Computers & Industrial Engineering 99 (2016) 382–400

Contents lists available at ScienceDirect

Computers & Industrial Engineering

journal homepage: www.elsevier.com/locate/caie

Domain-driven actionable process model discovery

Bernardo Nugroho Yahya [a], Minseok Song [b,*], Hyerim Bae [c], Sung-ook Sul [d], Jei-Zheng Wu [e]

[a] Department of Industrial and Management Engineering, Hankuk University of Foreign Studies, Oedae-ro 81, 17035 Yongin, South Korea
[b] Department of Industrial & Management Engineering, POSTECH (Pohang University of Science and Technology), 77 Cheongam-Ro, 37673 Pohang, South Korea
[c] Department of Industrial Engineering, Pusan National University, Pusandaehakyo-ro 63 bongil 2, Geumjong-Gu, 46241 Busan, South Korea
[d] Total Soft Bank, Ltd., 66-39 Bansong-ro 513, Haeundae, 48002 Busan, South Korea
[e] Department of Business Administration, Soochow University, 56 Kueiyang Street, Section 1, Taipei 100, Taiwan, ROC

ARTICLE INFO

Article history:
Available online 7 May 2016

Keywords:
Process mining
Business process
Proximity score
Users' knowledge
Integer linear programming

ABSTRACT

Process discovery is a type of process mining that constructs a process model from the event logs of an information system. The model discovered using process discovery techniques and the process as perceived by users will always differ in some ways and to some extents. In particular, less structured process, such as operational process in business and manufacturing, often result overly confusing, spaghetti-like, process models caused by the inherent complexity of the process. As a result, the mined model has many limitations for providing the users with explicit knowledge that can be directly used to influence behavior for the user's interest. Explicit knowledge, as later called by actionable knowledge, is an important representation on measuring the interestingness of mined patterns. This actionable knowledge, which is incorporated with users' background knowledge and based on some notions of actionable rules, can result an actionable process model. Undoubtedly, domain experts, who know the process well, play a key role to enhance the mined model into an actionable model by their involvements during the discovery process. This paper presents a discovery method to obtain an actionable process model that is based on both the event relation in the log and users' knowledge to improve the incompatibility of the traditional process mining approaches. Users can set their knowledge in terms of constraints. Unlike the existing approach, the proposed approach synthesizes the activity proximity and attempts to extract behavior satisfied by the constraints which may be hidden in the event logs for resulting an actionable process model. In addition, the proposed method is used in order to achieve a sound process model when the existence of the constraints does not satisfy the workflow soundness property. The method was implemented in the ProM framework and tested on a real process.

© 2016 Elsevier Ltd. All rights reserved.

1. Introduction

A process model is a means of communication that facilitates stakeholders' understanding of complex business processes (Kawalek & Kueng, 1997). Stakeholders typically formalize a process model using traditional top-down *a priori* design approaches. Although they have their own understandings about process model flow, a real application might result in different sequences. Process discovery, a type of process mining, constructs a process model from information systems' event logs. Such a discovered model, most probably, will differ from the process as perceived by stakeholders, since the discovery techniques can automatically derive a process model based on the statistics hidden in the event logs. As a result, stakeholders, most of the time, find difficulties on understanding the extracted patterns from the discovered model. Since the extracted patterns from process discovery are mostly related to the technical interestingness for process analysts, the business interestingness, called as actionable model, is often neglected.

In the field of knowledge discovery, the involvement of users, i.e., stakeholders, takes place to examine the extracted patterns for their subjective measures, as a measure of interestingness (Silberschatz & Tuzhilin, 1996). In the subjective (user-oriented) point of view, the extracted patterns can be classified as *unexpected* – a pattern which is "surprising" to the user- and *actionability* – a pattern that the user can act on it to his advantage. Therefore, actionable model can be considered as a result the involvement of users to afford important grounds to business decision makers. There was a highlight on the topic of actionable knowledge discovery that it will be one of the Grand Challenges for extant and future data mining (Ankerst, 2002; Fayyad, Shapiro, & Uthurusamy,

* Corresponding author.
E-mail addresses: bernardo@hufs.ac.kr (B.N. Yahya), mssong@postech.ac.kr (M. Song), hrbae@pnu.ac.kr (H. Bae), sosul@tsb.co.kr (S.-o. Sul), jzwu@scu.edu.tw (J.-Z. Wu).

http://dx.doi.org/10.1016/j.cie.2016.05.010
0360-8352/© 2016 Elsevier Ltd. All rights reserved.

2003). Since then, actionable knowledge discovery had been enhanced to satisfy the real business needs (Cao & Zhang, 2007; Cao et al., 2010; Yang, Yin, Ling, & Pan, 2007) because the real world problems are usually highly constraint-based and tightly embedded in domain-specific business rules. Nevertheless, the role of users, i.e., stakeholders, in process mining has been given little attention so far.

Process mining is presumed as an automated process to discover a model using algorithms and tools without human involvement. Process mining, which includes aspects such as discovery, conformance and enhancement, have been proved to solve business and industrial applications (Cho, Song, & Yoo, 2015; Goedertier, Weerdt, Martens, Vanthienen, & Baesens, 2011; Rebuge & Ferreira, 2012; van der Aalst et al., 2007; Wang, Caron, Vanthienen, Huang, & Guo, 2014). This situation partly results from the scenario that process mining is based on event logs where process mining algorithms extract patterns from data based on specific conditions, e.g. a log that has only start and end. Although a mined model can show patterns such as "hidden" and "unused" flow, it still remains some questions about how to extract specific patterns, i.e., actionable patterns. Some efforts have been considered to improve the mined model using proprietary filters. For example, the work on process mining of test processes in semiconductor manufacturing mentioned the necessity to apply various filtering techniques to show different level of abstractions (Rozinat, de Jong, Gunther, & van der Aalst, 2009). However, filtering the log requires some works, e.g., it is not interactive, and potentially loses the connection to the previous stage of filtering (Rozinat et al., 2009). In addition, the used of filters is intended to reduce incorrect or corrupted information in logs, i.e., noise. Since the nature of noise is assumed as either infrequent or incomplete information, the filter techniques may be insufficient to tackle a positive noise, e.g., purchasing a diamond that occurs infrequently but invaluable, that may provide business interestingness. As a result, actionable patterns, which are a part of business interestingness, are often neglected since they are hidden in large quantities of data. Thus, the involvement of users on process mining plays a key role for generating an actionable model.

In general, process discovery algorithms and techniques only focus on the discovery of model satisfying expected technical significance (Gunther & van der Aalst, 2007; Weijters, van der Aalst, & de Medeiros, 2006). Some process discovery techniques involve users to filter and configure the mined model (Bergenthum, Desel, Lorenz, & Mauser, 2007; van der Aalst, Weijters, & Maruster, 2004); however, the results remain unsound and incompatible. Other techniques related to the constraint-based approach, artificial generated negative events (AGNEs) (Goedertier, Martens, Vanthienen, & Baesens, 2009) for example, offer detailed descriptions of automatic generated artificial negative events rather than generating a model using domain experts' constraints. The work on process discovery via precedence constraints using constraint programming (Greco, Guzzo, & Pontieri, 2012) intended to involve users for discovering a model. However, the aspect of business interestingness, i.e., users' interactivity to build the constraints, had been neglected. Fundamental work on process discovery is therefore necessary to cater for critical elements in real-world applications such as expert knowledge. This is related to algorithm innovation and performance improvement that will balance technical interestingness and business interestingness.

This paper proposes a new discovery technique, called proximity miner, that is designed to discover an actionable process model with the involvement of domain experts. The main contributions of this work are developing an interactive mode process discovery tool with the involvement of users and demonstrating the effectiveness and flexibility of the proposed approach in tackling real-world process mining. In addition, this approach not only utilizes pre-processing tasks, i.e. using filtering and mining as a back-and-forth cycle procedure, but also applies post-analysis that enables users to refine mined model with a set of constraints for generating actionable process model. Actionable process model in this paper is a process model that satisfies the constraints from a domain-specific expert in terms of their value in decision making.

To some extent, the objective of this approach is similar to that of process discovery via precedence constraints, however, its strategy and contributions differ in five ways. First, it introduces an integer linear programming (ILP)-based *domain-constraints* approach that entails less computation than constraint programming (Salvagnin, 2008). Second, it presents the concept of *categorization*, which is required when the log consists of several event types. Third, it introduces *proximity*, which guarantees the inclusion of both the observed and the "hidden" behavior of the event logs in our ILP model. The use of *proximity* ensures the production of a sound model when users' background knowledge is not available ("hidden") from the log-derived graph. Fourth, this approach uses an *interactive* interface that allows users to iteratively set the constraints for refining the control-flow factors. Fifth and finally, it can be considered *an actionable model builder* that does not require either any effort in drawing from scratch or back-and-forth cycle procedure between filtering and mining tasks. The proposed approach was tested by means of a case study conducted in the port logistics process domain, and was verified on the basis of quantitative and qualitative evaluations.

The remainder of this paper is organized as follows. Section 2 discusses the relevant process mining work available in the literature. Sections 3 and 4 present a running example of the proposed discovery algorithm and the overall methodology, respectively, in demonstrating how the ILP is combined with the concept of proximity. Section 5 analyzes the main features of our approach, its implementation in the ProM framework, and the evaluation results. Section 6 concludes the paper.

2. Related work

2.1. Process mining

Process mining is a recognized and well-known technique for mining process models from logs. It includes classes such as discovery, conformance and enhancement. Various approaches to the construction of process models from logs (i.e. process discovery) and the measuring of discrepancy between logs and a given predefined process model (i.e. conformance testing) have been developed over the past few years (de Medeiros, Weijters, & van der Aalst, 2007; Gunther & van der Aalst, 2007; Kawalek & Kueng, 1997; Rozinat, de Medeiros, Gunther, Weitjers, & van der Aalst, 2007; van der Aalst, 2011; van der Aalst et al., 2004; Weijters et al., 2006). Enhancement, which also needs a prior model, is used to enrich the mined model based on some performance data (van der Aalst, 2011). The concept has been applied to some industries such as health, logistics and telecommunications (Cho et al., 2015; Goedertier et al., 2011; Rebuge & Ferreira, 2012; van der Aalst et al., 2007; Wang et al., 2014).

However, most of the applicability-related work has been concerned with the general concept of process mining, with little focus on domains. This means that when analysts are eager to analyze a domain-specific problem, domain experts must be involved to interpret the result. Some approaches to cover specific behaviors in a process in terms of particular flows have been proposed, such as non-free choice (Wen, van der Aalst, & Wang, 2007), size of discovery problems (Carmona & Cortadella, 2014), probabilistic analysis (Cook, Du, & Liu, 2004), control dependencies with conditions (Hwang & Yang, 2002), activities' lifespans (Pinter & Golani, 2004),

the decomposition approach (van der Aalst, 2013) and the light region-based approach (Carmona, 2012; Solé & Carmona, 2011). None of these previous studies considered the role of domain experts in process model discovery.

Some previous work has considered *apriori* knowledge setting, however, little attention has been taken toward actionable process model discovery. Using the traditional process mining techniques, two approaches, specifically alpha mining (van der Aalst et al., 2004) and region-based mining (Bergenthum et al., 2007), allow users' interventions to improve the quality of mined models. Unfortunately, since they use manual techniques, their modifications to reflect user requirements are error-prone and, in some aspects, unsound. For example, when a user sets two events as parallel, it results a connectedness problem (some activities may not reach finish properly). Other approach such as alpha+ miner (de Medeiros, van Dongen, van der Aalst, & Weijters, 2004) could result better than the previous approaches. However, the model still could not capture the business interestingness. The current approaches, including users' recommendations (Barba, Weber, Valle, & Jimenez-Ramirez, 2013), semantic log-purging (Ly, Indiono, Mangler, & Rinderle-Ma, 2012), artificial generated negative events (AGNEs) (Goedertier et al., 2009) and process discovery via precedence constraints (Greco et al., 2012) were designed to compensate for the limitations of the traditional techniques. Users' recommendations allow for their intervention in process execution optimization. Semantic log-purging applies expert knowledge as a semantic constraint by purging constraint-violating events from the log. Both users' recommendations and semantic log-purging differ from our approach in terms of process discovery. AGNEs, meanwhile, offers detailed descriptions of generated artificial negative events (i.e. negated events) rather than generating constraints. The generated negated events, which also include log noise, possibly lead to different outcomes and sometimes false conclusions. As for the work on process discovery via precedence constraints, it is quite similar to the present approach in some aspects. However, it included no discussion on the soundness property (e.g. an expert provides the desired constraints, which lead to the discovery of an unsound process model), which is the most important in the process discovery domain. Moreover, other issues such as additional constraints (i.e. designated start, designated end), which are critical to process discovery, and interface for the convenience of users have not been investigated in detail. The present study attempted to resolve the limitations of the previous work and enhance it by creating a support for critical issues of actionable process model discovery (i.e., soundness and constraints builder) and building interactive interfaces for users' convenience. Rather than receiving a collection of constraints once at a time, this approach allows users to iteratively set the constraints to refine the model.

There has already been a study on process mining using ILP (van der Werf, van Dongen, van Hee, Hurkens, & Serebrenik, 2009). ILP Miner had been proposed as a means of process model discovery based on the optimization problem. However, it focused on a scheme of process model extraction whereby all possible orderings of events must be satisfied under log behavior constraints. It thus differs from the proposed approach's expert-knowledge and soundness-property-based constraints. This method, as compared with that which uses a constraint programming problem, is expected to reduce the time taken to discover a model (Salvagnin, 2008).

With the proposed approach, additionally, an actionable process model, instead of a reference model, can be built. There are various works in terms of building reference model based on process model variants (Li, Reichert, & Wombacher, 2011; Yahya, Bae, Bae, & Kim, 2012a; Yahya, Wu, & Bae, 2012b). The previous work (Yahya, Bae, Bae, & Liu, 2012c) had proposed the proximity score to automatically generate a reference model. And in terms of process mining, most studies have used configurable reference (process) models to derive individual process variants (Gottschalk, van der Aalst, & Jansen-Vullers, 2008; La Rosa, Gottschalk, Dumas, & van der Aalst, 2008; van der Aalst et al., 2008). The proposed study, by contrast, employed an ILP approach to generate a *sound* process model and reutilized some of the measures in proximity score (Yahya et al., 2012c). In order to obtain more comprehensive results in comparison with the previous work (Yahya, Bae, Sul, & Wu, 2013), the technique focuses more on two salient features, comprehensive evaluation by both process analysts and domain experts and automatic categorization of multiple event types. With regard to the result, the present study result in the form of a graph is closer to that of the graph-based mining approaches (Agrawal, Gunopulos, & Leymann, 1998; Hwang & Yang, 2002).

2.2. Actionable knowledge discovery

Actionable knowledge discovery, an origin of actionable process model discovery, has been popular since it was first highlighted as a future data mining topic (Ankerst, 2002; Fayyad et al., 2003). As a matter of fact, the term *actionability* mainly measures the ability to suggest business decision-making actions, and it has been enhanced to satisfy real business needs (Cao & Zhang, 2007; Cao et al., 2010; Yang et al., 2007). It also has been pointed out that the involvement of domain experts plays a key role in knowledge discovery, particularly in the extraction of interestingness (Yoon, Henschen, Park, & Makki, 1999). Actionable knowledge discovery relates to beliefs that are obtained from interestingness measures. The interestingness measures are mostly related to objective measures, with less consideration of subjective measures. Taking account of subjective interestingness is to personalize the discovered model, in which a pattern of interest to one user might be of no interest to another. The pattern is either *actionable* or *unexpected*.

An *actionable* pattern is interesting because the user can do something about it; that is, the user can react to it to his or her advantage. To use another word, *actionability* is an important subjective measure of interestingness, because users are mostly interested in the newly discovered knowledge that permits them to do their jobs better by taking some specific actions in response to it (Bie, 2013; Liu, Hsu, Chen, & Ma, 2000). On the other hand, an *unexpected* pattern is surprising to the user; and it is certainly interesting, then, when the user can do something about it. Otherwise, the pattern contradicts the user's existing knowledge, and consequently holds less interest for that user. That is, the judging of unexpected patterns to be actionable or not depends on the user's system of beliefs. Thus, the interestingness rules fall into three categories: rules that are unexpected but actionable, those that are unexpected but not actionable, and those that are expected and actionable (Liu et al., 2000). Even though Silberschatz and Tuzhilin (1995) mentioned that unexpectedness and actionability are, in general, independent of each other, both of them are considered to fall between the aforementioned three categories. The users' role (which has been given little attention so far) in determining an actionable process model that provides a benefit for business decision making is therefore significant in this study. This study attempted to apply the three categories to some rules as the prior knowledge of domain experts.

3. Running example

In this study, we used the port logistics process of landside transport. In order to briefly explain the port logistics process, we define nine main activities as listed at Table 1. *Discharging by quay crane* activity followed by *discharging by yard crane* activity

Table 1
Nine main activities in port logistics process with index.

Activity name (abbreviation)	Index
Discharging by quay crane (QD)	A
Discharging by yard crane (YD)	B
Yard crane task for Gate Out (YO)	C
Yard crane task for Gate In (YI)	D
Loading by yard crane (YL)	E
Loading by quay crane (QL)	F
Remarshalling PickUp (RP)	G
Remarshalling Stack (RS)	H
Shuffling (S)	I

describes the process of discharging a container from a vessel using a quay crane and moving it into the yard using a yard crane. If a scheduled truck is coming into the port, the container should be ready for delivery. In any case, the container stays in the yard and waits for *Yard crane task for Gate Out*. This is a typical example of the landside discharging process.

Yard crane task for Gate In activity describes the movement of an incoming container as it proceeds into the port from outside. If a vessel is berthed at the port, the container should be ready for loading activity. So, the container will flow from *Yard crane task for Gate In* to *Loading by yard crane*, and will then proceed to the vessel via the *Loading by quay crane* activity. This is a typical example of the landside loading process.

There might be additional activities for relocation of the container due to schedule changes or process optimization: *remarshalling* and *shuffling*. Remarshalling refers to the task of rearranging export containers scattered around within a block into designated target bays of the same block. It consists of two steps: *remarshalling pickup* and *remarshalling stack*. Meanwhile, *shuffling* refers to the repositioning of a container to another storage location due to an inability to access other containers that are stored below it. Minimal occurrence of these two activities is preferable.

To illustrate the necessity of our approach, we use the sample log in Table 2. All dependency from the traces in Table 2 can be seen in Fig. 1. Suppose that there are 9 distinct traces among 5170 cases. Let L be an event log. Event log L is a multiset of traces, and each trace is a sequence of activities. Thus, a trace ⟨A, B, I, C⟩ denotes a sequence that is started with A, followed by B and I, and ended with C. The frequency of each trace is represented as a superscript. For example, $\langle A, B, I, C\rangle^{3000}$ means that there are 3000 cases of that trace. We are interested in finding actionable patterns that often are neglected by traditional process mining approaches. For example, low-frequency behavior such as $\langle A, F\rangle^{9}$ fails to be a pattern extracted by heuristic miner with the default parameters. However, from the perspective of domain experts, this pattern is actionable in the real world. For example, a container that has been discharged from a vessel through a quay crane will soon be allocated to another vessel for export. Thus, the container will be stored for loading by the nearest quay crane. When we consider only the occurrence of this pattern, it is almost impossible to include this pattern in the model. However, other traces such as $\langle A, B, E, F\rangle^{800}$ can reflect that A and F have a closeness (*proximity*) to some degree. Thus, a measure is needed to increase the confidence to include the pattern in the process model as an actionable pattern instead of omitting it due to low-frequency occurrence. In this sense, this pattern is unexpected but actionable.

Table 2
Fragment of traces with number of cases.

Traces	Number of cases
⟨A, B, I, C⟩	3000
⟨A, B, I⟩	1000
⟨A, B, E, F⟩	800
⟨A, B, I, I, I,C⟩	150
⟨A, B, I, I⟩	150
⟨A, B, G, H, G, H, C⟩	50
⟨D, G, H, E, F⟩	10
⟨A, F⟩	9
⟨G, H⟩	1

Other examples are related to activities I, G and H. Domain experts' belief suggests that there should be no iteration of activity I or of G or H. In fact, activity I is in the loop pattern in the traces $\langle A, B, I, I, I, C\rangle^{150}$ and $\langle A, B, I, I\rangle^{150}$. Since $\langle A, B, I, C\rangle^{3000}$ and $\langle A, B, I\rangle^{1000}$ dominate the other two, we can intuitively say that the loop is likely to be not actionable. As well as activity I, the pattern of the relation between G and H in the traces $\langle A, B, G, H, G, H, C\rangle^{50}$ and $\langle D, G, H, E, F\rangle^{10}$ dominates the trace $\langle G, H\rangle^{1}$. In other words, allocating activity G as a start activity and activity H as an end activity is not expected. Thus, it can be categorized as unexpected and not actionable.

This proposed study undertook to answer the question, "Does the process flow correctly?"; that is, based on event logs, we automatically constructed a process model showing the ordering and frequency of flows. Subsequently, we formulated a proposed new approach in order to answer the question, "How do we obtain an actionable process model based on user's knowledge?"

4. Proposed methodology

This section explains the proposed method. First, formal definitions are provided. Next, the ILP-based mathematical programming is described.

4.1. Formal definitions

This section presents the notations used in this paper. First, the definition of events is given. Second, the causality graph definition, as the result of this study, is given. Finally, the approach and domain knowledge definitions are described.

Definition 1 (*Event log (L)*). An event log is a multiset of traces. Each trace is mapped into a case and consists of a collection of events, which is defined as follows:

Event. E is a set of events and $E = A \times Y \times T$ where A is a set of activities, T is a set of timestamps and Y is a set of event types. It should be noted that although the event types are not explained in the earlier example, it will be used in the later section. To represent the activity, type and timestamp of each event, we use the following notation: *e.act* refers to the activity

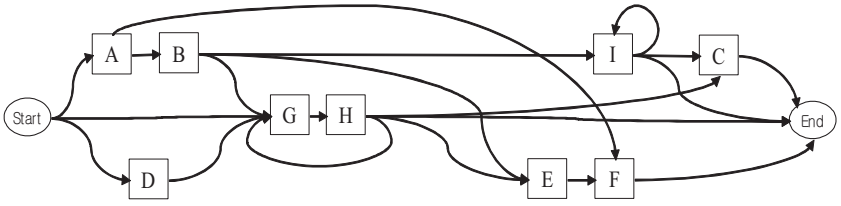


Fig. 1. Process model extracted based on traces in Table 2 with artificial start and end.

name, *e.type* to the event type, and *e.time* to the event timestamp. If e = (YardJobLoad, COMPLETE, 2012-06-12 01:54:32), then *e.act* = YardJobLoad, *e.type* = COMPLETE, and *e.time* = 2012-06-12 01:54:32.

Case. C is a set of events for its instance, called as case, and $C = \{c_k | k = 1, \ldots, K\}$. A case c_k corresponds to the trace σ_k, $\langle e_1^k, e_2^k, \ldots, e_n^k \rangle$ where each e_i^k denotes the i-th event in case k. $e_i^k \in \boldsymbol{E}$ is an event in a single workflow instance for $1 < i < I^k$, and I^k is the total number of events in the k-th case. The traces are the sequences of events that indicate the activity flow from beginning to end. There might be a trace with multiple occurrences. Thus, a superscript number on each trace represents the frequency of the trace. For example, event log L = [⟨A, B, I, C⟩^{3000}, ⟨A, B, I⟩^{1000}, ⟨A, B, E, F⟩^{800}, ⟨A, B, I, I, I, C⟩^{150}, ⟨A, B, I, I⟩^{150}, ⟨A, B, G, H, G, H, C⟩^{50}, ⟨D, G, H, E, F⟩^{10}, ⟨A, F⟩^{9}, ⟨G,H⟩^{1}].

The causal relations, which represent adjacent relationship, between events can be represented in a dependency graph.

Definition 2 (*Dependency Graph*). A dependency graph is a tuple of $G = (V, ED)$, which is defined as follows:

- $V \subseteq A \times Y$ is set of nodes (vertices) which represent a product of activity and event type.
- $ED \subseteq V \times V$ is a set of edges where two events are adjacent. Suppose, $ed_{ij}^k = \{(v_i, v_j) | v_i,\ v_j \in \boldsymbol{V}\}$, to represent $e_i^k \succ e_j^k$, where e_i^k is the immediate predecessor of e_j^k (e_j^k is the immediate successor of e_i^k) with no event $e_m^k \in E(L)$, such that $e_i^k \not\succ e_m^k$ and $e_m^k \not\succ e_j^k$ and $e_i^k.time < e_j^k.time$. Thus, two events are having *causal relations* from e_i^k to e_j^k and represented as a causal matrix F, where $F_{ij} \in \boldsymbol{F}$ is the number of events $e_i^k \succ e_j^k$ for all k in L.
- v_0 represents as a start node if there exist $v_i \in \boldsymbol{V}$ such that $\exists_k ed_{i0}^k = \varnothing$.
- v_N represents as an end node if there exists $v_j \in \boldsymbol{V}$ such that $\exists_k ed_{Nj}^k = \varnothing$.

For example, an event log L = [⟨A, B, I, C⟩^{3000}, ⟨A, B, E, F⟩^{800}, ⟨A, F⟩^{9}] can be projected into a dependency graph with the set of nodes ($\boldsymbol{V}$) are {A, B, C, E, F, I} and the set of edges are {(A, B), (B, I), (I, C), (B, E), (E, F), (A, F)} with A = v_0 and C, F = v_N representing the start and end nodes, respectively. Thus, the causal matrix $\boldsymbol{F}$ represents the causal relations between two nodes, which are the elements of the set of edges. For example, a relation between B and I (i.e. F_{BI} = 3000) shows as the number of cases that node B occurs as the immediate predecessor of node I. In other cases, there is a relation between B and E (i.e. F_{BE} = 800). The choice between either executing I or E represents a relation of exclusive OR (XOR-split).

Suppose, the event log L_M = [⟨M1, M2, M3, M4⟩^{100}, ⟨M1, M3, M2, M4⟩^{90}, ⟨M1, M5, M4⟩^{10}] which each index represents as follows; M1: Check Financial, M2: Check Loading certificate, M3: Check Unloading certificate, M4: Build Work Schedule, M5: Inspect the certificate. After executing of the first task M1, there is a choice between either executing M2 or M3 concurrently (i.e., in parallel or in any order) or just executing activity M5. The execution of both M2 and M3 in parallel is considered as AND relationship. Mining the AND relationship is difficult since the relationship is implicit in the event log. The traces already give the information that activities M2, M3 and M5 are the successor of activity M1. If two activities (e.g. M2 and M5) are in the AND relationship, the pattern ⟨…M2, M5…⟩ can appear in the event log. If two activities (e.g. M2 and M5) are in the XOR relationship, the pattern ⟨…M2, M5…⟩ is not possible. As aforementioned, I and E are in the XOR relationship. Based on the traces in the event log (refer to Table 2), there is no cases which have patterns such as either ⟨…I, E, …⟩ or ⟨…E, I, …⟩. LOOP relationship is the possibility to execute the same activity multiple times. A distance loop, for example the traces ⟨A, B, G, H, G, H, C⟩, and length-one loop, for example ⟨A, B, I, I⟩ are two possible LOOP relationships extracted from event log.

To mine such AND relationship, it needs a particular mechanism to extract the (hidden) pattern. The mechanism is adopted from heuristic miner to find the relationships of the two behaviors that are above a particular threshold (Weitjers et al., 2006). Suppose that $ed_{ij}^k = (v_i, v_j) | v_i,\ v_j \in V$ and $e_i^k.act = v_i.act$, $e_i^k.type = v_i.type$ and $e_j^k.act = v_j.act$, $e_j^k.type = v_j.type$ and $e_i^k, e_j^k \in E(L)$. Each behavior can be described as follows:

- AND behavior. There can be AND behavior such that $e_i^k \succ e_j^k \succ e_m^k$ appears in trace k, and $e_i^l \succ e_m^l \succ e_j^l$ in trace l to some degree. Thus, two nodes have AND behavior if and only if $(F_{jm} + F_{mj})/(F_{ij} + F_{im} + 1) > \Omega$, Ω is a threshold, and $(0, 1] \in \Omega$. That is, AND behavior exists on the model if and only if $F_{jm} > 0$ and $F_{mj} > 0$ and there exist node v_i such that $F_{ij} > 0$ and $F_{im} > 0$. For example, using the event log L_M, the value of the measure of M2 and M3 equals to $(100 + 90)/(100 + 90 + 1) = 0.9947$. If the value is higher than specified threshold (e.g., 0.5), then the two nodes (e.g., M2 and M3) are in the AND relationship. Two nodes could be in the AND relationship when both of the nodes have the same immediate predecessor node. Intuitively, it has path from immediate predecessor node to those two. If the value of relation of two nodes does not satisfy the conditions (e.g., greater than 0.5), it is an XOR relationship.
- LOOP behavior. There can be loop behavior such that $e_i^k \succ e_j^k$ and $e_i^k.act = e_j^k.act$ and $e_i^k.type = e_j^k.type$ appears in a trace. This loop is considered as a length-one loop, and F_{ii} is greater than 0. For a distance loop, it can easily be discovered when there is a distance that can be measured by proximity between two events, as will be explained in the later section.

Since an event can contain transactional information, the dependency graph can provide different result due to additional information on the causal relations. For example, an activity can have more than one event type, e.g., *start* and *complete*. The dependency graph can show misleading result to practitioners when there is no clear indication of the respective activity. A merging function, denoted as categorization, is necessary to perceive the process in regard to the activity.

Let $\boldsymbol{E} = \boldsymbol{A} \times \boldsymbol{Y} \times \boldsymbol{T}$ be a set of events over $\boldsymbol{A}$ and $\sigma = \langle e_1, e_2, \ldots, e_n \rangle$ an event trace and $\boldsymbol{Y} = \{0, 1\}$. σ is able to be categorized if and only if

1. $\forall\ e_i \in \boldsymbol{E} \wedge e_i.type = 0 \Rightarrow \exists e_j \in \boldsymbol{E} |\ e_j.type = 1 \wedge e_i.act = e_j.act \wedge j > i$, i.e. every START event has a corresponding COMPLETE event.
2. $\forall\ e_i \in \boldsymbol{E} \wedge e_i.type = 1 \Rightarrow \exists e_j \in \boldsymbol{E} |\ e_j.type = 0 \wedge e_i.act = e_j.act \wedge j < i$, i.e. every COMPLETE event has a corresponding START event.

Definition 3 (*Categorization*). Categorization is defined as the organization of events according to the activity name.

Let $e_1, e_2, e_3, e_4 \in \boldsymbol{E}$ where $e_1.act = e_3.act$ and $e_2.act = e_4.act$ and $e_1.type = e_2.type$ and $e_3.type = e_4.type$. Suppose $a_1, a_2 \in \boldsymbol{A}$ and $\{0, 1\} \in \boldsymbol{Y}$ and $v_1, v_2, v_3, v_4 \in \boldsymbol{V}$. By projecting each event into nodes, we get $v_1 = (a_1, 0)$, $v_2 = (a_2, 0)$, $v_3 = (a_1, 1)$ and $v_4 = (a_2, 1)$. The merging function is to group the same activity of different types regardless of the event sequence. For example, suppose there is a trace $\langle e_1, e_2, e_3, e_4 \rangle$. The sequence of the dependency graph follows the trace sequence while categorization attempt to highlight the activity instead of nodes. Suppose $G' \subseteq G$. Thus, the merging

function generates graph G' with four nodes $\{v_1, v_2, v_3, v_4\}$ and highlight the merge function result that is the set of precedence relationship of two events of same activity with different event types, $\{(v_1, v_3), (v_2, v_4)\}$. It should be noted that categorization, which is used to express the process flow on the basis of activity, is particularly for better visualization.

When an event that is not a start event is reachable from another event, we can show the relationship in terms of proximity. But before proximity can be measured, first we must define the terms traceable event and distance.

Definition 4 (*Traceable Event*). A traceable event is a subset of events such that a given event is reachable from another event. An event that is not a start activity should have a set of traceable events defined as follows.

Let L be an event log. Traceable event set (R) is denoted as $r_i \subseteq \{(e_i^k, e_j^k | e_i^k \gg e_j^k,\ \forall e_i^k, e_j^k \in E(L)\}$, where $e_i^k \gg e_j^k$ means that e_j^k is reachable from e_i^k in the k-th trace (i.e. e_i^k can reach e_j^k). The element $(e_i^k \gg e_j^k)$ represents the fact that e_i^k precedes e_j^k at certain distances, which is to say, it is not an immediate predecessor. When there exist $e_i^k \gg e_j^k$ and $e_i^k.act = e_j^k.act$ and $e_i^k.type = e_j^k.type$ then it considers as length-n loop which n represents the distance between two events.

To indicate the traceable events, some measures are needed. This study applies distance and proximity measures. The proximity of both events can be measured according to the distances.

Definition 5 (*Distance*). Distance (d_{ij}^k) is defined as an integer value indicating that two events e_i^k and e_j^k are either immediate predecessor (=1) or are between some other events (>1) (Yahya et al., 2012c). Distance between two events can be derived from a path. A path from a to b is denoted as a list of events, $\langle e_1^k, e_2^k, \ldots, e_{n-1}^k, e_n^k \rangle$ with $n > 1$ such that $e_1^k = a$ and $e_n^k = b$ and $\forall_{i \in \{1,\ldots,n-1\}}\ (e - i^k \succ e_{i+1}^k)$. Suppose that pa_{ij}^k is denoted as a sub-path from an event e_i^k to another event e_j^k in a case k, and that $|pa_{ij}^k|$ is the number of events in the sub-path.

- $pa_{ij}^k = \{e_i^k, e_{i+1}^k, e_{i+2}^k, \ldots, e_{j-1}^k, e_j^k\}$, $(e_i^k \succ e_{i+1}^k)$, $(e_{i+1}^k \succ e_{i+2}^k)$, $(e_{j-1}^k \succ e_j^k)$, $\forall e \in \sigma_k$

Hence, the distance between two events, e_i^k and e_j^k, and $1 \leqslant i < j \leqslant n$ is

$$d_{ij}^k = |pa_{ij}^k| - 1 \tag{1}$$

From the traces in the event log, we can measure the proximity based on the definitions of dependency graph, traceable event, and distance.

Definition 6 (*Proximity*). Proximity is defined as the closeness of two activities. Using the concept of causal relations, two events that exist in a trace have such closeness, and the causality with h_{ij}^k equals 1. This means that the distance of two events, e_i and e_j, with $d_{ij}^k > 0$, has the causality $h_{ij}^k = 1$, otherwise 0. When $d_{ij}^k = 1$, we say that the two events have a causal relation and that the occurrence of causal relations, F_{ij}, is incremented. However, when $d_{ij}^k > 1$, the two events are included in traceable events, and there is no increment of F_{ij}. To indicate the closeness, a proximity value is used. Proximity value $q_{ij}^k q_{ij}^k$ is denoted as in the following Eq. (2).

$$q_{ij}^k = \frac{h_{ij}^k}{d_{ij}^k} \tag{2}$$

To determine the overall proximity score (PS_{ij}), it is measured as in the following Eq. (3),

$$PS_{ij} = \frac{\sum_{k=1}^{K} q_{ij}^k}{|H_{ij}| + 1} \tag{3}$$

where $|H_{ij}|$ is the number of cases when $h_{ij}^k = 1$ for all k . Instead of proximity, PS_{ij} can be considered as the value of confidence within a certain threshold.

For example, let us consider an event log $L = [\langle A, B, I, C\rangle^{3000}, \langle A, B, E, F\rangle^{800}, \langle A, F\rangle^{9}]$. The distance of A and B in each trace is 1. Therefore, the causal matrix F for A and B is $F_{AB} = 3800$, and the overall proximity score is 0.9997. In terms of events A and F, the relationship consists of both an immediate predecessor and traceable events. Thus, the proximity between the two should be measured. For the events on which A and F are located as immediate predecessors, it updates the causal matrix F; that is, $F_{AF} = 9$. In the trace $\langle A, B, E, F\rangle$, the proximity value between A and F is 1/3 = 0.33, since $h_{AF}^k = 1$ and $d_{AF}^k = 3$. The overall proximity score for events A and F is $(0.33 * 800) + (1 * 9)/(800 + 9 + 1) = 0.337$. To obtain an actionable model using the proximity score, we need to ensure that those two events have a causal relationship. For example, in one case, events B and F have a causal relationship; their proximity score, therefore, is $(0.5 * 800) + (1 * 1)/(800 + 1 + 1) = 0.5$. If we want to find the actionable relation from the log with a confidence of 0.3, then both the relations A − F and B − F are included in the discovered model.

The overall proximity score is similar to transitive closure measures. In this study, we adopted the Floyd–Warshall algorithm and modified it to correspond to the data properties. The detailed procedure is as follows.

```
OverallProximity(Log L)
1 int distance, i, j, occur, occurrence;
2 double[][] d, q;
3 Matrix F, PS;
4 Map⟨key, value⟩ H;
5 foreach p_k ∈ L do{  //each case/process instance
6  foreach e_i ∈ p_k do{  // each event in a case
7    i ← index of e_i
8    j ← index of e_{i+1}
9    int occur = 1;
10     IF (F(i,j) exist) {
11       occurrence ← retrieve the value of F(i,j);
12       occur ← occurrence + 1;}
13   F(i,j) ← put occur;
14   for each e_j ∈ p_k and i < j do { //closure events
  in a case
15       j ← index of e_j;
16       d(i,j) ← j − i;
17       q(i,j) ← 1/distance;
18       PS(i,j) ← current value of PS(i,j) + q(i,
  j);
19       Let the value of H be added by 1 for the key
  i, j;
20   } //end foreach
21  } //end foreach
22} //end foreach
23
24 foreach item i and j in PS do{
25   double val ← value of PS, vals ← value of H;
26   update value of PS ← val/vals; //
27}
28 return PS;
```

The line numbers 1–4 are variable declarations for the algorithm. In lines 5–22, it runs for each case. This means that the traces are analyzed individually. First, the causal relations matrix is measured for variable $F(i,j)$. The measure starts to collect the index of each event (lines 7–8). Line 9 denotes a variable input to represent the occurrence of adjacency. If there exists a relationship between i and j (line 10), the value is retrieved from matrix $F(i,j)$ and the variable *occur* is updated by adding *occurrence* with the value of 1. Next, matrix $F(i,j)$ is updated (line 13). If there is no pre-existing relationship, the variable *occur* remains 1. After we obtain the adjacency of events, we want to measure the closure relations of events. By recursively finding the events in the same case as $i < j$ (line 15), the overall proximity score is calculated, starting with the subtraction of the index (line 16) followed by the measurement of the proximity (line 17) and ending with the updating of the overall proximity score (line 18). In this step, the overall proximity score is the summation of all proximity scores. Finally, the overall proximity score is updated by dividing by the number of cases, $|H|$ (lines 24–27). The running time of the overall proximity score is $O(kn^2)$, where k is the number of cases in log L and n is the maximum number of events in the existing cases.

According to the prior domain knowledge given by experts, there could be either unknown or additional relations that lead to problems such as soundness. Soundness aims to verify the model correctness; it means that all of the dependencies with respect to L exist in G; that is, for every causal relation with respect to L, there is a corresponding path in G. The overall proximity score attempts to verify the unsound relationship and ensure the building of a sound model. Thus, it is necessary to understand the soundness of the process model.

Definition 7 (*Soundness*). Soundness is considered as a correctness criterion of a process model (van der Aalst, 2011). A dependency graph is *sound* if

a. Any nodes excluding an end node have at least an immediate successor and a valid sequence until an end node.

A valid sequence means a flow that follows the traceable event (s). This condition requires at least a node follows a start node and ensures that the flow reaches the final state, i.e., end node. In other words, any activity that is not the end node should have, at least, an outgoing flow for generating an immediate successor.

b. Any nodes excluding a start activity have at least an immediate predecessor and a valid sequence from a start node.

A valid sequence means a flow that follows the traceable event (s). This condition requires a flow exist from any nodes to the end node. In this case, any node that is not start node should have, at least, an incoming flow for instantiating the immediate predecessors.

The two conditions enforce the model to reach the final state. In other words, the process model should follow a mandatory completion flow (Trcka, van der Aalst, & Sidorova, 2009).

All the definitions above support the domain knowledge. The domain knowledge, either given prior to process discovery or post-analysis, is formalized as follows.

Definition 8 (*Domain Knowledge (D)*). Domain knowledge aims to retrieve any causalities; such knowledge falls into one of three categories: rules that are both unexpected and actionable (UA), those that are unexpected but not actionable (UA), and rules that are actionable but expected (AE). The adoption of these terms (Liu et al., 2000), which later will be called domain knowledge, can be obtained either as *a priori* knowledge or post-analysis. Let x_{ij} be the decision variable of dependency between two nodes, node i and j. Domain knowledge in the form of actionable rules affects the decision of dependency between two nodes (x_{ij}) and is interpreted by means of graphs as follows.

- *Mandatory causality.* Node v_i has mandatory causality with v_j if v_i is the immediate predecessor of v_j. It is denoted as *MC*, where $MC \subseteq \{mc_{ij} = (v_i, v_j) \mid v_i,\ v_j \in V\}$ is the set of two vertices where $v_i\ v_j$; that is, v_i is the immediate predecessor of v_j. ($x_{ij} = 1$).
- *Indifferent causality.* Node v_i has *indifferent* causality with v_j if v_j is not *the* immediate successor of v_i. It is denoted as *IC*, where $IC \subseteq \{ic_{ij} = (v_i, v_j) \mid v_i,\ v_j \in V\}$ is the set of two vertices where $v_i \not\prec v_j$; that is, v_i is not the immediate predecessor of v_j ($x_{ij} = 0$).
- *Concurrent activities.* Node v_i has *interleaving* causality with v_j if v_i and v_j are concurrent activities. It is denoted as *CA*, where $CA \subseteq \{ca_{ij} = (v_i, v_j) \mid v_i,\ v_j \in V\}$ is the set of two *nodes* where ca_{ij} *indicates* that v_i appears before v_j in one execution and *that* v_i *appears after* v_j *in the other execution.* Thus, there is no relation between *the* two vertices v_i and v_j such as $x_{ij} = 0$ and $x_{ji} = 0$.
- *Designated Start.* Designated start (*ds*) is an activity assigned *as* a start activity. Hence, $ds = v_i \mid v_i = v_0 \wedge v_i,\ v_0 \in V$.
- *Designated End.* Designated end (*de*) is an activity assigned *as* an end activity. Hence, $de = v_i \mid v_i = v_N \wedge v_i,\ v_N \in V$.

Since the concept of proximity includes both traceable and direct events, all events will have a positive value of proximity score. The relevant behaviors are explicitly extracted from the logs. However, the irrelevant behaviors, which can only be derived from users' knowledge, can be considered to be relevant when the proximity score is high, for example, when there are more traceable events than direct events. Hence, it is necessary to prioritize the relevant behaviors by assigning benefit values. Since the irrelevant behaviors are less important than the relevant behaviors, they should be assigned a penalty value. Here, we introduce benefit and penalty as a trade-off score applied to obtain the required process model.

Definition 9 (*Benefit and Penalty*). Benefit (B_{ij}) and penalty (P_{ij}) score are introduced to represent the priority in the selection of extra behavior. The score to select the best direct event is necessary in two ways. First, it needs to choose the best direct event with the highest proximity (*benefit*). Second, it is required not to choose irrelevant behaviors (i.e. there is no direct event relation in the log), by imposing a *penalty*.

The user can choose the parallel, causality or not-related relationship according to their knowledge. Hence, when the given constraints generate an unsound process model, measures indicating the best relations are needed. If there exists a link between events i and j in the event log, the maximum *benefit* value of PS_{ij} and the 0 *penalty* value need to be assigned (since it is our intention to create the process model). On the other hand, we assign a 0 *benefit* value and the maximum *penalty* value to minimize the number of event relations that do not exist in the link. The initial *benefit* and the *penalty* values for all relationships are shown in Eqs. (4) and (5), respectively:

$$B_{ij} = \begin{cases} \max\limits_{i,j}(PS_{ij}) & \forall v_i, v_j \wedge F_{ij} > 0 \\ 0 & \forall v_i, v_j \wedge F_{ij} = 0 \end{cases} \tag{4}$$

$$P_{ij} = \begin{cases} \max\limits_{i,j}(PS_{ij}) & \forall v_i, v_j \wedge F_{ij} = 0 \\ 0 & \forall v_i, v_j \wedge F_{ij} > 0 \end{cases}. \tag{5}$$

For example, the possible maximum value of PS is 0.99. Suppose the maximum of PS is 0.99. The value of B_{ij} equals to 0.99 when $F_{ij} > 0$ or the value of P_{ij} equals to 0.99 when $F_{ij} = 0$.

By using soundness, it attempts to keep the dependency in G from any unknown relations from domain knowledge. It means, if an edge is removed according to domain knowledge, then there is a procedure – the one that is responsible for finding another best edge – to keep the soundness and compensate for the removed edge. To formalize the hidden relationship because of edge removal, we adopt integer linear programming that will be explained in the next section.

4.2. Integer linear programming

This section describe the model of integer linear programming due to the existing of domain knowledge (D) which can lead to unsoundness problem, e.g., some intermediate nodes have no either predecessors or successors. The objective function attempts to find the maximum value of proximity to mine a dependency graph from a log (Eq. (6)). If event j is the immediate successor of events i, it is obvious that the value of F_{ij} is greater than 0 ($F_{ij} > 0$, $B_{ij} > 0$, $P_{ij} = 0$). If there is no adjacent relation between events i and j, there are two options.

- First, for event i, event j is not immediate successor but it is a traceable event. Hence, the proximity score can have a greater value than 0 ($PS_{ij} > 0$, $B_{ij} = 0$, $P_{ij} > 0$), since the two activities are reachable in the event logs.
- Second, for event i, event j is neither immediate successor nor traceable event. Hence, there is no proximity score ($PS_{ij} = 0$, $B_{ij} = 0$, $P_{ij} > 0$), which means that there is no traceable event in the logs; therefore, the maximum proximity score for all events will be returned as a *penalty*.

In order to maximize W, the value of P_{ij} should be minimized. To discover a process model with the minimum penalty ($P_{ij} = 0$), that model should contain only nodes in causal matrix ($F_{ij} > 0$). The value of the *penalty* will be equal to $\max_{ij}(PS_{ij})$ when a behavior contradicts the existing log behavior, that is, when event j is not the immediate successor of event i such that $F_{ij} = 0$. In the case that P_{ij} is equal to $\max_{ij}(PS_{ij})$, B_{ij} is equal to 0.

$$\max\ W = \sum_{i,j}(F_{ij} + PS_{ij} + B_{ij} - P_{ij}) * x_{ij} \tag{6}$$

s.t.

$$x_{ij} = 0, x_{ji} = 0 \quad \forall(v_i, v_j) \in CA, \quad \forall(v_j, v_i) \in CA \tag{7}$$

$$x_{ij} = 0 \quad \forall(v_i, v_j) \in IC \tag{8}$$

$$x_{ij} = 1 \quad \forall(v_i, v_j) \in MC \tag{9}$$

$$\sum_j x_{ij} = 0 \quad v_i = de \tag{10}$$

$$\sum_j x_{ji} = 0 \quad v_i = ds \tag{11}$$

$$\sum_j x_{ij} \geqslant 1 \quad \forall v_i \neq v_N \wedge \forall v_i \neq de \tag{12}$$

$$\sum_j x_{ji} \geqslant 1 \quad \forall v_i \neq v_0 \wedge \forall v_i \neq ds \tag{13}$$

$$x_{ij} \in \{0, 1\}, \quad \forall(v_i, v_j) \in ED \tag{14}$$

Constraints (7)–(10) are relations obtained by domain experts. Constraints (7) dictates that the relations between vertices v_i and v_j equal 0 if two events are considered as being in parallel relation. Constraint (8) requires that the relations between vertices v_i and v_j equal 0 if two events are considered to be not-related. Constraint (9) specifies that the relations between vertices v_i and v_j equal 1 if two events are considered as having the causal relation.

Constraints (10) and (11) designate the end and start activities, respectively. Constraint (10) indicates that the relations between vertices v_i and v_j should be equal to 0 for any vertex v_i that is a designated end activity ($v_i = de$). Constraint (11) indicates that the relations between vertices v_i and v_j should equal 0 for any vertices v_i that are a designated start activity ($v_i = ds$).

In order to obtain a sound process model, it is necessary to build constraints that satisfy the soundness property defined in definition 7. For verification of the soundness of a process model (a valid sequence from a start activity to an end activity), there should be a constraint enforcing a connection between vertices that are neither start nor end activities. Constraint (12) dictates that the relations between vertices v_i and v_j be greater than or equal to 1 for any vertices v_i that are not an end activity ($v_i \neq v_N$) or a designated end activity ($v_i \neq de$). Meanwhile, constraint (13) requires that the relations between vertices v_i and v_j should be greater than or equal to 1 for any vertices v_i that are not a start activity ($v_i \neq v_0$) or a designated start activity ($v_i \neq ds$). A decision variable (x_{ij}) in Eq. (14) is equal to 1 if vertex v_i immediately precedes vertex v_j, where $(v_i, v_j) \in ED$; otherwise 0.

To illustrate the mathematical model, let consider two traces as follows, $\langle$A, B, C, D, E$\rangle^4$, $\langle$A, C, B, D, E$\rangle^3$. Without identifying the domain knowledge, there are two restrictions. First, there should be at least a link for all nodes when the immediate predecessor node is not an end node. That is, for each x_{ij} such that $i \neq E$ generates constraints as follows:

$$x_{AA} + x_{AB} + x_{AC} + x_{AD} + x_{AE} \geqslant 1$$
$$x_{BA} + x_{BB} + x_{BC} + x_{BD} + x_{BE} \geqslant 1$$
$$x_{CA} + x_{CB} + x_{CC} + x_{CD} + x_{CE} \geqslant 1$$
$$x_{DA} + x_{DB} + x_{DC} + x_{DD} + x_{DE} \geqslant 1$$

Second, there should be at least a link for all nodes when the immediate successor node is not a start node. That is, for each x_{ij} such that $j \neq A$

$$x_{AB} + x_{BB} + x_{CB} + x_{DB} + x_{EB} \geqslant 1$$
$$x_{AC} + x_{BC} + x_{CC} + x_{DC} + x_{EC} \geqslant 1$$
$$x_{AD} + x_{BD} + x_{CD} + x_{DD} + x_{ED} \geqslant 1$$
$$x_{AE} + x_{BE} + x_{CE} + x_{DE} + x_{EE} \geqslant 1$$

For each variable, it should be 0 or 1 value. In regard to the objective, it uses the method proposed in this study. For example, the relation between A and B produces value 5.4875 which is obtained from the summation of 4, 0.8, and 0.6875 that represents causal relation, benefit and proximity score, respectively.

4.3. Constructing a process model from ILP

The role of domain expert for imposing the constraints of ILP is important. By referring to the traces in Table 2, it is obvious that the low frequency behavior, for example $\langle$A, F$\rangle^9$, is unexpected on the model. Domain expert can impose the constraint by choosing the relationship of mandatory causality for the behavior. The model, discovered from the event log, will include the imposed behavior even though it is insignificant due to the frequency. It is also applicable to restrict the relatively frequent behavior, for example $\langle$A, B, I, I$\rangle^{150}$, that contains a loop of event I for being discovered in the model by imposing a constraint of indifferent causality. By choosing such constraint, the model is updated by excluding the loop. One of the challenge of actionable process discovery is to discover implicit behavior when existing behaviors are not actionable while the activity is not either a start or end activity. For example, domain expert claim that activity "Shuffling" should follow "Remarshalling Stack" since the activity "Remarshalling

Stack" could not immediately end without any further action. Hence, domain expert imposes a constraint of indifferent causality on "Remarshalling Stack" to "End" and a constraint of mandatory causality on "Remarshalling Stack" to "Shuffling".

Based on the restrictions obtained from the domain knowledge, a dependency graph can be automatically augmented with proximity information, called as causality assignment. A causality assignment based on proximity embed in the ILP and plays a pivotal role in discovering a sound process model. The important part is the propagation of the causality that a particular causality assignment usually constraints the assignment of set of other causalities. Constraint (12) and (13) are the central for causality assignment. Whenever a node which is not either a start or an end activity, it should have a causal relation from a start activity and to an end activity. For example, setting two nodes as indifferent causality leads to unsound process model and forces to determine other nodes as either direct predecessor or direct successor; or setting the node as designated start (designated end) causes every other predecessors (other successors) remain with no causality and assigns other nodes for satisfying soundness. The soundness properties guarantee the dependency graph hold the workflow property. However, it is fewer guarantees that the mined model has high quality since it is a kind of a trade-off between technical interestingness and business interestingness. This study investigates the causality assignment after domain knowledge is imposed. Theorem was built and the proof can be found in the appendix of this paper.

Theorem 1. *Let* $\boldsymbol{G} = \langle \boldsymbol{V}, \boldsymbol{ED} \rangle$ *be a sound graph and let* $\boldsymbol{L}$ *be a complete event log. For any* $mc_{ij} \in MC | v_i, v_j \in V \wedge (v_i, v_j) \in ED$ *implies soundness.*

1. *If* $\exists_k\ ed_{ij}^k = \varnothing$, *then a graph* $\boldsymbol{G}$ *with domain knowledge* mc_{ij} *is sound.*
2. *If* $\forall_k\ ed_{ij}^k = \varnothing$, *then a graph* $\boldsymbol{G}$ *with domain knowledge* mc_{ij} *is sound.*
3. *If* $\forall_k\ ed_{ii}^k = \varnothing$, *then a graph* $\boldsymbol{G}$ *with domain knowledge* mc_{ii} *is sound.*

A domain knowledge of mc_{ij} *indicates that an imposed constraint by domain expert is a mandatary relation between the two nodes. If the node* v_i *is not an end activity, there should be, at least one, successor node until an end activity. If there is no prior relation* $(v_i, v_j) \in ED$, *the additional relation from* v_i *to node* v_j *that is not a start activity keep the soundness property and impose an addition constraint of* $\exists_k\ ed_{ij}^k = \varnothing$. *Unless* v_i *is an end activity and* v_j *is a start activity, the dependency graph remains sound.*

Fragment of traces in Table 2 can be used to illustrate the theorem. Suppose there is no case of traces such as ⟨A–F⟩. *Since it is unexpected but actionable, domain expert imposes a mandatory constraint to relate A and F. Since the proximity between A and F can be derived from other traces such as* $\langle A\text{–}B\text{–}E\text{–}F \rangle^{800}$, *there exist such a high probability that these two nodes are related.*

Theorem 2. *Let* $\boldsymbol{G} = \langle \boldsymbol{V}, \boldsymbol{ED} \rangle$ *be a sound graph and let* $\boldsymbol{L}$ *be a complete event log. For any* $ic_{ij} \in IC | v_i, v_j \in V$ *implies soundness.*

Based on domain expert, the fragment log in Table 2 contains unexpected relationship such as ⟨ . . . I–I . . .⟩. *This self-loop is actually unavoidable in the real world process. However, domain expert can impose a constraint of indifferent causality to deselect the causal relations. By the definition of edge, there should be an immediate predecessor (or immediate successor) of two nodes. Hence, unsound process model could not happen when the two activities are disconnected. Instead, the causality assignment attempts to find any other nodes that either precede I or succeed I. In other word, there exists a possible relation for both predecessor and successor of I. This situation holds for any edges.*

Theorem 3. Let $G = \langle V, ED \rangle$ be a sound graph and let L be a complete event log. For any $ca_{ij} \in CA | v_i, v_j \in V$ implies soundness.

The fragment log in Table 2 contains no concurrent activities. Suppose, two nodes G and H have paths ⟨ . . . G, H, G, H, . . . ⟩. *The low frequency of those nodes does not hinder the discovery process. The elimination of two edges, for example* ⟨. . .G–H. . .⟩ *and* ⟨ . . . H–G. . .⟩ *are still able to mine a sound process model when there are another predecessor of G or H and there are another successor of G and H. For example, B and D could be the candidate of predecessor of G while C and E are the candidate of successor of* ⟨ . . . G–H . . .⟩. *Hence, the mined model holds the soundness.*

Theorem 4. *Let* $\boldsymbol{G} = \langle \boldsymbol{V}, \boldsymbol{ED} \rangle$ *be a sound graph and let* $\boldsymbol{L}$ *be a complete event log. For any* $v_j \in V$,

1. *If* $v_i \neq v_0$ *and* $v_i = ds$ *and there exists* $(v_j, v_i) \in \boldsymbol{ED}$, *then* G' *is an improved sound graph of* $G / \{(v_j, v_i) | \forall_j v_j \wedge (v_j, v_i) \in \boldsymbol{ED}\}$.
2. *If* $v_i = v_0$ *and* $v_i = ds$ *and there exists* $(v_j, v_i) \in \boldsymbol{ED}$, *then* G' *is an improved sound graph of* $G / \{(v_j, v_i) | \forall_j v_j \wedge (v_j, v_i) \in \boldsymbol{ED}\}$.

Based on the fragment log in Table 2, there are several possible start nodes. For example, a set of start activities comprise of {A, D, G}. Since G follows H and is followed by B, the model will show node G as an intermediary node. Intermediary node in this study is a node that has predecessor and successor. Suppose, the domain expert imposes a constraint to designate node G as a start activity. There could be a consideration to relate G with A since there is a proximity score between A and G. Although the proximity score exists, the intention to choose G as designated start activity enforces no predecessor of G and there should be a successor of G (i.e., node H). Therefore, the mined model is sound.

Theorem 5. *Let* $\boldsymbol{G} = \langle \boldsymbol{V}, \boldsymbol{ED} \rangle$ *be a sound graph and let* $\boldsymbol{L}$ *be a complete event log. For any* $v_i \in V$,

1. *If* $v_j \neq v_N$ *and* $v_j = de$ *and there exists* $(v_j, v_i) \in \boldsymbol{ED}$, *then* G' *is an improved sound graph of* $G / \{(v_j, v_i) | \forall_i v_i \wedge (v_j, v_i) \in \boldsymbol{ED}\}$.
2. *If* $v_j = v_N$ *and* $v_j = de$ *and there exists* $(v_j, v_i) \in \boldsymbol{ED}$, *then* G' *is an improved sound graph of* $G / \{(v_j, v_i) | \forall_i v_i \wedge (v_j, v_i) \in \boldsymbol{ED}\}$.

Based on the fragment log in Table 2, there are several possible end activities; {C, I, F, H}. Since there exists both predecessor and successor of H, node H is considered as an intermediary node. Suppose, the domain expert imposes a constraint to designate node H as an end activity. Causality assignment can relate H with F since there is a proximity score between H and F. Although the proximity score exists, the intention to choose H as designated end activity enforces no successor of H and there should be a predecessor of H (i.e., node G). Therefore, the mined model is sound.

Domain experts play an important role to build the domain knowledge (constraints). However, domain experts may not be familiar with process analysis. Since the main challenge of process mining is to automatically discover process model from event log, it is a time-consuming task when the domain experts associate with exaggerated constraints. In addition, since it needs the involvement of domain expert, it is difficult to apply in any

situations. Therefore, this study attempts to propose an approach to automatically mine a process model by reusing the ILP and excluding the domain knowledge. Instead, we use expert's belief that refers to a threshold value. This threshold, which later called by belief threshold, applies to ILP and manipulate the cost value in objective function. A high belief threshold aims to discover a model which covers high proximity score. Naturally, this approach should be comparable with the result of integer linear programming. However, actionable process discovery aims to discover a model that can represent the business interestingness. In this case, the business interestingness refers to the actionable aspects such as actionability in the real world. Instead of finding the optimality of the process model, this study focuses on the precision of the model with the domain knowledge.

5. Implementation and evaluation

This section describes the implementation of our approach. It also includes a case study in the domain of port logistics. Additionally, evaluation results, both quantitative and qualitative, are analyzed.

5.1. Dataset introduction for experiment

A case study of a port logistics process of landside transport was conducted (Gunther & Kim, 2005). In process mining, the common meta model, referred to as the event type, usually begins with *schedule*, proceeds to *start*, and ends with *complete* (Van Dongen & van der Aalst, 2005). However, the current event data-recording technology for ports is dependent on human operators, who sometimes fail, due for example to high workloads, to represent the real time, which is to say that they record only the completion of an activity without recording the *start* event. Thus, the port's expert might decide to analyze event logs using *schedule* and *complete* events that can have behavior differences compared with a common meta model derived from existing mining techniques. In this event log, the event type *schedule* means that a user receives an inquiry on that activity and schedules the task in the system.

As aforementioned with regard to a port logistics process, this study used nine main activities. The dataset was comprised of two event types, *schedule* and *complete*. For better representation, the experiments were conducted based on two datasets. The first dataset included only the *complete* event type and the second had both *schedule* and *complete* event types. Table 3 illustrates a fragment of an event log based on the two event types.

Table 3
Fragment of event log based on two event types.

CaseID	Activity	Timestamp	Event type
1	QuayJobDischarge	2012-10-23 08:09:35	Schedule
1	QuayJobDischarge	2012-10-23 11:10:25	Complete
1	YardJobDischarge	2012-10-23 11:45:10	Schedule
1	YardJobDischarge	2012-10-23 12:13:08	Complete
1	YardJobGateOut	2012-10-23 14:03:51	Schedule
1	YardJobGateOut	2012-10-24 10:11:47	Complete
2	QuayJobDischarge	2012-10-23 08:12:41	Schedule
2	QuayJobDischarge	2012-10-23 10:31:54	Complete
2	YardJobDischarge	2012-10-23 12:28:41	Schedule
2	YardJobDischarge	2012-10-23 15:48:17	Complete
3	RemarshallingPickUp	2012-10-23 14:05:38	Schedule
3	RemarshallingPickUp	2012-10-24 06:27:15	Complete
3	RemarshallingStack	2012-10-24 07:00:20	Schedule
3	RemarshallingStack	2012-10-24 07:42:19	Complete

5.2. Implementation

Proximity miner was implemented in the ProM[1] process mining framework and demonstrated using port logistics data. Each event, which consists of two types, was identified by a key concatenated with the activity name and the event type. The mathematical model was implemented using the commercial software LINGO 11.0 and an open-source tool, LpSolve 5.5.0.[2] The software was run on a Core i7 860 2.80 GHz, 8 GB RAM desktop computer.

This approach has been applied to a real-world process mining application for mining of actionable process models. This is to mine the discovered process patterns in port logistics. The word *actionable* indicates that the flow patterns should not only satisfy technical significance based on used process mining methods, i.e. conformance checking, but also should have potential in the physical world when they are executed in the field.

Fig. 2 shows proximity miner's initial page, which includes the sets of constraints. After the constraints are set, the graph result will be shown as in Fig. 4.

The experiment was run from two scenarios. First, the domain expert personally sets the specific constraints relevant to the application. This task consumes a lot of time, particularly when the number of activities is large. Second, the proposed approach affords users to adjust some minimum threshold (or confidence of expert's belief) for retrieving the actionable model. While the former guarantee soundness, the latter provides functions to show only the rules that are not expected or actionable based on the designated minimum threshold.

The dataset used in this study was a month's worth of container data in 2012: 66,962 cases and 295,379 events. An artificial start and an artificial end were added to the dataset for the purposes of the study.

Fig. 3(a) shows all of the relations between nodes including the two event types, schedule and complete. Initially, our approach does not exclude infrequent traces, unless they violate imposed constraints. The specific constraints relevant to the application can either be obtained by interviewing domain experts or set personally by domain experts. A domain expert suggests to set some constraints such as mandatory causality, which is denoted in the blue line of Fig. 3(a), and indifferent causality, which is denoted as cross icon in the figure. The choice of constraints used for determining actionable behaviors (insertion or deletion of relations) heavily affects the mined process models (Fig. 3(b)). Therefore, experimenting with an incremental number of constraints can be a helpful way to identify the constraint set that leads to the best results.

To simplify the result of the mined actionable process model without user intervention, the experiment used both the *complete* event type alone and *schedule* and *complete* together. Using dataset with *complete* event type alone, the initial result with all causal relations between nodes is shown in Fig. 4(a). The constructed process model in the figure was the result of our approach using a minimum threshold of 0.5. The red line indicates that the causal relations are not expected and not actionable to the degree of 0.5. By exploiting the result, the mined process model without those rules is shown in Fig. 4(b). Another approach to obtain the result is to allow domain experts to set the specific constraints relevant to the applications. The choice of constraints used for determining actionable behaviors can affect the mined process models. In other words, experimenting with an incremental number of constraints can be a helpful means of identifying the constraint set that leads to the best results.

[1] www.processmining.org.
[2] https://sourceforge.net/projects/proxi-miner/.

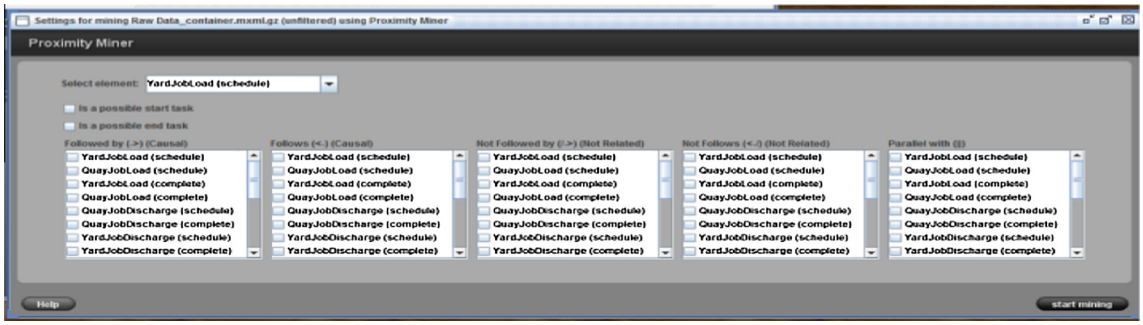
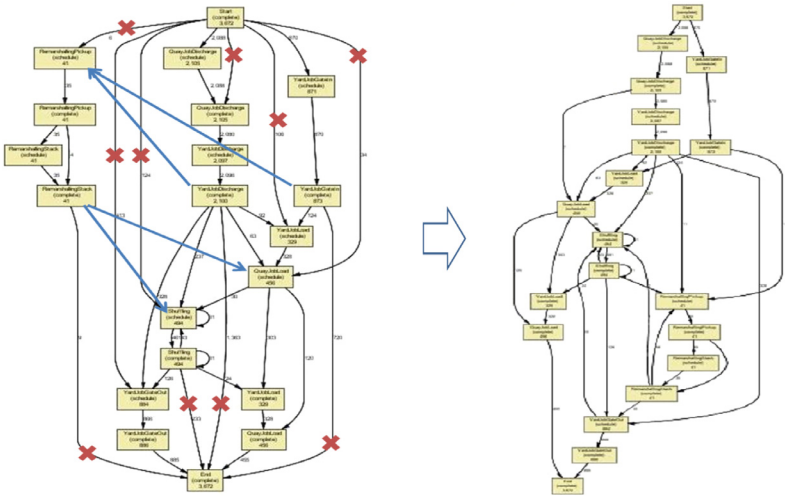
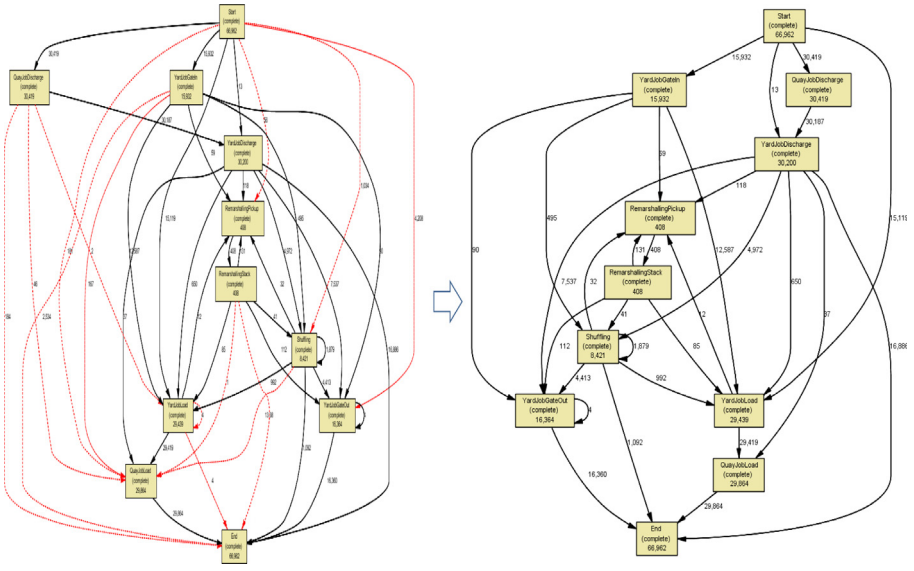


Fig. 2. First page of proximity miner.



(a) The result based on all events (b) The result based on constraints

Fig. 3. (a). Result of Proximity Miner based on all events relations and (b) the result after considering constraints.



(a) The result with minimum threshold = 0.5 (b) The mined result excluding the non - actionable relations

Fig. 4. (a). Result of Proximity Miner with all relations and (b) the result excluding the non-actionable relations.

The discovered process model is more complex when it contains two event types. For example, there can be some flows in which an activity with event type *schedule is followed by* another activity of event type *schedule*. This study focuses only on discovering an actionable process model and leaves the semantics due to the existence of two event types for future work. The representation of the categorization for event logs that contain two event types (without artificial start and artificial end) is shown in Fig. 5. While Fig. 5(a) shows the overall causal relations in the event log, Fig. 5(b) demonstrates the exclusion of the actionable relations with the threshold of 0.5. Thus, this approach also can be considered an interactive tool for finding an actionable process model. Remaining questions about fitness and behavioral appropriateness are discussed in the next section.

5.3. Quantitative evaluation

There are two measures used in this study. First, we applied measures related to process mining. Second, subjective interestingness measures are introduced to quantify the fitness of actionable knowledge over process model.

5.3.1. Measures on process mining

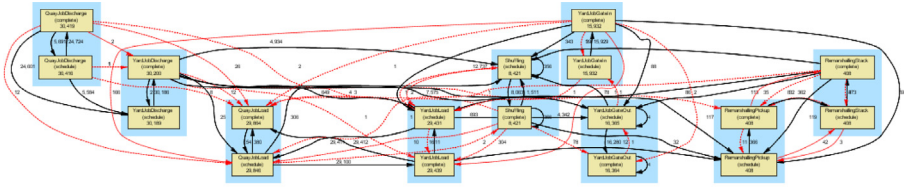
In this study, we applied measures from a previous work: fitness and behavioral appropriateness (Rozinat et al. 2007). Fitness (f) is a metric that is obtained by determining whether each (grouped) sequence in the event log can be reproduced by the generative model. It is often referred to as sequence replay. By using a generative model of Petri Net, the initial value of f at the start of sequence replay is a value of one. Behavioral Appropriateness (ABA) is a metric that is obtained by exploration of the state space of a Petri Net and by comparison of the different types of following and precedence relationships that exist in the state space with the different types of following and precedence relationships that exist in the event log. This proportion is defined as the number of following and precedence relationships that Petri Net has in common with the event log vis-a-vis the number of relationships that it, Petri Net, allowed. Degree Model Flexibility (DMF), as a part of ABA, is a metric that allows for flexibility in the model according to the activity relations derived from the log. It equals 0 for a model that only allows for a particular sequence of steps, or 1 for the "flower" model allowing for arbitrary execution of the contained steps. Those measures have been provided in ProM 5.2 tool and were utilized for this study. For further details, reader can refer to Rozinat et al. (2007).

Data experiments (as shown in Table 4) were performed to analyze the performance of proximity miner. The results show that proximity miner runs slower when data includes a lot of loop behavior. In the process mining setting, the running times of solution approaches do not usually represent a major issue (as algorithms are applied offline, i.e. during the (re)-design phase). Since the main focus of this study was on generating a better quality of mining result, i.e. an actionable process model, we set aside the performance problem for future work.

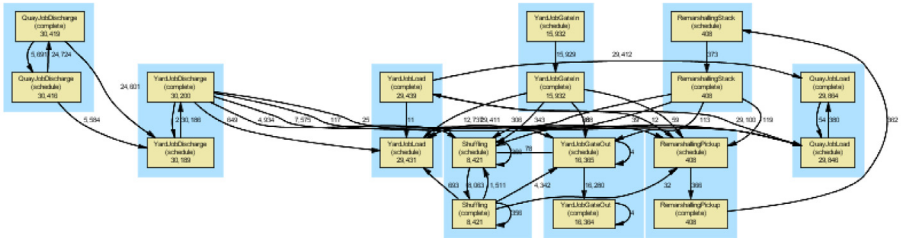
The experiment results for a real log of a logistics process based on a container (see Table 4 for the container data) are shown in Table 5. Since the measure run based on Petri Net, the proposed result was converted from dependency graph to Heuristic Net and, finally, reutilized the conversion tool from Heuristic Net to Petri Net. Afterward, the converted result was evaluated using the existing conformance checker tool. By applying a number of constraints from 0 to 30 at intervals of 5, the f-measure (fitness measure), ABA and DMF value results were obtained. The f-measure and DMF results decreased slightly when the constraints were larger. Since f-measure is a token (sequence) replay of a mined model from the event log, it is certain that it falls when the constraint is larger. As for DMF, larger constraints effect a slight increment in the measure with constraints equal to 10 and 15. Afterward, it was shown as the better model for representing the flexibility of sequences. The ABA measure indicated that the behavior of the mined model was gradually increasing when the constraint was larger. This means that the process model has a high

Table 4
Summary of data of port logistics process with time performance using proximity miner.

Dataset	Cases	Events	Elapsed time	Note
Container	14,481	65,307	9 s 302 ms	–
Vessel	22	15,787	1 min 50 s	High frequency of loops
Truck	18	21,270	1 min 23 s	High frequency of loops
Block	7624	25,580	2 s 267 ms	–
QuayCrane	8	16,211	1 min 47 s	High frequency of loops
YardCrane	24	56,975	8 min 30 s	High frequency of loops



(a) Categorization of actionable models with minimum threshold 0.5



(b) The mined result with categorization excluding non-actionable relations (threshold 0.5)

Fig. 5. (a). Result of Proximity Miner with all relations of two event types and (b) the result excluding the non-actionable relations.

Table 5
Experiment results from real log (full log) for proximity miner.

Measures	Number of constraints						
	0	5	10	15	20	25	30
f-measure	1.00	0.99	0.99	0.99	0.99	0.99	0.99
ABA	0.61	0.63	0.63	0.64	0.68	0.69	0.69
DMF	0.62	0.57	0.70	0.71	0.67	0.69	0.56
Time (s)	16.615	16.442	14.533	14.583	15.584	15.367	16.175

Table 6
Comparison of experiment results of real logs with other mining techniques.

Measures	Mining techniques				
	Alpha miner	Alpha +	Heuristic miner	Genetic miner	Proximity miner
f-			measure	0.82	0.82
	0.98	**0.99**			
0.98					
ABA	0.77	0.74	0.82	**0.90**	0.69
DMF	0.38	0.37	0.47	0.38	**0.56**
Time (s)	2.961	1.956	4.792	252,001	16.175

proportion of following and precedence relationships in the event log. For the purposes of a comparison, the experiment also was conducted using container data in alpha miner, alpha+, genetic miner, and heuristic miner (see Table 6). It should be noted that alpha miner, genetic miner, and heuristic miner are prominent discovery tools of process mining. In addition, in the user's perspective, those mining tools can represent the extracted patterns. This experiment exclude comparison analysis with other existing tools, i.e., AGNEs and precedence constraints, due to inaccesibility to those approaches.

The experiment results based on *f*-measure has led to the tool's effectiveness with regard to business interestingness issue in twofold. First, it identifies the effectiveness of proximity miner in term of business interestingness, as shown in Table 5. As the value one means all sequences in the event logs can be produced in the generative model, the decrement values of *f*-measures due to the increment of constraints indicate that generated constraints by the users satisfy the sequences in the event logs. In other words, the business interestingness is well-represented when the value is nearly one. Second, it determines the effectiveness of the proposed approach in comparison to some prominent tools, as shown in Table 6. Best scores are highlighted in bold. Proximity miner has been shown to outperform other approaches in terms of *f*-measure and DMF. Meanwhile, genetic miner performs better than all the other approaches in terms of ABA. Heuristics miner and genetic miner, the prominent tools in process mining, show a high value of *f*-measure. It means both heuristics miner and genetic miner can result a without-user-intervention process model which is almost the same with the proximity miner in term of business interestingness. Since there is no guarantee that the model generated by heuristics miner is an actionable model, which includes user's intervention, thus user have no confidence on the applicability of such model. Hence, this paper also conduct qualitative study for verifying the result based on the domain expert.

5.3.2. Measure of interestingness

This section discusses the measure *interestingness*. The interestingness is the degree of a discovered process model's compliance with domain knowledge. High interestingness means that the discovered process model has many matching relations with domain knowledge. On the other hand, low interestingness means that there is little correlation between the discovered process model and domain knowledge.

In this study, we utilized subjective interestingness measures to indicate the actionability of the discovered model in comparison to domain knowledge. The domain knowledge is obtained from domain experts who have long experience in the field. The domain knowledge will be classified into the three aforementioned categories: UA, U$\underline{A}$, and AE. For this study, the domain expert provides the domain knowledge pertaining to a specific port. Note that these relations were established based on the expert's experience in a specific port and could be different from other ports' characteristics. The relations (i.e. the interestingness based on domain knowledge) are illustrated in Fig. 6.

Fig. 6 illustrates the domain knowledge in the form of a matrix. Any relation with the text 'AE' represent a normal process as well as an actionable flow according to the expert's knowledge. This is presented as green-colored cells. Any relation in UA (in yellow[3]-colored cells) are considered possible relations (i.e., unexpected but actionable). This means that a real-world system can have an actionable flow even though it is unexpected. For example, QD is possible to be the immediate predecessor of YL for some reasons (e.g., distance and schedule). This flow can happen when a vessel is at a berth nearby the discharged location. In this sense, the quay crane (QC) grounds a container near the crane, instead of in the yard, and later moves it somewhere else or loads it directly onto the vessel. The relations with the text U$\underline{A}$ (red-colored cells) are not physically possible. In other words, the flow is unexpected and not actionable in the real world. For example, remarshalling pickup activity (RP) should not be followed by yard crane (YC) loading (YL), since it needs first to be grounded. The red-colored cells have two meanings: a relation that can appear in the log due to minor execution in the field, or a relation that has never been detected from the log. The former is illustrated by the cells with the text "U$\underline{A}$" and the latter is represented by the red-colored-blank cells.

The process on building the constraints by domain expert is error-prone, in particular, when the domain knowledge is complex. Here, we also propose a process discovery approach using ILP based on belief threshold. This approach attempts to discover a process model based on the designated belief threshold value. The threshold is a value between 0 and 1 which represent the belief of users. The belief threshold in this approach refer to the proximity score. It means, the proximity score which is higher than the belief threshold will be shown as the mined model. Otherwise, the ILP will ensure to choose relevant edges to guarantee the soundness while maximizing the objective function. However, unsound model occur when the belief threshold is high (e.g., more than 0.6) due to the existence of loop. Therefore, to tackle such problems, we develop additional function (i.e., MinimalSoundness) to produce a sound process model. The detailed procedure is as follows.

[3] For interpretation of color in Fig. 6, the reader is referred to the web version of this article.

	QD	YD	YI	YO	YL	QL	RP	RS	S
QD	UA	AE	UA	UA	UA	UA	UA		UA
YD		UA		AE	AE	UA	AE		AE
YI			UA	AE	AE	UA	AE		AE
YO				UA					UA
YL					UA	AE			
QL					UA		UA		UA
RP				UA	UA		UA	AE	
RS				AE	AE				AE
S				AE	AE		AE		UA

Fig. 6. Interestingness based on domain knowledge (refer to Table 1 for the activity description).

```
MinimalSoundness(CausalRelation F)
1 Matrix F ← F;// it is a matrix of causal relation of
  node i and j
2 Map⟨key, value⟩ incomingNode, incomingFreq,
  outgoingNode, outgoingFreq;
3 foreach x_ij ∈ F do{
4  if (incomingNode(j) exist){  //if there exist
  another incoming Node
5    if (freq(x_ij) < incomingFreq(j)){  //freq
  (xij) = frequency of xij
6      incomingNode(j) = i;      //update the
  incoming Node
7       incomingFreq(j) = freq(x_ij);
8   }}
9  else {        //add new incoming Node
10       incomingNode(j) = i;
11       incomingFreq(j) = freq(x_ij);
12 }
13 if (outgoingNode(i) exist){  //if there exist
  another outgoing Node
14  if (freq(x_ij) < outgoingFreq(i)){
15     outgoingNode(i) = j;    //update the
  outgoing Node
16       outgoingFreq(i) = freq(x_ij);
17   }}
18 else {        //add new outgoing Node
19     outgoingNode(i) = j;
20     outgoingFreq(i) = freq(x_ij);
21 }}
```

The lines 1–2 are variables declarations for the algorithm. In lines 3–21, it runs for each causal relation. When there exist a causal relation between two nodes, the incoming and outgoing nodes should be stored including the respective frequency. Lines 4–12 and 13–21 denote a procedure to store the incoming nodes and outgoing nodes, respectively. The procedure finds the maximum frequency among existing edges and stores the most frequent causal relations for each nodes. Eventually, the variable *incomingNode* and *outgoingNode* are used to produce a sound process model. To measure the effectiveness of this approach, this study explored the interestingness measure using precision and recall.

Suppose that D is the set of nodal relations representing domain knowledge and that G is a graph representing the discovered process model. The precision is the ratio of the number of correct relations discovered by an approach to the total number of relations returned by the algorithm. The recall is the ratio of the number of correct relations returned by an approach to the total number of relations in the domain knowledge. Thus, precision and recall can be formulized as follows in Eqs. (15) and (16).

$$precision = \frac{|D \cap G|}{|G|} \tag{15}$$

$$recall = \frac{|D \cap G|}{|D|} \tag{16}$$

The experiment utilized the data set represented in Table 7. The dataset is the data of the container flow at the port terminal, and was partitioned into seven fragments with gradually increasing numbers of cases and events. As a matter of fact, two partitioned datasets can have different graph patterns due to size variability. Thus, another measurement parameter, for example threshold, might need to be considered.

Finding the best belief threshold value is a general issue in the domain of data retrieval. By definition, a belief threshold is a ratio number by which to automatically prune the result based on the proximity score. Threshold adjustment determines the precision and recall. Thus, the present experiment was performed on the dataset with some reasonable threshold values such as 0, 0.5, 0.6, 0.7 and 0.8. A threshold value of 0 indicates that the approach discovered a process model as it is represented in the event log, which is to say, including noise. With a threshold value greater than 0, the graph will be pruned. A higher threshold value means that there is a higher probability of obtaining high precision. However, a higher precision value might be correlated with a lower recall value.

Note that this experiment was conducted by comparison with only heuristics miner (HM) due to two reasons. First, the *F*-measure of Proximity Miner and Heuristic Miner show slightly different (Table 6). Although Genetic Miner also shows its superiority, the high execution time is not much expected by users. Second, some measures such as *F*-Measure, ABA, and DMF can evaluate the technical interestingness. Fig. 7(a) and (b) shows the experiment results of *F*-Measure and ABA. Note that those two measures are sufficient to show the technical interestingness of the approaches. In term of business interestingness (or actionable model) which refers to domain knowledge, we measure precision and recall to measure the effectiveness of the approach.

Based on Fig. 7(c), Proximity Miner (PM) is superior to Heuristic Miner (HM) in precision. It shows that the precision of HM lies in between 0.6 and 0.7 meanwhile PM outperforms by showing the

Table 7
Dataset for precision and recall experiment.

Dataset	Cases	Events
1	4084	15,897
2	9980	40,194
3	21,529	91,101
4	33,559	146,249
5	39,494	173,108
6	51,870	229,368
7	66,962	295,379

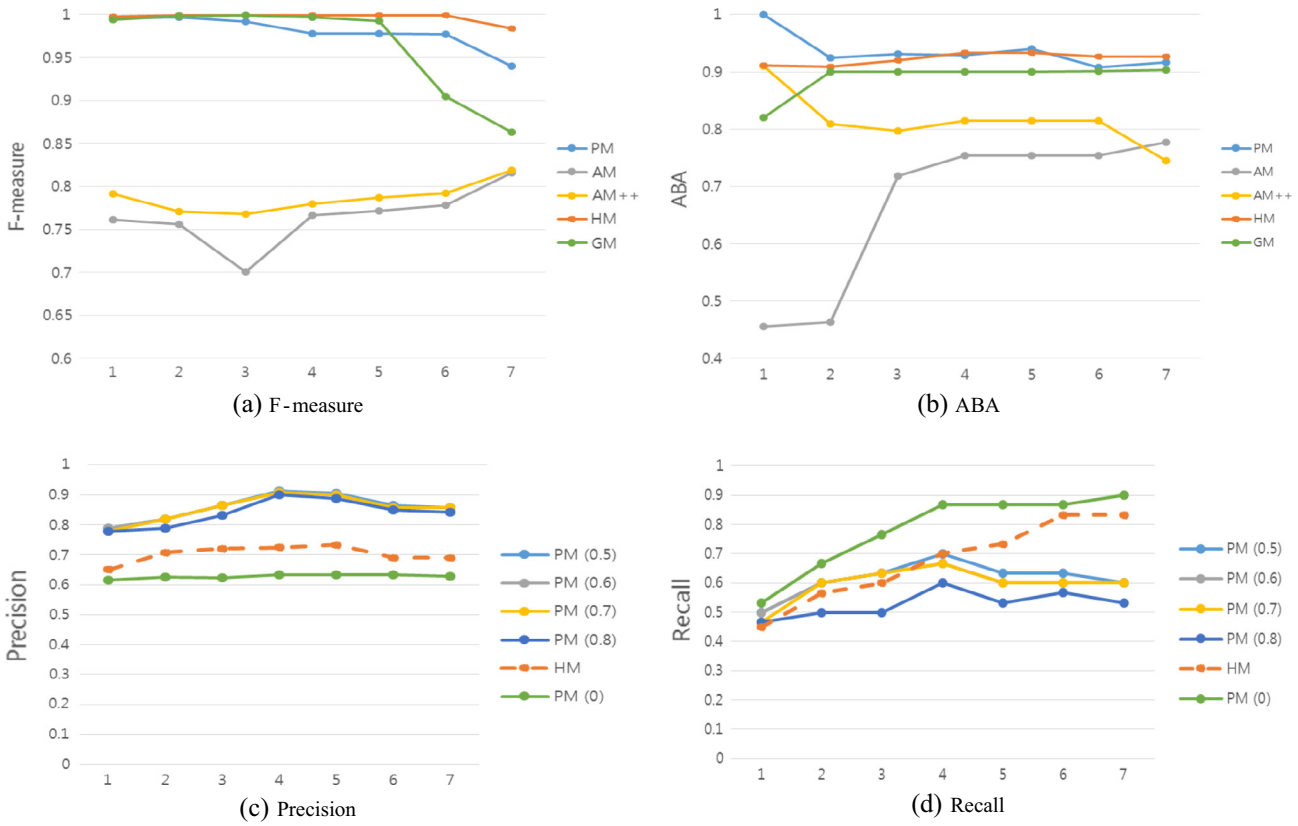


Fig. 7. Measures based on (a) *F*-measure, (b) ABA, (c) precision and (d) recall using data container.

precision higher than HM when the belief threshold is over 0.5. On the other hand, HM dominates PM in recall for a large dataset. For a small dataset however, PM can provide high recall for the belief thresholds 0.5 and 0.6. Thus, PM is better than HM to obtain an actionable process model from a small dataset.

5.4. Qualitative evaluation

This study obtained an evaluation from a domain expert. Since the case study pertained to a logistics process, we interviewed a domain expert working in the port logistics domain with regard to the mining result. For this study, the business interestingness is instantiated in terms of subjective factors from technical and business sides.

As a domain expert should be familiar with process mining terminology, he was involved in the development of this proposed algorithm, specifically by proposing some improvements to the existing techniques. The proximity miner results were compared with the process mining techniques alpha miner, heuristics miner and fuzzy miner. The proposed approach is not comparable to some other approaches, i.e. AGNEs and process discovery using precedence constraints due to the following reasons. First, AGNEs has different perspectives on building the constraints, i.e. it automatically generates negated events based on the event logs. Second, process discovery using precedence constraints had limitations on loop behavior which is one of the important behavior in this approach. In the evaluations, seven mined-model quality categories were applied: interpretability, loop detection, noise detection, exactness, scalability, categorization, and time performance. The results are shown in Table 8.

Table 8
Qualitative evaluation results.

Category	Measures	AM	HM	FM	PM
Process model	Interpretability	⊙	•	•	•
	Loop detection	○	•	•	•
	Noise detection	⊙	•	•	•
	Exactness	○	⊙	⊙	•
	Scalability	•	⊙	•	•
	Categorization	•	○	○	•
Performance	Time performance	•	•	•	⊙

•: Significant, ⊙: Moderate, ○: Insignificant. AM : Alpha Miner, HM : Heuristics Miner, FM : Fuzzy Miner, PM : Proximity Miner.

5.4.1. Interpretability

Interpretability can be described as the possibility of interpreting the mined model for users. Heuristics, fuzzy miner and proximity miner basically employ a graph to represent the process model. The domain expert agreed that these techniques are easier to interpret than alpha miner, which uses Petri Net to represent the result. Alpha mining was chosen because of the possibility to discover the control-flow. Heuristics and fuzzy mining were chosen due to their robustness and expressiveness, respectively. As a result, each approach show the result variability due to the mechanism used in the respective techniques. It should also be noted that the present study used the default settings of existing mining techniques. As a consequence, the help of experts in constructing an actionable model from the logs was necessary.

5.4.2. Loop detection

Loop detection indicates the extent to which the approach can detect a simple loop, which is an iterative activity. In the case of

a port logistics process, there is much in the way simple loop behavior. Alpha miner has a drawback in retrieving a simple loop from event logs (van der Aalst et al., 2004). As for the heuristics and fuzzy miners, the parameter default settings can detect a simple loop from event logs; however, there are some cases that cannot be accounted for, due to bad guesses on the parts of those algorithms (i.e. the disregard of infrequent traces).

5.4.3. Noise detection

Noise detection is intended to identify and show noise. Since alpha miner is a noise-free algorithm, we disregarded it. Heuristics miner, contrastingly, is considered a robust discovery technique that can effectively remove noise. Fuzzy miner can also mine a process model of a less-structured process by aggregating the infrequent traces (i.e. noise) and abstracting the process flow. In fact, a user also needs to know the noise retrieved from event logs. Proximity miner, instead of removing noise, includes all possible relations and shows the overall results for an expert's review and decision.

5.4.4. Exactness

Exactness is regarded as an evaluation technique to determine the exactitude to which the mined model represents users' knowledge. Alpha miner is an algorithm to detect activity instead of events; and since a log includes the event types *schedule* and *complete*, alpha miner cannot precisely discover a mined model. It should be noted that the results in alpha miner were obtained by disaggregating the events into the two nodes. Whereas heuristics miner can discover the mined model similarly to users' knowledge, it still has some problematic behaviors: for example, it does not keep infrequent traces. Fuzzy miner abstracts the process based on the frequency, but its sub-processing of the mining result is not accurate. The starting point of proximity miner, by contrast, is to perceive all behaviors from the event log. Thus, an exact process model can be obtained by modifying it according to users' knowledge.

5.4.5. Scalability

Scalability is considered as an evaluation technique for measuring the degree to which an algorithm satisfies users' requirements. Heuristics miner has no function for modifying the event relations in the process model. However, according to some designated parameters, a different mined model can be extracted. Both alpha miner and fuzzy miner have additional functions for both inserting and deletion of event relations in the process model; but with neither method is there any guarantee of a sound process model. Proximity miner guarantees the soundness of a process model by activities to a set of constraints.

For example, heuristics miner is considered as the best alternative tool to discover the model. According to the result, domain expert could not agree for a link. Hence, there should be a constraint from domain expert to limit the flow, i.e., indifference causality, from certain activities. In case of both alpha miner and fuzzy miner, they will give immediate result with no link. Proximity miner, thus, attempt to capture the soundness of the model and construct another link, which is an alternative to be the successor of given activity, using ILP.

5.4.6. Categorization

Categorization is a method of merging events representing more than one event type. In the present case study, each activity had two event types, *schedule* and *complete*. Alpha miner can merge those events, though it cannot precisely show the relationship between two events. The other techniques, heuristics miner and fuzzy miner, have limitations with respect to the categorization of two events. Proximity miner can categorize and merge event types without disregard for their behavior (e.g. a *schedule* event type should precede a *complete* event type). Fig. 8 displays a fragment of a port logistics process with merged events.

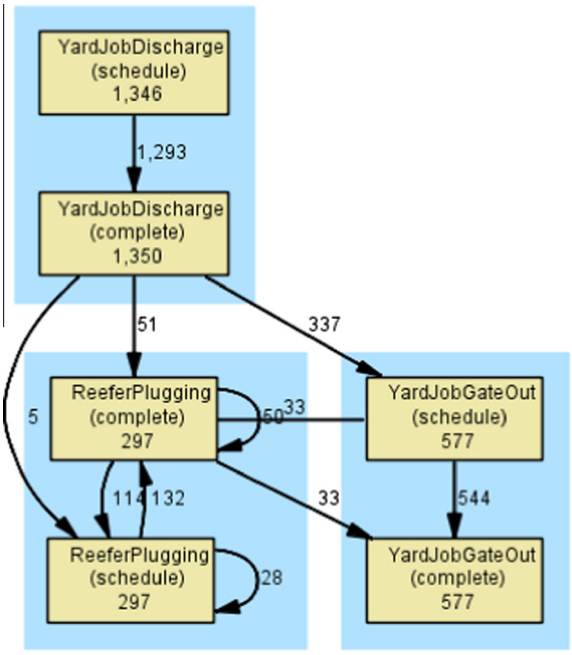


Fig. 8. Example of fragment of port logistics process with merged events.

5.4.7. Time performance

The existing techniques perform better than proximity miner in terms of time. The quantitative results listed in Tables 4 and 6 show that proximity miner suffers in its time performance when the data is larger, as relates to its iterative or loop behavior.

Among all of the aforementioned factors, the expert deemed interpretability, exactness, scalability and categorization the most important to process analysis in the port logistics domain. Since one of the purposes of process model discovery is provision of a communication tool for other users or stakeholders, a generated process model using a graph representation is more preferable. Exactness, scalability and categorization, moreover, also enable the possibility of enhancing the discovered model based on the event relationships. Table 9 exists in a log (QL = *QuayJobLoad*, and YL = *YardJobLoad*). According to the expert, all of the traces have three key relations, as shown in the column of key relations. A key relationship between two events, for example, that of YL (*schedule*) and QL (*schedule*), is considered highly important, whereas other relations are considered less important. The traces can be varied depending on the implementation; however, the key relationships remain the same. Thus, the involvement of an expert in mining a better quality of the process model is necessary.

According to the interview with the domain expert, the results of the proposed approach show merits and demerits. First, a merit

Table 9
Key relations in port logistics based on process expert's opinion.

Case	Traces	Key relations (Expert's involvement)
(1)	QL (schedule) – YL (schedule) – YL (complete) – QL (complete)	YL (schedule) – YL (complete)
(2)	YL (schedule) – QL (schedule) – YL (complete) – QL (complete)	YL (complete) – QL (complete)
(3)	YL (schedule) – YL (complete) – QL (schedule) – QL (complete)	QL (schedule) – QL (complete)

is the ability to identify the extra behavior that is not included in an event log. Second, the proposed approach can show either event-based or activity-based relations, since it is able to group the workflow model elements according to the event type, i.e. *schedule* and *complete*. The demerit of the proposed approach is its dependence on a 3rd library, which slows performance when data becomes large.

5.5. Discussion

This approach, as a part of process mining, aims to discover the as-is process based on event logs not only in the port logistics domain but also in other fields that have process flows. The discovered process model can be used to improve, for example by simulation, the as-is process. Process improvement, one of the hot topics in the field of industrial engineering, should start by analyzing all of the aspects of the process including attributes (e.g., resources that have been utilized) available in the log, and should then return the discovered model and relevant results for further procedure. Hence, user's interestingness for process improvement should be involved during the process to discover the process model.

The present studies show an analysis to the relationships between technical interestingness and business interestingness from the point of view of control-flow. The experiment results among several prominent process discovery tools demonstrate the effectiveness of proximity miner and the potential use of using other similar logs recorded in information systems. It has three salient features. First, discovered actionable process model represents a match between the flow existing in the logs and the actionable rules set by the users. In other words, the fitness measure can be regarded as an indicator to show that this approach represent a degree of subjective (user-driven) measures. Second, discovered actionable process model can be a to-be process model for enhancement since the support of interactivity mode would be a benefit for users who novice in process mining terminology. Finally, although the main motivation behind our work is the business interestingness of event logs, the techniques presented herein can also be applied to manufacturing domain such as semiconductor in which event logs are commonly recorded by information systems and contain less structured process. For example, this approach could cover the limitations of process mining as pointed at a case study of test processes in semiconductor (Rozinat et al., 2009).

It should be mentioned that there exist some limitations on the proposed approach. First, in this study, partial information of port logistics process, i.e., activities and their occurring time stamps, has been used to support actionable process model. Although this study reveals that this partial information is sufficient in discovering actionable process model, there are even more complex analysis and evaluation tasks that need to be considered. As a matter of fact, business decisions in any enterprises are commonly based on cost and quality, which is clearly beyond the support of data used in this study. Second, the existing of loop as the sub-process can result an illogical actionable process model. It means that when the users point out an indifference causality rule from an activity that has only one and the only one flow to another activity as the loop, the former will link to end activity for enforcing the soundness property. In this state, there should be additional properties to keep maintaining that kind of behavior. In addition, the existing of loop causes unsound model when the user apply proximity miner approach with high belief threshold. Third, AND control-flow was not included in the experiment, due to the nature of the event log used in this study. Along with domain knowledge, a more significant event log that includes various behaviors will be necessary to comprehensively prove the utility of this approach. Fourth, although this approach guarantees to result a sound actionable process model, it still remains performance issues with regard to execution time when the activity numbers becomes large. Moreover, the complexity of actionable rules increases when the number of event types and the activity numbers are also increased. These issues will be addressed for future work.

6. Conclusions

Process discovery is a technique by which a process model can be constructed from the event logs of information systems. The current process mining techniques, however, have limitations in representing the actionable knowledge, i.e., actionable process model, based on domain knowledge.

This study formulated a method by which domain experts can be involved in model discovery. The proposed algorithm, known as proximity miner, has been developed to utilize users' knowledge in constructing an actionable process model. Additionally, it offers five process mining advantages. First, it introduces proximity score as a method for finding extra behaviors as potential direct following (adjacent) events when the mined model is not sound. Second, the proposed proximity score both resolves the soundness problems of previous approaches and enhances those approaches. Third, the result can be used to simplify the model by categorizing the event types, or in other words, by merging them according to activity. Fourth, this approach provides an interactive mode for users to determine the actionable rules. Fifth and finally, it is a useful tool for generating an actionable process model based on the knowledge of domain experts that does not require either any effort in drawing from scratch or back-and-forth cycle procedure between filtering and mining tasks. It was demonstrated in ProM, showing its usefulness in the port logistics domain.

There remain some issues for future work. First, the mined model based on users' knowledge is very subjective, to the extent that the knowledge brought to bear is only that of domain experts. Thus, it requires measurement of the quality of the mined model based on additional parameters. With a port logistics process for example, the common principles among ports should be followed. Second, the loop behavior can cause sub-clustering in a process model, which is unsound. Third, AND behavior should be justified with relevant real log. Finally, it requires a method, for example a heuristic approach with users' interventions, to reduce the computational time. These outstanding issues will be considered in our future work.

Acknowledgements

This research was supported by the Basic Science Research Program through the National Research Foundation of Korea (NRF) (Nos. 2011-0010561 and 2013R1A1A2063316). An earlier, abridged version of this paper has been presented at the 1st International Conference on AP-BPM in Beijing, August 2013.

Appendix A

Theorem 1. *Let* $\boldsymbol{G} = \langle \boldsymbol{V}, \boldsymbol{ED} \rangle$ *be a sound graph and let* $\boldsymbol{L}$ *be a complete event log. For any* $mc_{ij} \in MC|v_i, \ v_j \in V \wedge (v_i, v_j) \in ED$ *implies soundness.*

1. *If* $\exists_k \ ed_{ij}^k = \varnothing$, *then a graph G with domain knowledge* mc_{ij} *is sound.*
2. *If* $\forall_k \ ed_{ij}^k = \varnothing$, *then a graph G with domain knowledge* mc_{ij} *is sound.*
3. *If* $\forall_k \ ed_{ii}^k = \varnothing$, *then a graph G with domain knowledge* mc_{ii} *is sound.*

Proof.

1. Assume there exist a causal relation of two nodes such that $ed_{ij}^k = \varnothing$. From Definition 2, we know that $ed_{ij}^k = \varnothing$ implies that $e_i^k \succ e_j^k$ and hence exists as a causal relation F_{ij} or an edge of two nodes $(v_i, v_j) \mid v_i, v_j \in V \wedge (v_i, v_j) \in ED$. A constraint of mandatory causality $mc_{ij} \in MC \mid v_i, v_j \in V$ ensures that such relation exists in the k-th trace, $\sigma_k = \langle e^k{}_1, e^k{}_2, \ldots, e_n^k \rangle \in L$, and can be seen in the dependency graph G. Then, the mined model holds the soundness.
2. Assume there does not exist a causal relation of two nodes such that $ed_{ij}^k = \varnothing$. From Definition 2, we know that $ed_{ij}^k = \varnothing$ implies that $e_i^k \succ e_j^k$ and there is neither causal relation F_{ij} nor an edge of two nodes $(v_i, v_j) \mid v_i, v_j \in V \wedge (v_i, v_j) \in ED$. A constraint of mandatory causality $mc_{ij} \in MC \mid v_i, v_j \in V$ creates such relation. If the node v_i is not an end activity, there should be, at least one, successor node until an end activity. The additional relation to node v_j that is not a start activity keep the soundness property and impose an addition constraint of $ed_{ij}^k = \varnothing$. Unless v_i is an end activity and v_j is a start activity, the dependency graph remains sound. Then, the mined model holds the soundness.
3. Assume that a constraint of mandatory causality $mc_{ii} \in MC \mid v_i \in V$ is imposed by domain expert. Since the dependency graph G has hold the soundness property, given a new created relation such as a self-loop will not affect the soundness. Then, the mined model keep holding the soundness. □

Theorem 2. *Let $\boldsymbol{G} = \langle \boldsymbol{V}, \boldsymbol{ED} \rangle$ be a sound graph and let $\boldsymbol{L}$ be a complete event log. For any $ic_{ij} \in IC \mid v_i, v_j \in V$ implies soundness.*

Proof. Assume there exists a causal relation of two nodes such that $(v_i, v_j) \neq \varnothing$ and it implies that $\exists_k\, e_i^k \succ e_j^k$. A constraint of indifferent causality $ic_{ij} \in IC \mid v_i, v_j \in V$ exhibits no causal relation between two vertices. Hence, it disconnects two vertices $(v_i, v_j) = \varnothing$. Suppose that $v_i \neq v_N$ $(v_j \neq v_0)$, there should be a successor (a predecessor) and a valid sequence until an end node (a start node) according to the soundness property. If the imposed constraint cause no connectivity $\forall_{l \in \{1,\ldots,|A|\}, l \neq i} (v_i, v_l) \in ED$ and $\forall_{m \in \{1,\ldots,|A|\}, m \neq j} (v_m, v_j) \in ED$, constraints 12 and 13 in the ILP have been taken into account to do causality assignment for discovering sound process model. It means, those constraints will search the best proximity to be the successor and predecessor of v_i and v_j, respectively. Then, the mined model holds the soundness. □

Theorem 3. *Let $\boldsymbol{G} = \langle \boldsymbol{V}, \boldsymbol{ED} \rangle$ be a sound graph and let $\boldsymbol{L}$ be a complete event log. For any $ca_{ij} \in CA \mid v_i, v_j \in V$ implies soundness.*

Proof. From Theorem 2, we can deduce soundness when indifferent causality disconnects two nodes, v_i and v_j such that $v_i \neq v_N$ and $v_j \neq v_0$. Similarly, there might be disconnected nodes due to concurrent activities. Since Theorem 2 hold the soundness, hence intuitively this theorem holds the soundness as well. □

Theorem 4. *Let $\boldsymbol{G} = \langle \boldsymbol{V}, \boldsymbol{ED} \rangle$ be a sound graph and let $\boldsymbol{L}$ be a complete event log. For any $v_j \in V$,*

1. *If $v_i \neq v_0$ and $v_i = ds$ and there exists $(v_j, v_i) \in \boldsymbol{ED}$, then G' is an improved sound graph of $G/\{(v_j, v_i) | \forall_j v_j \wedge (v_j, v_i) \in \boldsymbol{ED}\}$.*
2. *If $v_i = v_0$ and $v_i = ds$ and there exists $(v_j, v_i) \in \boldsymbol{ED}$, then G' is an improved sound graph of $G/\{(v_j, v_i) | \forall_j v_j \wedge (v_j, v_i) \in \boldsymbol{ED}\}$.*

Proof. Now we should prove the above two sub theorems respectively.

1. Assume $v_i \neq v_0$. For any $(v_j, v_i) \in ED \mid v_i, v_j \in V$, there exists at least one v_j to have causal relation with v_i to produce a sound process model. If $v_i \neq v_0$, then there is a possibility of v_i to be v_N. In the case that $v_i = v_N$, according to the definition of path (Definition 5), a trace should have events at least two. Hence, a designated start of v_i which carry out property $v_i = v_0$ will lead to unsound process model. On the other hand, when $v_i \neq v_N$, the designated start of v_i will disconnect the relation between v_j and v_i and keep producing a sound model following the soundness property to have flow of a node either from start node or to end node. Hence, the theorem holds the soundness property.
2. Assume $v_i = v_0$. Although v_i is a start node, there could be a relation of $(v_j, v_i) \in ED \mid v_i, v_j \in V, \exists_k\, e_j^k \succ e_i^k$. If $v_i = v_0$, then $v_i \neq v_N$. If v_i is assigned as designated start, it should satisfy the start node as defined in Definition 2. As a result, the disconnection of the relation (v_j, v_i) will not lead to unsound process model since $\exists_k$ $e_i^k \succ e_l^k$ such that $i < l$. Hence, the theorem holds the soundness property. □

Theorem 5. *Let $\boldsymbol{G} = \langle \boldsymbol{V}, \boldsymbol{ED} \rangle$ be a sound graph and let $\boldsymbol{L}$ be a complete event log. For any $v_i \in V$,*

1. *If $v_j \neq v_N$ and $v_j = de$ and there exists $(v_j, v_i) \in \boldsymbol{ED}$, then G' is an improved sound graph of $G/\{(v_j, v_i) | \forall_i v_i \wedge (v_j, v_i) \in \boldsymbol{ED}\}$.*
2. *If $v_j = v_N$ and $v_j = de$ and there exists $(v_j, v_i) \in \boldsymbol{ED}$, then G' is an improved sound graph of $G/\{(v_j, v_i) | \forall_i v_i \wedge (v_j, v_i) \in \boldsymbol{ED}\}$.*

Proof. Similar with previous theorems, we should prove the above two sub theorems respectively.

1. Assume $v_j \neq v_N$. For any $(v_j, v_i) \in ED \mid v_j, v_i \in V$, there exists at least one v_i to have causal relation with v_j to produce a sound process model and it has a possibility of $v_i = v_N$. In the case that $v_i = v_N$, according to the definition of path (Definition 5), a trace should have events at least two. Hence, a designated start of v_i which carry out property $v_i = v_0$ will lead to unsound process model. On the other hand, when $v_i \neq v_N$, the designated start of v_i will disconnect the relation between v_j and v_i and keep producing a sound model following the soundness property to have flow of a node either from start node or to end node. Hence, the theorem holds the soundness property.
2. Assume $v_j = v_N$. Although v_j is an end node, there could be a relation of $(v_j, v_i) \in ED \mid v_i, v_j \in V, \exists_k\, e_j^k \succ e_i^k$. If $v_j = v_N$, then $v_j \neq v_0$. If v_j is assigned as designated end, it should satisfy the end node as defined in Definition 2. As a result, the disconnection of the relation (v_j, v_i) will not lead to unsound process model since $\exists_k$ $e_l^k \succ e_j^k$ such that $l < j$. Hence, the theorem holds the soundness property. □

References

Agrawal, R., Gunopulos, D., & Leymann, F. (1998). Mining process models from workflow logs. In *Proceedings of the 6th international conference on extending database technology (EDBT), Valencia, Spain*.

Ankerst, M. (2002). Report on the SIGKDD-2002 panel the perfect data mining tool: interactive or automated? *ACM SIGKDD Explorations Newsletter, 4*(2), 110–111.
Barba, I., Weber, B., Valle, C. D., & Jimenez-Ramirez, A. (2013). User recommendations for the optimized execution of business processes. *Data & Knowledge Engineering, 86*, 61–84.
Bergenthum, R., Desel, J., Lorenz, R., & Mauser, S. (2007). Process Mining based on regions of languages. In *BPM 2007, LNCS* (Vol. 4714, pp. 375–383). Brisbane, Australia.
Bie, T. D. (2013). Subjective interestingness in exploratory data mining. *Chapter in Advances in Intelligent Data Analysis XII, 8207*, 19–31.
Cao, L., & Zhang, C. (2007). Domain-driven. *Actionable Knowledge Discovery, IEEE Intelligent Systems, 22*(4), 78–88.
Cao, L., Zhao, Y., Zhang, H., Luo, D., Zhang, C., & Park, E. K. (2010). Flexible frameworks for actionable knowledge discovery. *IEEE Transactions on Knowledge and Data Engineering, 22*(9), 1299–1312.
Carmona, J. (2012). Projection approaches to process mining using region-based techniques. *Data Mining and Knowledge Discovery, 24*(1), 218–246.
Carmona, J., & Cortadella, J. (2014). Process discovery algorithms using numerical abstract domains. *IEEE Transactions on Knowledge and Data Engineering, 26*(12), 3064–3076.
Cho, M., Song, M., & Yoo, S. (2015). A systematic methodology for outpatient process analysis based on process mining. *International Journal of Industrial Engineering: Theory, Applications, and Practices, 22*(4), 480–493.
Cook, J. E., Du, Z., & Liu, C. (2004). Discovering models of behavior for concurrent workflows. *Computers in Industry, 53*(3), 297–319.
de Medeiros, A. K. A., van Dongen, B. F., van der Aalst, W. M. P., & Weijters, A. J. M. M. (2004). Process mining for ubiquitous mobile systems: An overview and a concrete algorithm. *UMICS*, 151–165.
de Medeiros, A. K. A., Weijters, A. J. M. M., & van der Aalst, W. M. P. (2007). Genetic process mining: An experimental evaluation. *Data Mining and Knowledge Discovery, 14*(2), 245–304.
Fayyad, U., Shapiro, G., & Uthurusamy, R. (2003). Summary from the KDD-03 panel - Data mining: The next 10 years. *ACM SIGKDD Explorations Newsletter, 5*(2), 191–196.
Goedertier, S., Martens, D., Vanthienen, J., & Baesens, B. (2009). Robust process discovery with artificial negative events. *Journal of Machine Learning Research, 10*, 1305–1340.
Goedertier, S., Weerdt, J. D., Martens, D., Vanthienen, J., & Baesens, B. (2011). Process discovery in event logs: An application in the telecom industry. *Applied Soft Computing, 11*, 1697–1710.
Gottschalk, F., van der Aalst, W.M.P., & Jansen-Vullers, M.H. (2008) Mining reference process models and their configurations. In *International Workshop on enterprise integration, interoperability and networking. OTM 2008, LNCS* (Vol. 5333, pp. 263–272).
Greco, G., Guzzo, A., & Pontieri, L. (2012). Process discovery via precedence constraints. In Frontiers in artificial intelligence and applications: 20th European conference of artificial intelligence (pp. 366–371). Montpellier, France.
Gunther, C.W., & van der Aalst, W.M.P. (2007). Fuzzy mining – Adaptive process simplification based on multi-perspective metrics. In G. Alonso, P. Dadam, & M. Rosemann (Eds.), *BPM 2007, LNCS* (Vol. 4714, pp. 328-343).
Gunther, H. O., & Kim, K. H. (2005). *Container terminals and automated transport systems: Logistics control issues and quantitative decision support*.Berlin: Springer-Verlag.
Hwang, S. Y., & Yang, W. S. (2002). On the discovery of process models from their instances. *Decision Support Systems, 34*(1), 41–57.
Kawalek, P., & Kueng, P. (1997). The usefulness of process models: A lifecycle description of how process models are used in modern organizations. *International Workshop on Evaluation of Modelling Methods in Systems Analysis and Design*, 1–12.
La Rosa, M., Gottschalk, F., Dumas, M., & van der Aalst, W.M.P. (2008). Linking domain models and process models for reference model configuration. In B.B.A. ter Hofstede & H.-Y. Paik (Eds.), BPM 2007 Workshops LNCS (Vol. 4714. pp. 417–430).
Li, C., Reichert, M., & Wombacher, A. (2011). Mining business process variants: Challenges, scenarios, algorithms. *Data & Knowledge Engineering, 70*(5), 409–434.
Liu, B., Hsu, W., Chen, S., & Ma, Y. (2000). Analyzing the subjective interestingness of association rules. *Journal IEEE Intelligent Systems, 15*(5), 47–55.
Ly, L.T., Indiono, C., Mangler, J., & Rinderle-Ma, S. (2012). Data transformation and semantic log purging for process mining, In: International conference on advanced information systems engineering (CaISE) (pp. 238–253).
Pinter, S. S., & Golani, M. (2004). Discovering workflow models from activities' lifespans. *Computers in Industry, 53*(3), 283–296.
Rebuge, A., & Ferreira, D. R. (2012). Business process analysis in healthcare environments: A methodology based on process mining. *Information Systems, 37*, 99–116.
Rozinat, A., de Jong, I. S. M., Gunther, C. W., & van der Aalst, W. M. P. (2009). Process mining applied to the test process of wafer scanners in ASML. *IEEE Systems, Man and Cybernetics, 39*(4), 474–479.
Rozinat, A., de Medeiros, A., Gunther, C., Weitjers, A., & van der Aalst, W. M. P. (2007). Towards an evaluation framework for process mining algorithm. In *BPM center report BPM-07-06*. Eindhoven: Eindhoven University of Technology.
Salvagnin, D. (2008). *Constraint programming techniques for mixed integer linear programming* (p. 97). University of Padova, Padova: Department of Information Engineering.
Silberschatz, A., & Tuzhilin, A. (1995). On subjective measures of interestingness in knowledge discovery. In *KDD-95 Proceedings*.
Silberschatz, A., & Tuzhilin, A. (1996). What makes patterns interesting in knowledge discovery systems. *IEEE Transactions on Knowledge and Data Engineering, 8*(6), 970–974.
Solé, M., & Carmona, J. (2011). Light region-based techniques for process discovery. *Fundamenta Informaticae, 113*(3–4), 343–376.
Trcka, N., van der Aalst, W. M. P., & Sidorova, N. (2009). Workflow completion patterns. In *Proceeding of the fifth annual IEEE international conference on automation science and engineering* (pp. 7–12).
van der Aalst, W. M. P. (2011). *Process mining: Discovery, conformance and enhancement of business processes*.Berlin: Springer-Verlag.
van der Aalst, W. M. P. (2013). Decomposing Petri nets for process mining: A generic approach. *Distributed and Parallel Databases, 31*(4), 471–507.
van der Aalst, W. M. P., Dumas, M., Gottschalk, F., ter Hofstede, A. H. M., La Rosa, M., & Mendling, J. (2008). Correctness-preserving configuration of business process models. In J. L. Fiadeiro & P. Inverardi (Eds.), *Fundamental Approaches to Software Engineering. LNCS* (Vol. 4961, pp. 46–61).
van der Aalst, W. M. P., Reijers, H. A., Weijters, A. J. M. M., van Dongen, B. F., Alves de Medeiros, A. K., Song, M., & Verbeek, H. M. W. (2007). Business process mining: An industrial application. *Information Systems, 32*(5), 713–732.
van der Aalst, W. M. P., Weijters, A., & Maruster, L. (2004). Workflow mining: Discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering, 16*(9), 1128–1142.
van der Werf, J. M. E. M., van Dongen, B. F., van Hee, K. M., Hurkens, C. A. J., & Serebrenik, A. (2009). Process discovery using integer linear programming. *Journal Fundamenta Informaticae – Petri Nets, 94*(3–4), 387–412.
Van Dongen, B. F., & van der Aalst, W. M. P. (2005). A meta model for process mining data. In J. Casto & E. Teniente (Eds.), *CAiSE'05 Workshops (EMOI-INTEROP Workshop)* (pp. 309–320).
Wang, Y., Caron, F., Vanthienen, J., Huang, L., & Guo, Y. (2014). Acquiring logistics process intelligence: Methodology and an application for a Chinese bulk port. *Expert Systems with Applications, 41*, 195–209.
Weijters, A., van der Aalst, W.M.P., & de Medeiros, A. (2006). Process mining with heuristic miner algorithm. In *BETA Working Paper Series, WP 166*. Eindhoven University of Technology: Eindhoven.
Wen, L., van der Aalst, W. M. P., & Wang, J. (2007). Mining process models with non-free-choice constructs. *Data Mining and Knowledge Discovery, 15*(2), 145–180.
Yahya, B. N., Bae, H., Bae, J., & Kim, D. (2012a). Generating valid reference business process model using genetic algorithm. *International Journal of Innovative Computing, Information and Control, 8*(2), 1463–1477.
Yahya, B. N., Bae, H., Bae, J., & Liu, L. (2012c). Tool support for process modeling using proximity score measurement. *International Journal of Innovative Computing, Information and Control, 8*(7B), 5381–5399.
Yahya, B. N., Bae, H., Sul, S., & Wu, J.-Z. (2013). Process discovery by synthesizing activity proximity and user's domain knowledge. In M. Song, M. T. Wynn, & J. Liu (Eds.), *Asia Pacific conference on business process management 2013. LNBIP* (Vol. 159, pp. 92–105).
Yahya, B. N., Wu, J.-Z., & Bae, H. (2012b). Generation of business process reference model considering multiple objectives. *Industrial Engineering & Management Systems, 11*(3), 233–240.
Yang, Q., Yin, J., Ling, C., & Pan, R. (2007). Extracting actionable knowledge from decision trees. *IEEE Transactions on Knowledge and Data Engineering, 19*(1), 43–56.
Yoon, S., Henschen, L., Park, E., & Makki, S. (1999). Using domain knowledge in knowledge discovery. In *Processing of the eight international conference of information and knowledge management* (pp. 243–250).