}$	1.000	1.000	0.970	1.000

Event Log	Algorithm	Fitness (0.0 – 1.0)	Precision (0.0 – 1.0)	Generalisation (0.0 – 1.0)	Simplicity (0.0 – 1.0)
	Fodina	1.000	0.870	0.970	1.000
$\bar{x}$	**Proposed method (GAITN)**	**1.000**	**1.000**	**0.910**	**1.000**
	GIT	**0.779**	**0.292**	**0.909**	**0.948**
	$\alpha^{\$}$	**0.536**	**0.322**	**0.706**	**0.641**
	Fodina	**0.735**	**0.471**	**0.909**	**0.965**

Note: - indicates that the algorithm cannot complete the task (timeout error), $\bar{x}$ is the average measurement value for each algorithm

In the second experiment, we compared the maximum activity that can be formed by GAITN, GIT, $\alpha^{\$}$ and Fodina algorithms. As shown in Table 5, GAITN is the only algorithm that can process models up to 2.6 x 10^5 events, while GIT, Fodina, and $\alpha^{\$}$ handled 1.5 x 10^5, 5.6 x 10^4 and 5.6 x 10^4, respectively. The computing time of GIT and Fodina increased slowly; however, $\alpha^{\$}$ has a sharp increase in computing time when the number of events reached 5.6 x 10^4 events. Fodina has the fastest computing time among other algorithms; however, this algorithm cannot handle more 5.6 x 10^4 events. The increase in computing times of GAITN, GIT, and $\alpha^{\$}$ in line with the increase in number of events. The computing time of Fodina is more affected by the number of traces than number of events. The experiments show that GAITN can handle more events than the other algorithms.

Table 5. Scalability: Computing Time of Discovery and Maximum Handled Events

Total Events	Computing Time of Discovery (s)			
	Proposed method (GAITN)	**$\alpha^{\$}$**	**GIT**	**Fodina**
1.7 x 10^2	0.340	0.153	0.340	0.168
2.0 x 10^2	0.342	0.158	0.342	0.170
5.5 x 10^2	0.849	0.417	0.849	0.590
3.0 x 10^3	2.978	1.235	2.978	0.780
5.9 x 10^3	8.381	3.351	8.381	0.350
5.6 x 10^4	11.971	519.890	11.971	0.650
1.5 x 10^5	32.385	-	32.385	-

Total Events	Computing Time of Discovery (s)			
	Proposed method (GAITN)	$\alpha^{\$}$	GIT	Fodina
2.6 x 10^5	45.274	-	-	-

Note: - indicates that the algorithm cannot complete the task (timeout error)

5. Conclusions

In this study, we have proposed an algorithm called GAITN, which utilises a graph database to model invisible non-prime task, identify invisible tasks of stacked branching relationships (IT-SBR) in non-free choice (IT-SBR-NFC) and form process models based on huge volumes of event logs. We have shown that storing relationships in a graph database can simplify the rules of process discovery, because a relationship can be constructed based on other discovered relationships. For example, using a graph database, the IT-SBR was mined using obtained branching relationships, and IT-SBR-NFC construct was formed using IT-SBR and XOR relationships. This also makes it easier to extend an algorithm, as shown in our case, where we extend the GIT to the GAITN simply by adding rules that form IT-SBR and IT-SBR-NFC relationship using relationships that can be discovered by the GIT. Experiments have shown that GAITN performs better than GIT, $\alpha^{\$}$ and Fodina, based on fitness, precision, simplicity measures, and the maximum activity that can be handled. The result also shows that our idea of splitting event logs and creating new rules of sequence relationships allows GAITN to handle large event logs. GAITN enhances the model quality by its capability to mine IT-SBR and ITS-BR-NFC and increases the scalability of process discovery by partitioning event logs and merging snippets of the event log.

The drawbacks of using graph database in the graph-based process mining algorithm is the computing time increases significantly when processing large amounts of events. In future work, we aim to as a pre-processing stage or optimise Cypher Query Language (CQL) to reduce computing time of the GAITN.

Conflict of Interest

No potential conflict of interest is reported by the authors.

Ethical Approval and Consent to Participate

Ethical approval was not required, because the event logs were public datasets or simulated event logs based on public process models.

References

Anugrah, I. G., Sarno, R., & Anggraini, R. N. E. (2015). Decomposition using Refined Process Structure Tree (RPST) and control flow complexity metrics. *2015 International Conference on Information & Communication Technology and Systems (ICTS)*, 203–208. https://doi.org/10.1109/ICTS.2015.7379899

Back, C. O., Manataki, A., & Harrison, E. (2020). Mining patient flow patterns in a surgical ward. *HEALTHINF 2020 - 13th International Conference on Health Informatics, Proceedings; Part of 13th International Joint Conference on Biomedical Engineering Systems and Technologies, BIOSTEC 2020*, *Biostec*, 273–283. https://doi.org/10.5220/0009181302730283

Battineni, G., Chintalapudi, N., & Amenta, F. (2020). Model discovery, and replay fitness validation using inductive mining techniques in medical training of CVC surgery. *Applied Computing and Informatics*. https://doi.org/10.1016/j.aci.2020.01.001

Beeson, I., Green, S., Sa, J., & Sully, A. (2002). Linking Business Processes and Information Systems Provision in a Dynamic Environment. *Information Systems Frontiers*, *4*(3), 317–329. https://doi.org/10.1023/A:1019910722321

Berger, S., van Dun, C., & Häckel, B. (2022). IT Availability Risks in Smart Factory Networks – Analyzing the Effects of IT Threats on Production Processes Using Petri Nets. *Information Systems Frontiers*. https://doi.org/10.1007/s10796-022-10243-y

Choueiri, A. C., & Portela Santos, E. A. (2021). Discovery of path-attribute dependency in manufacturing environments: A process mining approach. *Journal of Manufacturing Systems*, *61*, 54–65. https://doi.org/https://doi.org/10.1016/j.jmsy.2021.08.005

Choueiri, A. C., Sato, D. M. V., Scalabrin, E. E., & Santos, E. A. P. (2020). An extended model for remaining time prediction in manufacturing systems using process mining. *Journal of Manufacturing Systems*, *56*, 188–201. https://doi.org/https://doi.org/10.1016/j.jmsy.2020.06.003

De Roock, E., & Martin, N. (2022). Process mining in healthcare – An updated perspective on the state of the art. *Journal of Biomedical Informatics*, *127*(November 2021), 103995. https://doi.org/10.1016/j.jbi.2022.103995

Erdogan, T. G., & Tarhan, A. (2018). Systematic Mapping of Process Mining Studies in Healthcare. *IEEE Access*, *6*, 24543–24567. https://doi.org/10.1109/ACCESS.2018.2831244

Francis, N., Green, A., Guagliardo, P., Libkin, L., Lindaaker, T., Marsault, V., Plantikow, S., Rydberg, M., Selmer, P., & Taylor, A. (2018). Cypher: An Evolving Query Language for Property Graphs. *Proceedings of the 2018 International Conference on Management of Data*, 1433–1445. https://doi.org/10.1145/3183713.3190657

Guo, Q., Wen, L., Wang, J., Yan, Z., & Yu, P. S. (2015). Mining Invisible Tasks in Non-free-choice Constructs. In *Lecture Notes in Computer Science* (pp. 109–125). Springer International Publishing. https://doi.org/10.1007/978-3-319-23063-4_7

Hamdani, A., & Abdelli, A. (2020). Towards modelling and analyzing timed workflow systems with complex synchronizations. *Journal of King Saud University - Computer and Information Sciences*, *32*(4), 491–504. https://doi.org/10.1016/j.jksuci.2019.08.007

Hua, X., Huang, M. C., & Liu, P. (2018). Hadoop Configuration Tuning with Ensemble Modeling and Metaheuristic Optimization. *IEEE Access*, *6*, 44161–44174. https://doi.org/10.1109/ACCESS.2018.2857852

Imran, M., Ismail, M. A., Hamid, S., & Nasir, M. H. N. M. (2022). Complex Process Modeling in Process Mining: A Systematic Review. *IEEE Access*, *10*(September), 101515–101536. https://doi.org/10.1109/ACCESS.2022.3208231

Kim, K. S., Pham, D. L., Park, Y. I., & Kim, K. P. (2022). Experimental verification and validation of the SICN-oriented process mining algorithm and system. *Journal of King Saud University - Computer and Information Sciences*, *34*(10), 9793–9813. https://doi.org/10.1016/j.jksuci.2021.12.013

Kurniati, A. P., Kusuma, G., & Wisudiawan, G. (2016). Implementing Heuristic Miner for Different Types of Event Logs. *International Journal of Applied Engineering Research*, *11*(8), 5523–5529.

Marin-Castro, H. M., & Tello-Leal, E. (2021). An end-to-end approach and tool for BPMN process discovery. *Expert Systems with Applications*, *174*(January 2020), 114662. https://doi.org/10.1016/j.eswa.2021.114662

Namaki, S., Fontanili, F., Lamine, E., & Okongwu, U. (2022). Stable heuristic miner : Applying statistical stability to discover the common patient pathways from location event logs. *Intelligent Systems with Applications*, *14*. https://doi.org/10.1016/j.iswa.2022.200071

Oussous, A., Benjelloun, F.-Z., Ait Lahcen, A., & Belfkih, S. (2018). Big Data technologies: A survey. *Journal of King Saud University - Computer and Information Sciences*, *30*(4), 431–448. https://doi.org/10.1016/j.jksuci.2017.06.001

Pika, A., Wynn, M. T., Budiono, S., Hofstede, A. H. M. T., van der Aalst, W. M. P., & Reijers, H. A. (2020). Privacy-preserving process mining in healthcare. *International Journal of Environmental Research and Public Health*, *17*(5). https://doi.org/10.3390/ijerph17051612

Saad, M., Zhang, Y., Tian, J., & Jia, J. (2023). A graph database for life cycle inventory using Neo4j. *Journal of Cleaner Production*, *393*, 136344. https://doi.org/https://doi.org/10.1016/j.jclepro.2023.136344

Sarno, R., Sungkono, K., Johanes, R., & Sunaryono, D. (2019). Graph-Based Algorithms for Discovering a Process Model Containing Invisible Tasks. *International Journal of Intelligent Engineering and Systems*, *12*(2), 85–94. https://doi.org/10.22266/ijies2019.0430.09

Sarno, R., Sungkono, K. R., Taufiqulsa'di, M., Darmawan, H., Fahmi, A., & Triyana, K. (2021). Improving Efficiency for Discovering Business Processes Containing Invisible tasks in Non-free Choice. *Journal of Big Data*, *8*(113). https://doi.org/10.21203/rs.3.rs-71558/v1

Šestak, M., & Turkanovic, M. (2023). Extended Property-level k -vertex Cardinality Constraints Model for Graph Databases. *Journal of King Saud University – Computer and Information Sciences*, *35*, 126–138. https://doi.org/10.1016/j.jksuci.2023.03.013

Syring, A. F., Tax, N., & van der Aalst, W. M. P. (2019). Evaluating Conformance Measures in Process Mining Using Conformance Propositions. In *Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics): Vol. 11790 LNCS* (pp. 192–221). https://doi.org/10.1007/978-3-662-60651-3_8

van der Aalst, W. (2011). Process mining: discovering and improving Spaghetti and Lasagna processes.

IEEE Symposium on Computational Intelligence and Data Mining, 1–7. https://doi.org/10.1109/cidm.2011.6129461

van der Aalst, W. (2016). *Process Mining* (2nd ed.). Springer Berlin Heidelberg. https://doi.org/10.1007/978-3-662-49851-4

van Dongen, B. (2012a). *BPI Challenge 2012*. Eindhoven University of Technology. https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f

van Dongen, B. (2012b). *BPI Challenge 2012*. https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f

van Dongen, B. (2020). *BPI Challenge 2020: Domestic Declarations*. 4TU.Centre for Research Data. https://doi.org/10.4121/uuid:3f422315-ed9d-4882-891f-e180b5b4feb5

van Dongen, B. F. (2012). *BPIC 2011 Hospital Log*. https://doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f

vanden Broucke, S. K. L. M., & De Weerdt, J. (2017). Fodina: A robust and flexible heuristic process discovery technique. *Decision Support Systems*, *100*, 109–118. https://doi.org/10.1016/j.dss.2017.04.005

Waspada, I., Sarno, R., & Sungkono, K. R. (2020). An Improved Method of Parallel Model Detection for Graph-Based Process Model Discovery. *International Journal of Intelligent Engineering and Systems*, *13*(2), 127–138. https://doi.org/10.22266/ijies2020.0430.13

Weber, P., Backman, R., Litchfield, I., & Lee, M. (2018). A Process Mining and Text Analysis Approach to Analyse the Extent of Polypharmacy in Medical Prescribing. *2018 IEEE International Conference on Healthcare Informatics (ICHI)*, 1–11. https://doi.org/10.1109/ICHI.2018.00008

Wen, L., van der Aalst, W. M. P., Wang, J., & Sun, J. (2007). Mining process models with non-free-choice constructs. *Data Mining and Knowledge Discovery*, *15*(2), 145–180. https://doi.org/10.1007/s10618-007-0065-y

Wen, L., Wang, J., van der Aalst, W. M. P., Huang, B., & Sun, J. (2010). Mining process models with prime invisible tasks. *Data & Knowledge Engineering*, *69*(10), 999–1021. https://doi.org/10.1016/j.datak.2010.06.001

Yan, Z., Sun, B., Chen, Y., Wen, L., Hu, L., Wang, J., Yang, M., & Wang, L. (2019). Decomposed and parallel process discovery: A framework and application. *Future Generation Computer Systems*, *98*, 392–405. https://doi.org/https://doi.org/10.1016/j.future.2019.03.048

Zayoud, M., Kotb, Y., & Ionescu, S. (2019). β Algorithm: A New Probabilistic Process Learning Approach for Big Data in Healthcare. *IEEE Access*, *7*, 78842–78869. https://doi.org/10.1109/ACCESS.2019.2922635

Zheng, W., Du, Y., Wang, S., & Qi, L. (2019). Repair Process Models Containing Non-Free-Choice Structures Based on Logic Petri Nets. *IEEE Access*, *7*, 105132–105145. https://doi.org/10.1109/ACCESS.2019.2932260

APPENDIX A
A Solid Medical Waste Handling Simulation Model

A.1. *Purpose of the model*

This study presents a discrete event simulation (DES) model to simulate a standard operating procedure (SOP) for Solid Medical Waste Handling and generates an event log based on the simulation. Fig 5 presents the SOP of Solid Medical Waste Handling.

A.2. *Model output*

The output is an event log that records the executed activities of all traces in the simulation model.

A.3. *Software or programming language*

DES model is developed using Anylogic 8.7.5 Personal Learning Edition

A.4. *Model Structure*

The DES model is shown in Fig 7.

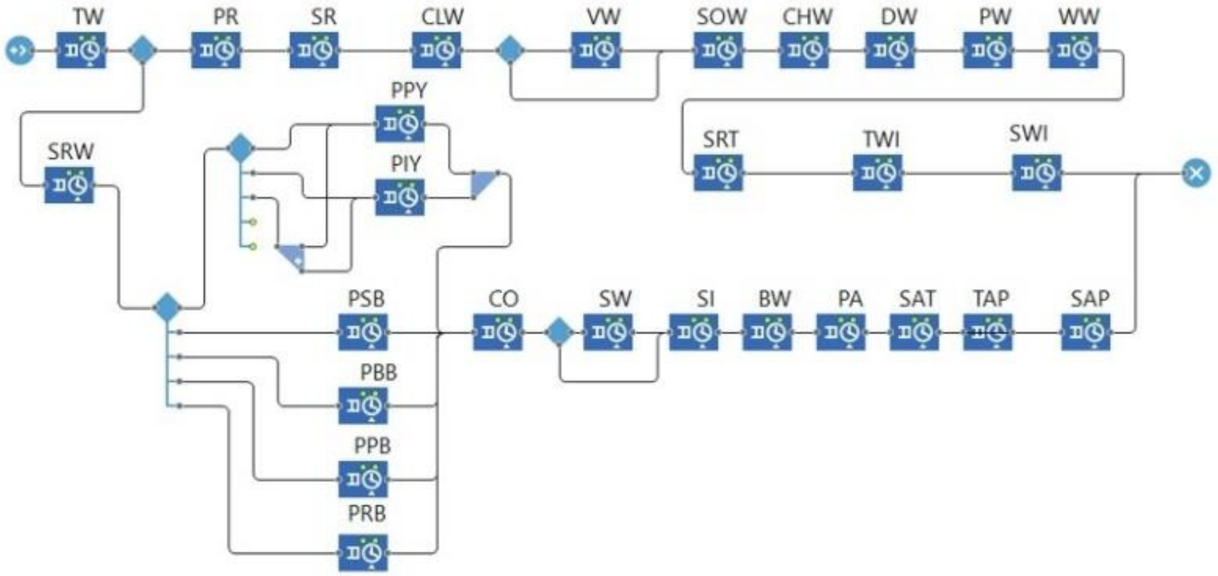
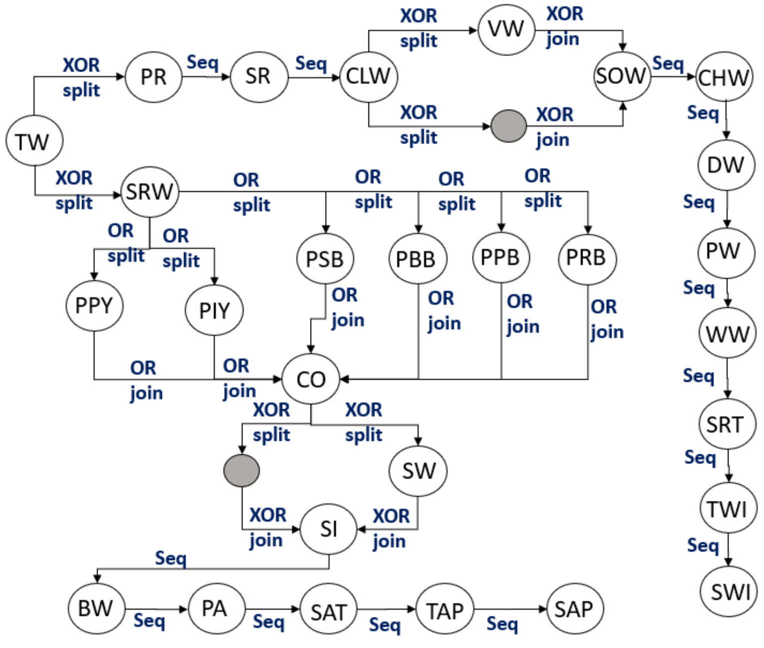


Fig 7. A simulation model in Anylogic Software

APPENDIX B

Table 6. Process Models from solid-waste handling event log

GIT



α$

No process model because the algorithm cannot process the event log (timeout)

Fodina

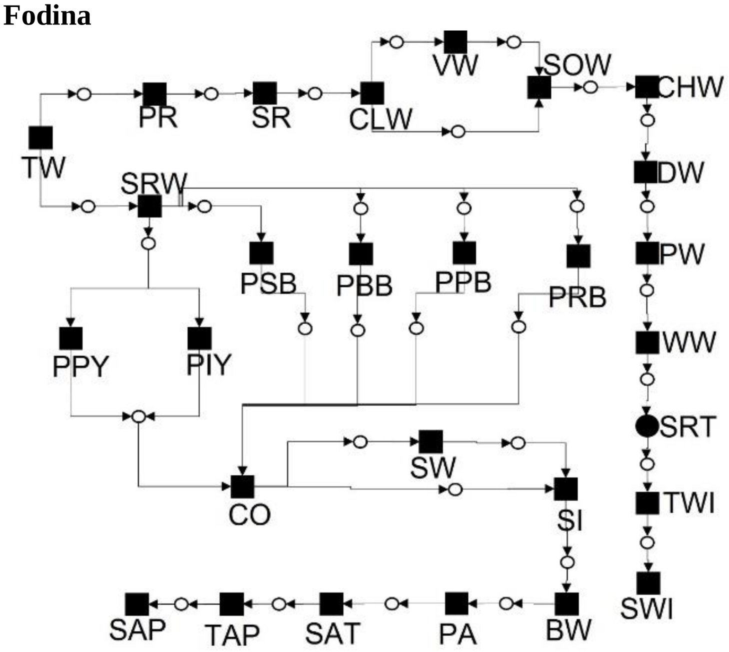
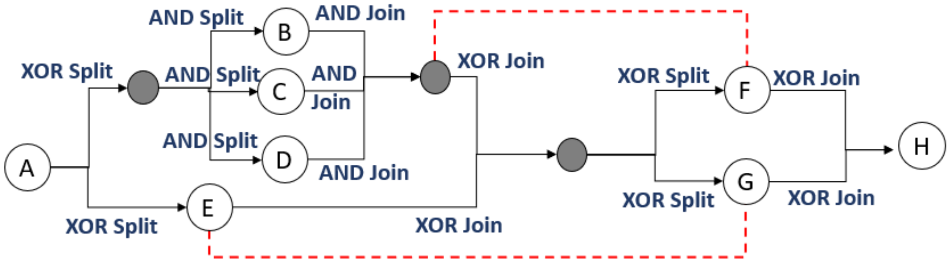


Table 7. Process Models from L1 event log

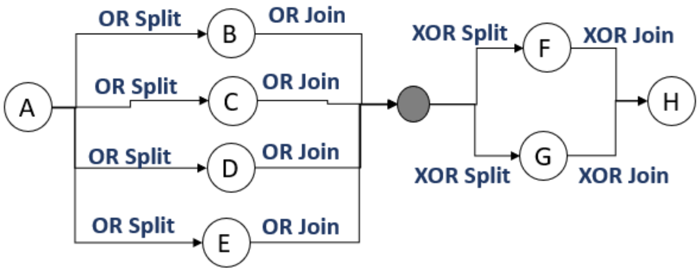
<table>
<tr><td>Traces of Event Log</td><td>L1 = { (A,B, C, D, F, H), (A, B, D, C, F, H), (A, C, B, D, F, H), (A, C, D, B, F, H), (A, D, C, B, F, H), (A, D, B, C, F, G), (A, E, G, H) }</td></tr>
<tr><td colspan="2">

GAITN



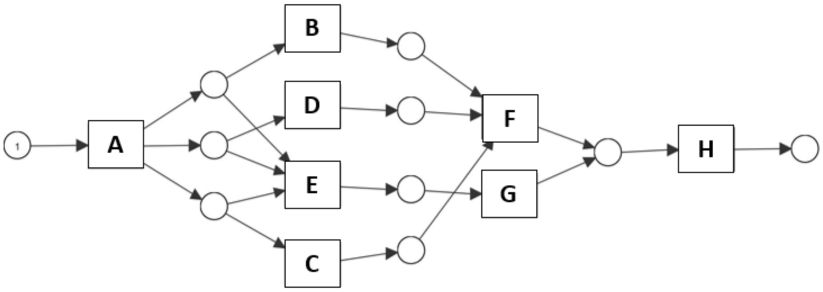
</td></tr>
<tr><td colspan="2">

GIT



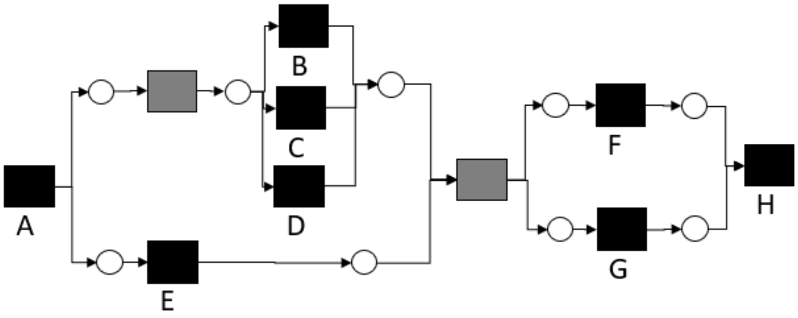
</td></tr>
<tr><td colspan="2">

α$^{\$}$



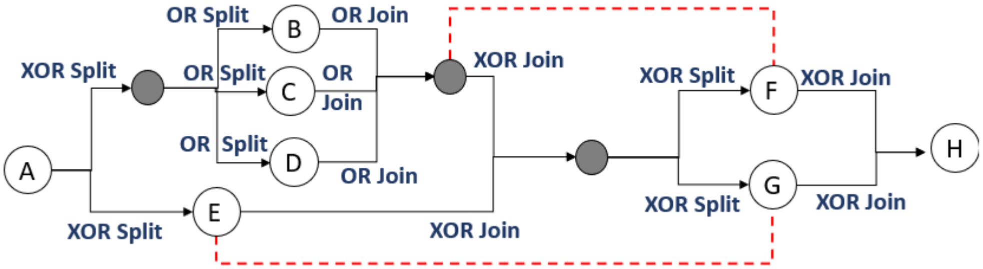
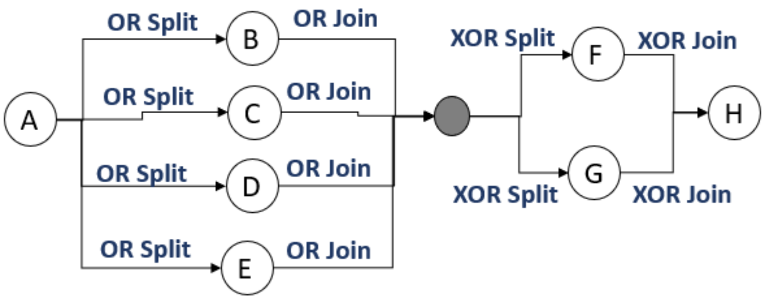
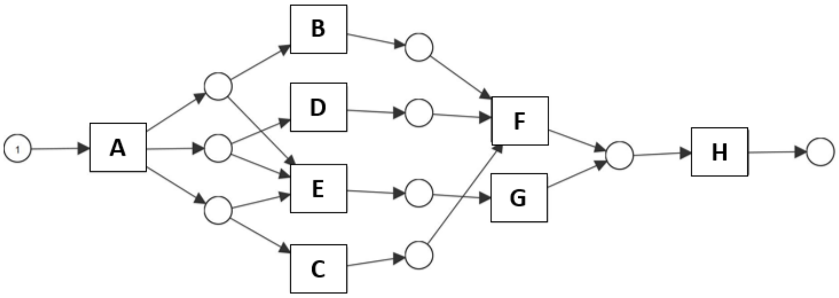
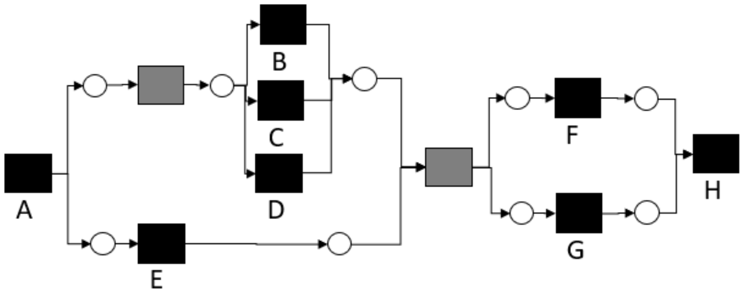
</td></tr>
<tr><td colspan="2">

Fodina



</td></tr>
</table>

Table 8. Process Models from L2 event log

Traces of Event Log	L2 = { (A, B, C, F, H), (A, B, D, F, H), (A, C, B, F, H), (A, C, D, F, H), (A, D, C, F, H), (A, D, B, F, G), (A, E, G, H) }
GAITN	 A XOR Split XOR Split OR Split B OR Join OR Split C OR Join OR Split D OR Join E XOR Join XOR Join XOR Split F XOR Join XOR Split G XOR Join H
GIT	 A OR Split B OR Join OR Split C OR Join OR Split D OR Join OR Split E OR Join XOR Split F XOR Join XOR Split G XOR Join H
α$	
Fodina	

Conflict of Interest

No potential conflict of interest is reported by the authors.

Author Agreement

Submission of work requires that the piece to be reviewed has not been previously published. Upon acceptance, the Author assigns to the Journal of King Saud University - Science (JKSUS) the right to publish and distribute the manuscript in part or in its entirety. The Author's name will always be included with the publication of the manuscript.

The Author has the following nonexclusive rights: (1) to use the manuscript in the Author's teaching activities; (2) to publish the manuscript, or permit its publication, as part of any book the Author may write; (3) to include the manuscript in the Author's own personal or departmental (but not institutional) database or on-line site; and (4) to license reprints of the manuscript to third persons for educational photocopying. The Author also agrees to properly credit the Journal of King Saud University - Science (JKSUS) as the original place of publication.

The Author hereby grants the Journal of King Saud University - Science (JKSUS) full and exclusive rights to the manuscript, all revisions, and the full copyright. The Journal of King Saud University - Science (JKSUS) rights include but are not limited to the following: (1) to reproduce, publish, sell, and distribute copies of the manuscript, selections of the manuscript, and translations and other derivative works based upon the manuscript, in print, audio-visual, electronic, or by any and all media now or hereafter known or devised; (2) to license reprints of the manuscript to third persons for educational photocopying; (3) to license others to create abstracts of the manuscript and to index the manuscript; (4) to license secondary publishers to reproduce the manuscript in print, microform, or any computer-readable form, including electronic on-line databases; and (5) to license the manuscript for document delivery. These exclusive rights run the full term of the copyright, and all renewals and extensions thereof.

I hereby accept the terms of the above Author Agreement.

Author :- Riyanarto Sarno   Date :-

Editor in Chief:- Ahmed H. Alghamdi   Date:-

Author Agreement

Submission of work requires that the piece to be reviewed has not been previously published. Upon acceptance, the Author assigns to the Journal of King Saud University - Science (JKSUS) the right to publish and distribute the manuscript in part or in its entirety. The Author's name will always be included with the publication of the manuscript.

The Author has the following nonexclusive rights: (1) to use the manuscript in the Author's teaching activities; (2) to publish the manuscript, or permit its publication, as part of any book the Author may write; (3) to include the manuscript in the Author's own personal or departmental (but not institutional) database or on-line site; and (4) to license reprints of the manuscript to third persons for educational photocopying. The Author also agrees to properly credit the Journal of King Saud University - Science (JKSUS) as the original place of publication.

The Author hereby grants the Journal of King Saud University - Science (JKSUS) full and exclusive rights to the manuscript, all revisions, and the full copyright. The Journal of King Saud University - Science (JKSUS) rights include but are not limited to the following: (1) to reproduce, publish, sell, and distribute copies of the manuscript, selections of the manuscript, and translations and other derivative works based upon the manuscript, in print, audio-visual, electronic, or by any and all media now or hereafter known or devised; (2) to license reprints of the manuscript to third persons for educational photocopying; (3) to license others to create abstracts of the manuscript and to index the manuscript; (4) to license secondary publishers to reproduce the manuscript in print, microform, or any computer-readable form, including electronic on-line databases; and (5) to license the manuscript for document delivery. These exclusive rights run the full term of the copyright, and all renewals and extensions thereof.

I hereby accept the terms of the above Author Agreement.

Author :- Kelly Rossa Sungkono

Date :-

Editor in Chief:- Ahmed H. Alghamdi

Date:-

Author Agreement

Submission of work requires that the piece to be reviewed has not been previously published. Upon acceptance, the Author assigns to the Journal of King Saud University - Science (JKSUS) the right to publish and distribute the manuscript in part or in its entirety. The Author's name will always be included with the publication of the manuscript.

The Author has the following nonexclusive rights: (1) to use the manuscript in the Author's teaching activities; (2) to publish the manuscript, or permit its publication, as part of any book the Author may write; (3) to include the manuscript in the Author's own personal or departmental (but not institutional) database or on-line site; and (4) to license reprints of the manuscript to third persons for educational photocopying. The Author also agrees to properly credit the Journal of King Saud University - Science (JKSUS) as the original place of publication.

The Author hereby grants the Journal of King Saud University - Science (JKSUS) full and exclusive rights to the manuscript, all revisions, and the full copyright. The Journal of King Saud University - Science (JKSUS) rights include but are not limited to the following: (1) to reproduce, publish, sell, and distribute copies of the manuscript, selections of the manuscript, and translations and other derivative works based upon the manuscript, in print, audio-visual, electronic, or by any and all media now or hereafter known or devised; (2) to license reprints of the manuscript to third persons for educational photocopying; (3) to license others to create abstracts of the manuscript and to index the manuscript; (4) to license secondary publishers to reproduce the manuscript in print, microform, or any computer-readable form, including electronic on-line databases; and (5) to license the manuscript for document delivery. These exclusive rights run the full term of the copyright, and all renewals and extensions thereof.

I hereby accept the terms of the above Author Agreement.

Stephan Onggo

Author :-Bhakti S. Onggo Date :-

Editor in Chief:- Ahmed H. Alghamdi Date:-

Author Agreement

Submission of work requires that the piece to be reviewed has not been previously published. Upon acceptance, the Author assigns to the Journal of King Saud University - Science (JKSUS) the right to publish and distribute the manuscript in part or in its entirety. The Author's name will always be included with the publication of the manuscript.

The Author has the following nonexclusive rights: (1) to use the manuscript in the Author's teaching activities; (2) to publish the manuscript, or permit its publication, as part of any book the Author may write; (3) to include the manuscript in the Author's own personal or departmental (but not institutional) database or on-line site; and (4) to license reprints of the manuscript to third persons for educational photocopying. The Author also agrees to properly credit the Journal of King Saud University - Science (JKSUS) as the original place of publication.

The Author hereby grants the Journal of King Saud University - Science (JKSUS) full and exclusive rights to the manuscript, all revisions, and the full copyright. The Journal of King Saud University - Science (JKSUS) rights include but are not limited to the following: (1) to reproduce, publish, sell, and distribute copies of the manuscript, selections of the manuscript, and translations and other derivative works based upon the manuscript, in print, audio-visual, electronic, or by any and all media now or hereafter known or devised; (2) to license reprints of the manuscript to third persons for educational photocopying; (3) to license others to create abstracts of the manuscript and to index the manuscript; (4) to license secondary publishers to reproduce the manuscript in print, microform, or any computer-readable form, including electronic on-line databases; and (5) to license the manuscript for document delivery. These exclusive rights run the full term of the copyright, and all renewals and extensions thereof.

I hereby accept the terms of the above Author Agreement.

Author :- Muhammad F. Haykal

Date :-

Editor in Chief:- Ahmed H. Alghamdi

Date:-