Process Science

RESEARCH Open Access

Using translucent activity relationships frequencies to enhance process discovery

Harry H. Beyel[1*] and Wil M. P. van der Aalst[1]

*Correspondence:
beyel@pads.rwth-aachen.de

[1] Chair of Process and Data Science, RWTH Aachen University, Aachen, Germany

Abstract

Event logs are the primary source of information in process mining. An event log consists of events, each having three mandatory attributes: a case identifier, an activity, and a timestamp. Translucent event logs add a fourth mandatory attribute: *enabled* activities. These are activities for which execution was possible besides the executed activity. Information on enabled activities can be accessed in task management software, in tasks executed within a desktop environment, or by utilizing domain knowledge. Information on enabled activities is valuable for process discovery and other process mining tasks. For example, utilizing the information when discovering process models from translucent event logs results in more robust and generalizable process models. In recent work, translucent activity relationships were defined, and their application was shown by extending the inductive mining approach. This work extends the defined relationships by also introducing frequencies. As a result, the Inductive Miner—infrequent can be extended. Hence, it is possible to use these relationships in the presence of noise in the recorded event logs. We introduce three translucent variants for the Inductive Miner—infrequent, each using translucent activity relationships at different points. Moreover, we allow for different graphs when fall-throughs within the Inductive Miner occur. Using artificial and enriched real-life event logs, we show that considering these relationships, more data-specific process models that still have desired generalization capabilities can be discovered.

Keywords: Translucent process mining, Translucent event log, Process discovery, Inductive Miner, Inductive Miner–infrequent

Introduction

Process mining deals with analyzing event logs, which can be extracted from organizations' databases (van der Aalst 2016; Dumas et al. 2018). Process mining can be divided into three areas: *process discovery* (Augusto et al. 2019a), *conformance checking* (Carmona et al. 2018), and *process enhancement* (de Leoni 2022). Process discovery techniques aim to automatically discover a process model given an event log. Such a model aims to represent the underlying process comprehensively. Conformance checking describes and quantifies how well a model corresponds to an event log. Process enhancement combines a process model and an event log to extend or improve the provided model. Examples are decision mining and prediction. An event log, the data source for these techniques, is a collection of events, and each event consists of

(a) Choosing a proposal (b) Conducting reviews (c) Decision

Fig. 1 Example Graphical User Interface (GUI) and workflow for proposal approval

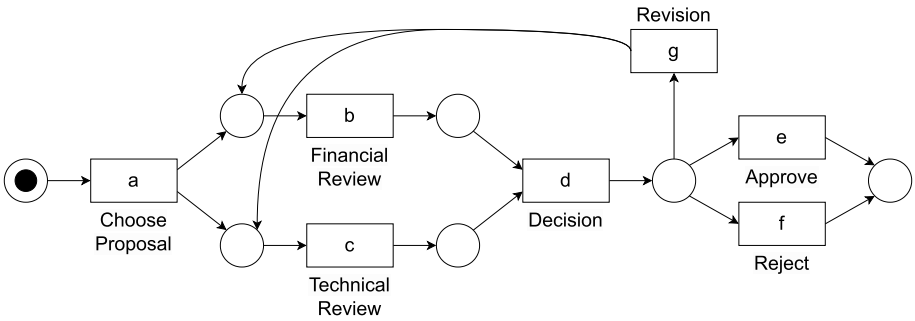


Fig. 2 Petri net describing the original workflow

at least three attributes: a *case identifier*, an *activity*, and a *timestamp*. Thus, event logs only capture what happened — not what *could* have happened. The information on *enabled activities* besides the executed action is valuable. Event logs that contain information on enabled activities are called *translucent event logs*.

Two prominent ways of receiving information on enabled activities are shown in Beyel and van der Aalst (2022) and Beyel et al. (2024). The first method involves injecting domain knowledge into an event log. For example, information on enabled activities can be added by replaying an event log on a process model. The second way is related to user interactions. When monitoring a user's interactions within a desktop environment, screenshots can be taken. In a screenshot, enabled activities besides the user's executed activity can be identified.

There is already work in the area of process discovery that uses information on enabled activities. A two-step method is presented in van der Aalst (2019). The method builds on state-based-region techniques and, thus, is not applicable to larger real-life datasets. In Beyel and van der Aalst (2024a), *translucent activity relationships* are introduced that are embedded in the Inductive Miner (Leemans et al. 2013a) using translucent directly-follows graphs derived from the relationships. To showcase the value of techniques that utilize information on enabled activities, consider the example desktop environment shown in Fig. 1. After choosing a proposal, a user has to submit a financial and technical review. When both are done, the proposal gets approved or rejected. Also, asking for a revision is possible. Note that we assume for this example that there is an assignment between users and proposals, i.e., one proposal can only be worked on by exactly one user. An overview of the workflow is shown in Fig. 2. By recording these interactions and taking screenshots of each interaction, the

approaches in Beyel and van der Aalst (2022) and Beyel et al. (2024) can create translucent event logs.

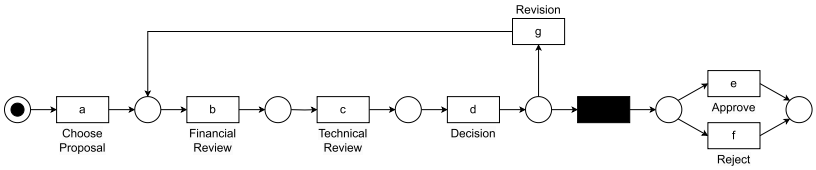
An example log based on this extraction is shown in Table 1. By considering only the information on executed activities, we observe that there are three cases $\langle a, b, c, d, e \rangle$, $\langle a, b, c, d, f \rangle$, and $\langle a, b, c, d, g, b, c, d, e \rangle$. To showcase the enabled activities in this representation, we write $\langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg} \rangle$, $\langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{ef}\, g \rangle$, $\langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{ef}\, \underline{g}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg} \rangle$. Applying the Inductive Miner (IM) (Leemans et al. 2013a) on the translucent event log results in the Petri net shown in Fig. 3a.

The workflow shown in the discovered model is purely sequential, contrasting the real process that allows for concurrency. However, when we consider the information on enabled activities, we denote that activities b and c are enabled when appearing, and c remains enabled after b's execution. This indicates that these activities could be parallel to each other. Applying the work presented in Beyel and van der Aalst (2024a) leads to discovering the Petri net depicted in Fig. 3b. As we can denote, the behavior is the same as in the original workflow (see Fig. 2).

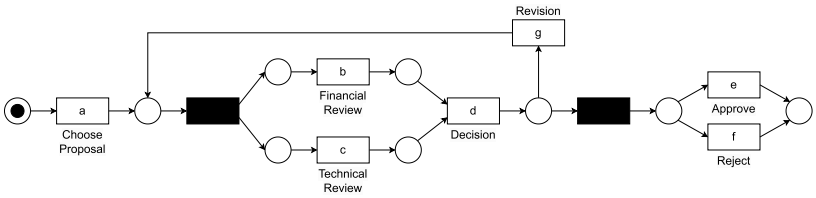
However, the approach in Beyel and van der Aalst (2024a) suffers from noise within the relationships. Let us assume the following event log has been extracted: $\langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg} \rangle$, $\langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{ef}\, g \rangle$, $\langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{ef}\, \underline{g}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg} \rangle$, $\langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{ef}\, \underline{g}, \underline{ef} \rangle$. This event log has an issue that can be caused by a wrong extraction from screenshots. For the last trace, after executing g, b and c should be enabled, and one of them should be executed. However, instead, e and f are enabled, and f is executed. When applying the work presented in Beyel and van der Aalst (2024a), the resulting model (see Fig. 4a) allows for much more behavior than captured in the log. Using the techniques shown in this work, we discover a model that represents the underlying process better (see Fig. 4b). As a result of this shortcoming, we extend (Beyel and van der Aalst 2024a)

Table 1 Example translucent event log

Event	Case	Activity	Enabled activities	Timestamp
e_1	1	a	{a}	2024 - 10- 20 13:37:37
e_2	1	b	{b, c}	2024 - 10- 20 13:37:38
e_3	1	c	{c}	2024 - 10- 20 13:37:39
e_4	1	d	{d}	2024 - 10- 20 13:37:40
e_5	1	e	{e, f, g}	2024 - 10- 20 13:37:41
e_6	2	a	{a}	2024 - 10- 20 13:40:37
e_7	2	b	{b, c}	2024 - 10- 20 13:40:38
e_8	2	c	{c}	2024 - 10- 20 13:40:39
e_9	2	d	{d}	2024 - 10- 20 13:40:40
e_{10}	2	f	{e, f, g}	2024 - 10- 20 13:40:41
e_{11}	3	a	{a}	2024 - 10- 20 13:45:37
e_{12}	3	b	{b, c}	2024 - 10- 20 13:45:38
e_{13}	3	c	{c}	2024 - 10- 20 13:45:39
e_{14}	3	d	{d}	2024 - 10- 20 13:45:40
e_{15}	3	g	{e, f, g}	2024 - 10- 20 13:45:41
e_{16}	3	b	{b, c}	2024 - 10- 20 13:47:38
e_{17}	3	c	{c}	2024 - 10- 20 13:47:39
e_{18}	3	d	{d}	2024 - 10- 20 13:47:40
e_{19}	3	e	{e, f, g}	2024 - 10- 20 13:47:42

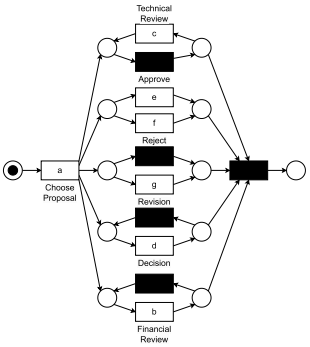


(a) Petri net discovered by the IM (Leemans et al. 2013a)

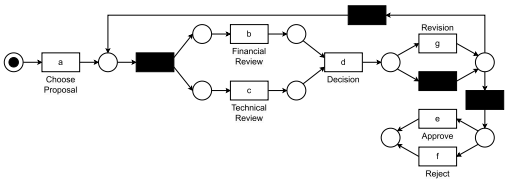


(b) Petri net discovered by utilizing the IMto (Beyel and van der Aalst 2024)

Fig. 3 Process models discovered on the traces $\langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg} \rangle, \langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{ef}\,\underline{g} \rangle, \langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{ef}\,\underline{g}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg} \rangle$. The example shows that IMto can discover the underlying process using less data



(a) Petri net discovered by utilizing the IMto (Beyel and van der Aalst 2024)



(b) Petri net discovered by utilizing the IMfto presented in this work

Fig. 4 Process Models discovered on the traces $\langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg} \rangle, \langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{ef}\,\underline{g} \rangle, \langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{ef}\,\underline{g}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg} \rangle$, $\langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{ef}\,\underline{g}, \underline{ef} \rangle$. Unlike the IMto, the IMfto can handle infrequent outliers

in multiple ways. First, we introduce symmetry in translucent activity relationships. Symmetry is used to construct the graphs needed for the discovery algorithms. Second, we introduce frequencies when incorporating translucent activity relationships in the Inductive Miner (Leemans et al. 2013a). In this process, we embed translucent exclusive-choice relationships that were not used before to filter out noise and allow for representing log behavior in more detail. Also, we utilize frequencies as the Inductive Miner—infrequent (Leemans et al. 2013b) does. Third, when fall-throughs happen, we allow for using the traditional directly-follows graph or our translucent directly-follows graph. As we show in our evaluation, by using translucent activity relationships, fewer process variants are needed to discover a well-representative process model that generalizes well. Furthermore, we present how exclusive-choice relationships change the underlying graph necessary for the Inductive Miner—infrequent.

The remainder of the work is structured as follows. In the Related work section, related work is presented. Subsequently, in the Preliminaries section, preliminaries for this work are presented and defined. We define translucent activity relationships in the Translucent activity relationships section, followed by the translucent Inductive Miner—infrequent in the Translucent Inductive Miner—infrequent section. We evaluate our algorithms in the Evaluation section and discuss our approach in the Discussion section. Finally, we provide a conclusion on our work in the Conclusion section.

Related work

This section is divided into two parts. First, we present related process-discovery techniques. Second, we dive into the current state of translucent process mining.

Process discovery

Process discovery focuses on uncovering a comprehensive process model that captures the underlying behavior reflected in a given event log (van der Aalst 2016). Numerous process-discovery techniques address this challenge, but all rely on identifying relationships between activities recorded in the event log. A prominent technique is the Inductive Miner (IM) (Leemans et al. 2013a). A key concept in IM is the use of directly-follows graphs (DFGs). If an activity follows an activity, an edge connects them. A DFG is partitioned through multiple cuts to create sublogs. For each partition, a DFG is generated, and the method is applied recursively. The IM is a discovery algorithm that provides guarantees that not all discovery techniques have, e.g., providing a sound workflow net as output or a perfect fitness w.r.t. to the log used as input. The Inductive Miner—infrequent (IMf) extends the IM. Infrequent behavior can be filtered out by keeping track of the number of occurrences of the directly-follow relationship. In this process, perfect fitness is no longer guaranteed. There are also other variants of the IM framework, e.g., an approximate variant (van Detten et al. 2023).

The Split Miner (Augusto et al. 2019b, 2020) also uses DFGs as input. The Split Miner follows six steps: First, it constructs a DFG and identifies self-loops and short-loops. Then, concurrency relations are discovered from the DFG. Next, the DFG is filtered to balance fitness and precision in the final process model. In the subsequent steps, split and join gateways are identified, and finally, OR-joins are refined into XOR or AND gateways.

Another approach is region-based process discovery. This discovery technique builds on region theory (Ehrenfeucht and Rozenberg 1990a, b). There are two approaches: language-based region theory (Mauser and Lorenz 2009) and state-based regions (van der Aalst et al. 2010; Cortadella et al. 1998; Solé and Carmona 2011, 2012, 2013). In general, state-based region process-discovery techniques are a two-step approach. First, a transition system is created. Second, minimal regions are extracted from the created transition system. Each minimal region corresponds to a place of a Petri net. Discovery techniques using region theory aim to find a fitting, precise process model.

For more information on various process-discovery techniques, we refer to van der Aalst (2016), van der Aalst (2022) and Augusto et al. (2022).

Translucent process mining

Translucent process mining deals with using information on enabled activities in the different areas of process mining. So far, works on creating translucent event logs, process discovery, and conformance checking have been published.

As described earlier, two approaches for receiving translucent event logs are described in Beyel and van der Aalst (2022). The first approach uses a Petri net and an event log as input. Based on the replay of the traces, enabled activities are identified at each step. This approach can be used to utilize domain knowledge to add information on enabled activities since the model does not have to be discovered. The second approach builds on template matching (Brunelli 2009). An interaction log is created by recording user interactions, where each entry is linked to a screenshot showing the state of the screen. It is checked whether a labeled template is contained in the screenshot, and if so, the label is added as enabled activity. In Beyel et al. (2024), an automatic framework for extracting enabled activities from screenshots is presented. The framework uses more advanced techniques to detect elements of interest within a screenshot and labels them.

The previously mentioned process discovery technique deals with classic event data. Event data in translucent event logs have an additional attribute, enabled activities, allowing sophisticated techniques. Two techniques have been proposed for process discovery using translucent event logs.

In van der Aalst (2019), a state-based region approach is used. Each set of enabled activities in a translucent event log represents its own state. Arcs connecting these states are labeled with the executed activity that transitions one set to another, as indicated by the order of events within a case in the translucent event log. In the final step, this system is transformed into a Petri net. However, since this approach relies on state-based regions and thus falls under region-based process discovery, it may encounter similar problems, such as extended computation times or overly complex models.

In Beyel and van der Aalst (2024a), the foundation of this work, successive events within traces in a translucent event log are used to derive translucent activity relationships that extend directly-follows graphs of the IM. The work defined three activity relationships: *translucent directly-follow relationship*, *translucent parallel relationship*, and *translucent exclusive-choice relationship*. Each relationship checks the executed activity and enabled activities of an event and the enabled activities of the subsequent event. The executed activity has a translucent directly-follow relationship with all subsequent enabled activities. Also, the executed activity has a translucent parallel relationship with all activities that were enabled during and after the execution. Finally, the executed activity has a translucent exclusive-choice relationship with all activities that were enabled during and are not enabled after the execution. Also, all activities at the start or end of traces are considered translucent start or end activities. The translucent directly-follow and parallel relationship, as well as translucent start or end activities, are used to add arcs in the DFG used by the IM (Leemans et al. 2013a). Three variants of the IM are introduced in Beyel and van der Aalst (2024a) that use the information at various points. The first variant only uses the DFG enriched with information on enabled activities (*IMto*). The second variant first uses the DFG enriched with information on enabled activities, and if no cut is detected, a classic DFG is used (*IMtf*). The third variant first uses the classic DFG, and if no cut is detected, a DFG enriched with information on enabled activities

is used (*IMts*). As shown in Beyel and van der Aalst (2024a), when only a fraction of the data is available, incorporating these relationships in the discovery process helps to find more robust process models. This is of high interest when considering that only a fraction of data from a system is used to discover the processes contained in the data.

In this work, we extend the approach presented in Beyel and van der Aalst (2024a) in several ways. First, we define symmetric relationships. These relationships allow for capturing a holistic view of activity relationships. Second, we introduce frequencies to allow for more robust discovery. Moreover, by utilizing frequencies of the occurrence of these relationships in the data, we can utilize the translucent exclusive-choice relationship, which has been unused before. We use them to filter out noise in translucent directly-follows and parallel relationships. Consequently, this work unlocked a greater potential for using translucent activity relationships than the previous work. In addition, when utilizing frequencies, we allow for filtering infrequent relationships between activities, similar to the Inductive Miner—infrequent (IMf) (Leemans et al. 2013b). Third, we allow users to choose a classic DFG or a DFG enriched with information on enabled activities when applying fall-throughs.

In Beyel and van der Aalst (2024 d), a precision metric for Petri nets and translucent event logs is presented. The metric is based on escaping arcs (Munoz-Gama and Carmona 2010). Parallelism in a model lowers the precision score by introducing more escaping arcs if not covered in classical event logs. Thus, the mentioned work involves enabled activities in the computation to allow for some escaping arcs if they are present in the translucent event log.

Preliminaries

In this section, we define the preliminaries of our work. We start with basic notations.

Definition 1 (Basic Notations). *Given a set X, $\mathcal{B}(X)$ denotes the set of all multisets over set X. E.g., if $X = \{x, y, z\}$, a possible bag is $[x, x, y] = [x^2, y]$. The conjunction of two multisets is denoted with $\uplus$, e.g., $[x^2, y] \uplus [x, z] = [x^3, y, z]$. Given a multiset $M \in \mathcal{B}(X)$, $|M|$ denotes the size of the multiset. We define $|[\,]| = 0$, and $|A \uplus [x]| = |A| + 1$. For example, $[x^3, y, z] = 5$. We denote the Cartesian product of two sets X and Y as $X \times Y = \{(x, y) \mid x \in X \wedge y \in Y\}$. Given a set X, a sequence $\sigma \in X^*$ of length $n \in \mathbb{N}$ assigns an enumeration to elements of the set, i.e., $\sigma : \{1, ..., n\} \to X$. We denote this with $\sigma = \langle \sigma_1, \ldots, \sigma_n \rangle$, i.e., σ_i to the sequence's i-th element. $\langle\rangle$ is the empty sequence. Given a sequence σ, $|\sigma|$ denotes the length of a sequence. Given two sequences $\sigma = \langle \sigma_1, \ldots, \sigma_{|\sigma|} \rangle$ and $\sigma' = \langle \sigma'_1, \ldots, \sigma'_{|\sigma'|} \rangle$, a conjunction is denoted as $\sigma \cdot \sigma' = \langle \sigma_1, \ldots, \sigma_{|\sigma|}, \sigma'_1, \ldots, \sigma'_{|\sigma'|} \rangle$. A sequence σ' is a subsequence of sequence σ, written $\sigma' \sqsubseteq \sigma$, if $\sigma' = \langle \sigma_l, \sigma_{l+1}, \ldots, \sigma_m \rangle$ and $1 \leq l < m \leq |\sigma|$. Given $\sigma \in X^*$, we define $\langle \sigma_0, \ldots, \sigma_0 \rangle = \langle\rangle$, and $\langle \sigma_0, \ldots, \sigma_n \rangle = \langle \sigma_1, \ldots, \sigma_n \rangle$.*

We consider translucent event logs as input for our process-discovery techniques, i.e., for each event, information on enabled activities besides the executed activity is available. Important to note is that the executed activity of an event is also an enabled activity in the corresponding event.

Definition 2 (Translucent Event Log). *$\mathcal{U}_{act}$ is the universe of activity names. An event e is a tuple $e = (en, act) \in \mathcal{P}(\mathcal{U}_{act}) \times \mathcal{U}_{act}$ such that $act \in en$. We define $\pi_{en}(e) = en$ and $\pi_{act}(e) = act$. A trace $\sigma = \langle e_1, \dots, e_n \rangle \in (\mathcal{P}(\mathcal{U}_{act}) \times \mathcal{U}_{act})^*$ is a sequence of events. A translucent event log $L \in \mathcal{B}((\mathcal{P}(\mathcal{U}_{act}) \times \mathcal{U}_{act})^*)$ is a multiset of traces. For simplicity, we define $\pi_{act}(L) = \bigcup_{\sigma \in L} \bigcup_{e \in \sigma} \{\pi_{act}(e)\}$. Moreover, we assume $\bigcup_{\sigma \in L} \bigcup_{e \in \sigma} \pi_{en}(e) = \pi_{act}(L)$. Furthermore, $|L|$ denotes the number of traces in L.*

Let $L_{ex} = [\sigma_1, \sigma_2, \sigma_3] = [\langle (\{a\}, a), (\{b, c\}, b), (\{c\}, c), \dots \rangle, \dots]$ be an example translucent event log. Then, $\pi_{act}((\{b, c\}, b)) = b$ and $\pi_{en}((\{b, c\}, b)) = \{b, c\}$. In addition, $\pi_{act}(L_{ex}) = \{a, b, c, \dots\}$. In the remainder of this work, we use a shorthand notation for which list all activities and underline the executed activity: $L_{ex} = [\langle \underline{a}, \underline{bc}, \underline{c}, \dots \rangle, \dots]$. Also, $|L_{ex}| = 3$.

Furthermore, we define the projection of translucent traces. In addition to the classical projection of executed activities, the enabled activities are also projected. Still, an event is not projected if the executed activity is not part of the given set of activities. The following defines that.

Definition 3 (Translucent Trace Projection). *Let $L \in \mathcal{B}((\mathcal{P}(\mathcal{U}_{act}) \times \mathcal{U}_{act})^*)$ be a translucent event log and $\langle (X_1, x_1), (X_2, x_2), \dots, (X_{|\sigma|}, x_{|\sigma|}) \rangle = \sigma \in L$ be a translucent trace. Let $A \subseteq \pi_{act}(L)$ be a set of activities. $\sigma \restriction_A$ is the translucent projection of σ onto a set A, i.e., $\sigma \restriction_A = \langle (X_i \cap A, x_i) \mid (X_i, x_i) \in \sigma \ \wedge \ x_i \in A \rangle$.*

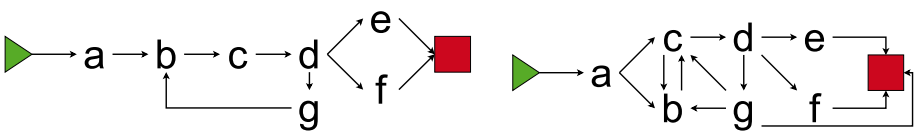
For example, $\langle (\{a\}, a), (\{b, c\}, b), (\{c\}, c) \rangle \restriction_{\{a,b\}} = \langle (\{a\}, a), (\{b\}, b) \rangle$.

Our work extends the IMf (Leemans et al. 2013b) to consider information on enabled activities. To allow for the incorporation of this information, we introduce directly-follows graphs, translucent directly-follows graphs from Beyel and van der Aalst (2024a), and shortly explain the IMf.

A directly-follows graph is derived from a (translucent) event log and shows three aspects: start activities, end activities, and directly-follows relationships between activities recorded in the provided event log. Directly-follows relationships show which activities directly-follows an activity over all traces in an event log. The information is collected in a graph where nodes represent start and end and the activities in a log. Edges connect the start node and activity nodes if traces start with them. Edges connect activity nodes with the end node if traces and with them. Two activities are connected if one activity is followed by another activity.

Definition 4 (Directly-Follows Graph). *Let $L \in \mathcal{B}((\mathcal{P}(\mathcal{U}_{act}) \times \mathcal{U}_{act})^*)$ be a translucent event log with $\blacksquare \notin \pi_{act}(L)$ and $\blacktriangleright \notin \pi_{act}(L)$. A Directly-Follows Graph (DFG) of L is a directed graph $DFG^L = (V, E)$ for which $V = \pi_{act}(L) \cup \{\blacktriangleright, \blacksquare\}$, where $\blacktriangleright$ is the start node, $\blacksquare$ is the end node, and $E \subseteq (\{\blacktriangleright\} \times \bigcup_{\sigma \in L} \{\pi_{act}(\sigma_1)\}) \cup (\bigcup_{\sigma \in L} \{\pi_{act}(\sigma_{|\sigma|})\} \times \{\blacksquare\})$ $\cup (\{\blacktriangleright\} \times \{\blacksquare\}) \cup (\bigcup_{\sigma \in L} \bigcup_{1 \leq i < |\sigma|} \{(\pi_{act}(\sigma_i), \pi_{act}(\sigma_{i+1}))\})$. Note that the empty trace ($\langle\rangle$) can be captured within this graph.*

The DFG of the translucent event log $L = [\langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg} \rangle, \langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{ef}\,\underline{g} \rangle, \langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{ef}\,\underline{g}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg} \rangle]$ is provided in Fig. 5a. As denoted in the DFG, activity a is start activities, and e and f are end activities. Furthermore, a is connected to b, and b is connected to c, and c to d. d is connected to e, f, and g. g is connected to b.



(a) DFG (b) tDFG

Fig. 5 DFG and tDFG obtained from translucent event log $L = [\langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg} \rangle, \langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg} \rangle, \langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg} \rangle]$

A translucent DFG extends the former DFG. First, starting from the executed activity of an event, arcs are added to all enabled activities of the subsequent event. Suppose some activities were enabled during and after the execution of an activity. In that case, arcs are also added from them to the executed activity since this can indicate parallelism (the execution did not influence them). In addition, arcs from the start node to all enabled activities at the beginning of traces are added. Similarly, arcs for the enabled activities at the end of traces to the end node are added. The following captures this.

Definition 5 (Translucent Directly-Follows Graph). *Let* $L \in \mathcal{B}((\mathcal{P}(\mathcal{U}_{act}) \times \mathcal{U}_{act})^*)$ *be a translucent event log with* $\blacksquare \notin \pi_{act}(L)$ and $\blacktriangleright \notin \pi_{act}(L)$. *A translucent Directly-Follows Graph (tDFG) of* L *is a directed graph* $DFG^L = (V, E)$ *for which* $V = \pi_{act}(L) \cup \{\blacktriangleright, \blacksquare\}$, *where* $\blacktriangleright$ *is the start node,* $\blacksquare$ *is the end node, and* $E \subseteq (\{\blacktriangleright\} \times \bigcup_{\sigma \in L} \pi_{en}(\sigma_1)) \cup (\bigcup_{\sigma \in L} \pi_{en}(\sigma_{|\sigma|}) \times \{\blacksquare\}) \cup (\{\blacktriangleright\} \times \{\blacksquare\}) \cup (\bigcup_{\sigma \in L} \bigcup_{1 \leq i < |\sigma|} \{\pi_{act}(\sigma_i)\} \times \pi_{en}(\sigma_{i+1})) \cup (\bigcup_{\sigma \in L} \bigcup_{1 \leq i < |\sigma|} \{\pi_{act}(\sigma_i)\} \times (\pi_{en}(\sigma_i) \cap \pi_{en}(\sigma_{i+1})))$. *Note that the empty trace* ($\langle\rangle$) *can be captured within this graph.*

A tDFG is displayed in Fig. 5b. As shown there, the parallelism between activities b and c is captured, as well as the option to execute c after g.

The IMf (Leemans et al. 2013b) is a well-established state-of-the-art process discovery algorithm. A high-level overview of the steps performed in the IMf is shown in Fig. 6. The IMf operates as follows: if the given event log is a base case, a process model is generated. If not, a DFG is constructed, and the algorithm checks for the presence of a cut in the graph. There are four types of cuts: *exclusive-choice* ($\times$), *sequence* ($\rightarrow$), *parallel* ($+$), and *redo-loop* ($\circlearrowleft$). These cuts analyze multiple arcs in the DFG, capturing the relationships between various activities. When a cut is detected, the event log is projected according to the activity partitions created by the cut, with each activity belonging to exactly one partition. The process continues until the final partition contains a single activity, at which point a base case is reached. If no cut is found, frequencies of the directly-follow relationships in the graph are incorporated, and less frequent occurrences are filtered out by setting a user-specific threshold. For each node, the maximum frequent edge is denoted, and outgoing edges that are below a threshold influenced by the maximum frequency are removed. Hence, the discovered process models are less likely to be influenced by rare behavior. If no cut is

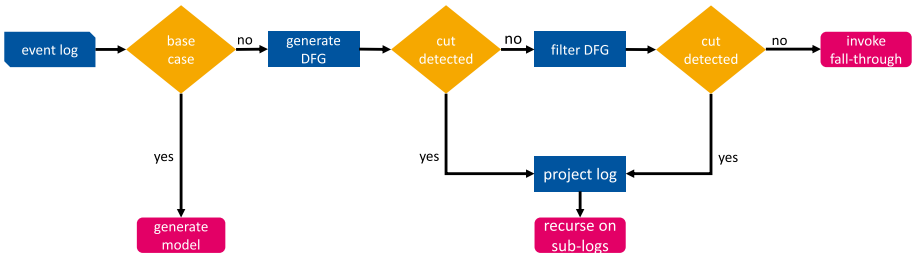
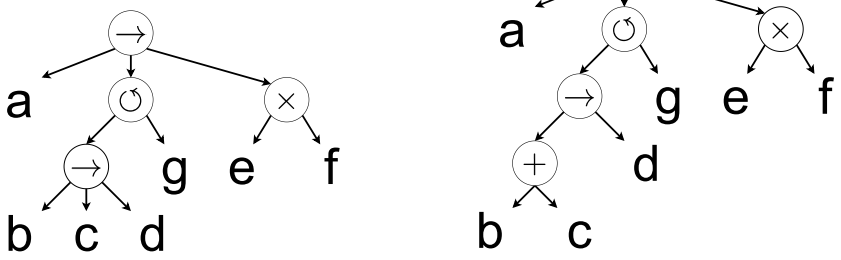


Fig. 6 High-level overview of the Inductive Miner—infrequent (Leemans et al. 2013b)



(a) Process tree created by utilizing traditional IM (Leemans et al. 2013a)

(b) Process tree created by utilizing IMto (Beyel and van der Aalst 2024)

Fig. 7 Process trees obtained from translucent event log $L = [\langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg} \rangle, \langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg} \rangle, \langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg} \rangle]$

detected, fall-through strategies are applied. For further details, we refer to Leemans et al. (2013a) and Leemans et al. (2013b). In the following, we define a cut.

Definition 6 (Cut). *Let* $L \in \mathcal{B}((\mathcal{P}(\mathcal{U}_{act}) \times \mathcal{U}_{act})^*)$ *be a translucent event log. A cut* $(\oplus, A_1, \ldots, A_n)$ *of* L *is a tuple of a control flow operator* $\oplus \in \{\times, \rightarrow, +, \circlearrowleft\}$ *and a partitioning of activities into* $n \geq 2$ *disjoint subsets, i.e.,* $\pi_{act}(L) = \bigcup_{i \in \{1,\ldots,n\}} A_i$ *and* $A_i \cap A_j = \emptyset$ *for* $1 \leq i < j \leq n$.

To visualize the cuts and, thereby, the output produced by using inductive mining approaches, we showcase two process trees in Fig. 7 based on the graphs displayed in Fig. 7. The first cut of both trees can be written as $(\rightarrow, \{a\}, \{b, c, d, g\}, \{e, f\})$. Note that both trees can be translated to Petri nets (see Fig. 3a for the Petri net based on Fig. 7a and Fig. 3b for the Petri net based on Fig. 7b).

In this work, we enhance DFGs to capture translucent information and the frequencies between translucent activity relationships and thus allow the IMf (Leemans et al. 2013b) to consider the knowledge of enabled activities. In the remainder of this work, we show Petri nets as process model representations. An introduction to Petri nets is provided in van der Aalst (2016) and Reisig (1985).

Translucent activity relationships

Translucent activity relationships are crucial in integrating information about enabled activities into process mining. To discover these relationships, we employ a sequence-based approach. This involves traversing the cases within the provided translucent event log and collecting the activity relationships by examining the executed activity of an event along with its enabled activities and those of the subsequent event. As in Beyel and van der Aalst (2024a), we introduce three types of translucent frequent activity relationships: *directly-follows*, *parallel*, and *exclusive-choice*. Additionally, we specify translucent start and end activities. For each relationship, we provide a detailed explanation of its meaning. We extend the former work by introducing the symmetry of parallel and exclusive choice relationships. Moreover, we introduce frequency to all three relationships, similar to other discovery algorithms. In the remainder of this section, we use $L = [\langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg} \rangle, \langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{ef}\,\underline{g} \rangle, \langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{ef}\,\underline{g}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg} \rangle, \langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{ef}\,\underline{g}, \underline{ef} \rangle]$ as example translucent event log.

Translucent directly-follows relationship frequencies

The translucent directly-follows relationship extends the traditional directly-follows relationship. It accounts for all enabled activities following the execution of an activity, meaning that self-loops are included, even when absent from the sequences of executed activities. Given the example log, there are some translucent directly-follows relationships, for example, between choosing a proposal (a) and conducting both reviews (b and c). We count how often this relationship occurs between two activities. The following captures that.

Definition 7 (Translucent Directly-Follows Relationship Frequency). *Let $L \in \mathcal{B}((\mathcal{P}(\mathcal{U}_{act}) \times \mathcal{U}_{act})^*)$ be a translucent event log and $a, b \in \pi_{act}(L)$ be activities. The frequency of the occurrence of the translucent directly-follows relationship between activities a and b is defined as*

$$df^L_{freq}(a,b) = \sum_{\sigma \in L} \sum_{1 \leq i < |\sigma|} \begin{cases} 1, & \pi_{act}(\sigma_i) = a \ \wedge \ b \in \pi_{en}(\sigma_{i+1}) \\ 0, & \textit{otherwise} \end{cases}$$

For our example log, we denote the following translucent directly-follows relationships frequencies: $df^L_{freq}(a,b) = 4$ and $df^L_{freq}(a,d) = 0$, $df^L_{freq}(g,f) = 1$, $df^L_{freq}(g,e) = 1$.

Translucent parallel relationship frequencies

The translucent parallel relationship builds upon the previously introduced translucent directly-follows relationship. By leveraging information about the executed activity, the event's enabled activities, and the succeeding enabled activities, we identify which activities run parallel with the executed activity. The idea is that the execution of an activity does not influence the possible execution of other activities. Given the example log, this would be the case when conducting the reviews (b and c) but also once for revision, approval, and rejection (g, e, and f). As with the directly-follows relationship, we count how often we observe this relationship in a given event log. The following captures this.

Definition 8 (Translucent Parallel Relationship Frequency). *Let $L \in \mathcal{B}((\mathcal{P}(\mathcal{U}_{act}) \times \mathcal{U}_{act})^*)$ be a translucent event log and $a, b \in \pi_{act}(L)$ be activities. The frequency of the occurrence of the translucent parallel relationship between activities a and b is defined as*

$$par^L_{freq}(a,b) = \sum_{\sigma \in L} \sum_{1 \leq i < |\sigma|} \begin{cases} 1, \ \pi_{act}(\sigma_i) = a \ \wedge \ b \in \pi_{en}(\sigma_i) \ \wedge \ b \in \pi_{en}(\sigma_{i+1}) \\ 0, \ otherwise \end{cases}$$

For the example translucent event log, we denote the following translucent parallel relationship frequencies: $par^L_{freq}(a,b) = 0$, $par^L_{freq}(b,c) = 5$, and $par^L_{freq}(c,b) = 0$, $par^L_{freq}(g,e) = 1$, $par^L_{freq}(g,f) = 1$.

In traditional discovery approaches, not using translucent, concurrency is symmetric. This is also natural. Therefore, we extend the former definition.

Definition 9 (Translucent Symmetric Parallel Relationship Frequency). *Let $L \in \mathcal{B}((\mathcal{P}(\mathcal{U}_{act}) \times \mathcal{U}_{act})^*)$ be a translucent event log, $a, b \in \pi_{act}(L)$* be activities, and *par^L_{freq} as defined before. The frequency of the occurrence of the translucent symmetric parallel relationship between activities a and b is defined as $par^L_{freq\_sym}(a,b) = par^L_{freq}(a,b) + par^L_{freq}(b,a)$. Note that now $par^L_{freq\_sym}(a,b) = par^L_{freq\_sym}(b,a)$.*

For the example log, we denote the following translucent symmetric parallel relationship frequencies: $par^L_{freq\_sym}(a,b) = 0$, $par^L_{freq\_sym}(b,c) = 5$, and $par^L_{freq\_sym}(c,b) = 5$, $par^L_{freq}(g,e) = 1$, $par^L_{freq}(g,f) = 1$.

Translucent exclusive-choice relationship frequencies

The concept of the translucent exclusive-choice relationship is similar to that of translucent parallel relationships. However, rather than focusing on the "remaining" activities, we concentrate on the "removed" activities. To identify these "removed" activities, we utilize the information about the executed activity of an event and the difference between the event's enabled activities and the enabled activities of the succeeding event. Given the example log, there is a translucent exclusive-choice relationship between *e*, *f*, and *g*. The following captures that.

As before, we count how often we observe translucent exclusive-choice relationships.

Definition 10 (Translucent Exclusive-Choice Relationship Frequency). *Let $L \in \mathcal{B}((\mathcal{P}(\mathcal{U}_{act}) \times \mathcal{U}_{act})^*)$ be a translucent event log and $a, b \in \pi_{act}(L)$ be activities. The frequency of the occurrence of the translucent exclusive-choice relationship between activities a and b is defined as*

$$exc^L_{freq}(a,b) = \sum_{\sigma \in L} \sum_{1 \leq i < |\sigma|} \begin{cases} 1, \ \pi_{act}(\sigma_i) = a \ \wedge \ b \in \pi_{en}(\sigma_i) \ \wedge \ b \notin \pi_{en}(\sigma_{i+1}) \\ 0, \ otherwise \end{cases}$$

For the example translucent event log, we denote the following translucent exclusive-choice relationship frequencies: $exc^L_{freq}(a,b) = 0$, $exc^L_{freq}(b,d) = 0$, and $exc^L_{freq}(g,e) = 1$, $exc^L_{freq}(g,e) = 1$, $exc^L_{freq}(e,g) = 0$.

As before, we want to make the relationship symmetric since this seems natural. Therefore, we extend the frequency notation.

Definition 11 (Translucent Symmetric Exclusive-Choice Relationship Frequency). *Let $L \in \mathcal{B}((\mathcal{P}(\mathcal{U}_{act}) \times \mathcal{U}_{act})^*)$ be a translucent event log, $a, b \in \pi_{act}(L)$ be activities, and exc^L and exc_{freq} as defined before. The frequency of the occurrence of the translucent symmetric parallel relationship between activities a and b is defined as $exc^L_{freq\_sym}(a,b) = exc^L_{freq}(a,b) + exc^L_{freq}(b,a)$. Note that now $exc^L_{freq\_sym}(a,b) = exc^L_{freq\_sym}(b,a)$.*

For L_{ex}, we denote the following translucent symmetric exclusive-choice relationship frequencies: $exc^L_{freq\_sym}(a,b) = 0$ and $exc^L_{freq\_sym}(e,g) = 1$.

Translucent start and end activities

In addition to the previously defined translucent activity relationships, we introduce translucent start and end activities.

Translucent start activities are all enabled activities in traces' first events and executed at some point in the log.

We also count how often an activity is a translucent start activity.

Definition 12 (Translucent Start Activity Frequency). *Let $L \in \mathcal{B}((\mathcal{P}(\mathcal{U}_{act}) \times \mathcal{U}_{act})^*)$ be a translucent event log. For an activity $act \in \pi_{act}(L)$, the following defines a counter on how often act is a translucent start activity:*

$$Start^L_{freq}(act) = \sum_{\sigma \in L} \begin{cases} 1, & act \in \pi_{en}(\sigma_1) \\ 0, & otherwise \end{cases}$$

For L_{ex}, we denote $Start^{L_{ex}}(a) = 4$ and $Start^{L_{ex}}(e) = 0$.

Similar to translucent start activities, translucent end activities are all enabled activities of traces' last events of a translucent event log. Also, they have to be executed at some point in the log.

Again, we denote also frequencies for end activities. The following captures that.

Definition 13 (Translucent End Activity Frequency). *Let $L \in \mathcal{B}((\mathcal{P}(\mathcal{U}_{act}) \times \mathcal{U}_{act})^*)$ be a translucent event log. For an activity $act \in \pi_{act}(L)$, the following defines a counter on how often act is a translucent start activity:*

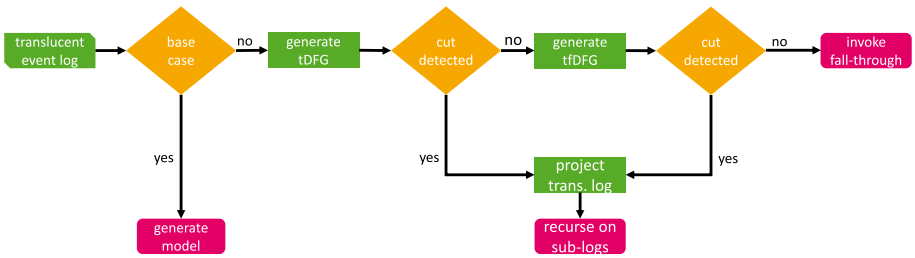
$$End^L_{freq}(act) = \sum_{\sigma \in L} \begin{cases} 1, & act \in \pi_{en}(\sigma_{|\sigma|}) \\ 0, & otherwise \end{cases}$$

For L_{ex}, we denote $End^{L_{ex}}_{freq}(a) = 0$, $End^{L_{ex}}_{freq}(e) = 4$, and $End^{L_{ex}}_{freq}(g) = 3$.

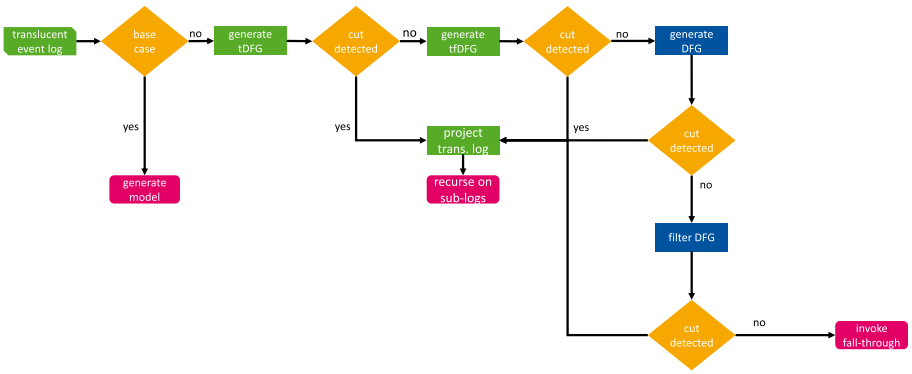
Translucent Inductive Miner—infrequent

Translucent relationships represent a general concept that can be incorporated into various process discovery algorithms. In this section, we show an approach inspired by the Inductive Miner–infrequent (Leemans et al. 2013b).

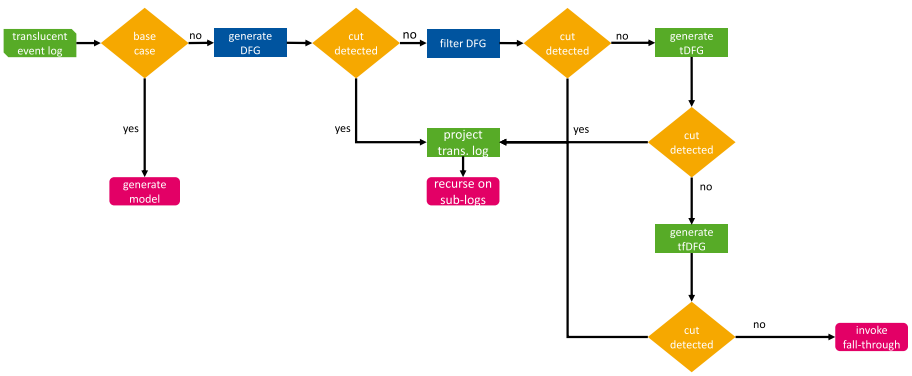
Similar to the translucent Inductive Miner presented in Beyel and van der Aalst (2024a), that creates a translucent DFG (tDFG) as we have shown before, we also incorporate translucent relationships and their frequency within the Inductive Miner—infrequent (IMf) (Leemans et al. 2013b). To do so, we use translucent frequent DFGs (tfDFGs). As in Beyel and van der Aalst (2024a), we define three approaches, illustrated in Fig. 8. The first approach (*IMfto*), shown in Fig. 8a, creates a tDFG if there is no base case. If no cut is found, a tfDFG is created. This approach only used



(a) IMf only using tDFGs (IMfto)



(b) IMf first using tDFG, then DFG (IMftf)



(c) IMf first using DFG, then tDFG (IMfts)

Fig. 8 Abstract overview of the different approaches for the IMf

information on enabled activities. The second approach (*IMftf*), showcased in Fig. 8b, starts with a base case. If there is no base case, a tDFG is created. If no cut is found, a tfDFG is created. If no cut is detected, a classic DFG is created. If again no cut is found, the DFG gets filtered by frequency as defined by the IMf (Leemans et al. 2013b). The third approach (*IMfts*) is similar to the second, but information on enabled activities is considered after the classical approach fails. In the case of fall-throughs, we allow for the usage of translucent and classic DFGs. In the following, we show how to incorporate frequencies and how the splits in the frequency setting are defined.

Translucent frequent DFG

As the tDFG, introduced in Beyel and van der Aalst (2024a) and presented earlier, the translucent frequent DFG (tfDFG) utilizes information from translucent relationships. While the translucent exclusive-choice relationship is unused in the tDFG, it is used in this approach to lower connection frequencies. This further filters out noise. In the following, we define the set of arcs that are later used in our tfDFG.

First, we define *frequent translucent directly-follow arcs*. It is counted as how often a translucent directly-follow relationship between two activities occurs in a given translucent event log. The count is subtracted by the number of times an exclusive-choice relationship was observed between these two activities to filter out noise. As for the IMf (Leemans et al. 2013b), a user can define a threshold f. Only relationships between two activities are kept if they occur frequently enough.

Definition 14 (Frequent Translucent Directly-Follow Arcs). *Let $L \in \mathcal{B}((\mathcal{P}(\mathcal{U}_{act}) \times \mathcal{U}_{act})^*)$ be a translucent event log, and $a, b \in \pi_{act}(L)$. We define $\rightarrow^L_{freq}(a,b) = df^L_{freq}(a,b) - exc^L_{freq\_synch}(a,b)$. Let $0 \leq f \leq 1$. The set of frequent translucent directly-follows arcs is defined as $E^{L,f}_{df} = \{(a,b) \in \pi_{act}(L) \times \pi_{act}(L) \mid \rightarrow^L_{freq}(a,b) > 0 \wedge \rightarrow^L_{freq}(a,b) > f \cdot \max_{c \in \pi_{act}(L)} \rightarrow^L_{freq}(a,c)\}$.*

Let $L = [\langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg} \rangle, \langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{ef}\,\underline{g} \rangle, \langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{ef}\,\underline{g}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg} \rangle, \langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{ef}\,\underline{g}, \underline{ef} \rangle]$ be again the previous example translucent event log and $f = 0$. Then we denote $\rightarrow^L_{freq}(a,b) = 4, \rightarrow^L_{freq}(b,c) = 5, \rightarrow^L_{freq}(c,b) = 0$, and $\rightarrow^L_{freq}(g,e) = 1 - 1 = 0$.

Second, we define *frequent translucent parallel arcs*. As with the translucent directly-follows relationships, we count again how often a parallel relationship occurs. To count this, we used the previously introduced symmetric count. Again, we subtract by the number of observed exclusive-choice relationships to filter out noise. As for the IMf (Leemans et al. 2013b), and as previously mentioned, a user can define a threshold f for which behavior is filtered out.

Definition 15 (Frequent Translucent Parallel Arcs). *Let $L \in \mathcal{B}((\mathcal{P}(\mathcal{U}_{act}) \times \mathcal{U}_{act})^*)$ be a translucent event log, and $a, b \in \pi_{act}(L)$. We define $+^L_{freq}(a,b) = par^L_{freq\_sym}(a,b) - exc^L_{freq\_synch}(a,b)$. Let $0 \leq f \leq 1$. The set of frequent translucent parallel arcs is defined as $E^{L,f}_{par} = \{(a,b) \in \pi_{act}(L) \times \pi_{act}(L) \mid +^L_{freq}(a,b) > 0 \wedge +^L_{freq}(a,b) > f \cdot \max_{c \in \pi_{act}(L)} +^L_{freq}(a,c)\}$.*

Let $L = [\langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg} \rangle, \langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{ef}\,\underline{g} \rangle, \langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{ef}\,\underline{g}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg} \rangle, \langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{ef}\,\underline{g}, \underline{ef} \rangle]$ be again the previous example translucent event log and $f = 0$. Then we denote $+^L_{freq}(a, b) = 0, +^L_{freq}(b, c) = 5, +^L_{freq}(c, b) = 5$, and $+^L_{freq}(g, e) = 1-1 = 0$.

Third, we define *frequent translucent start arcs*. Frequent translucent start arcs are defined similarly to the IMf (Leemans et al. 2013b).

Definition 16 (Frequent Translucent Start Arcs). *Let $L \in \mathcal{B}((\mathcal{P}(\mathcal{U}_{act}) \times \mathcal{U}_{act})^*)$ be a translucent event log, $\blacktriangleright \notin \pi_{act}(L)$, and $0 \leq f \leq 1$. The set of frequent translucent start arcs is defined as $E^{L,f}_{Start} = \{(\blacktriangleright, a) \in \{\blacktriangleright\} \times \pi_{act}(L) \mid Start_{freq}(a)^L > f \cdot \max_{c \in \pi_{act}(L)} Start_{freq}(c)^L\}$.*

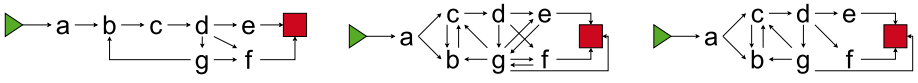
Finally, we define *frequent translucent end arcs*. Also, frequent translucent end arcs are defined similarly to the IMf (Leemans et al. 2013b).

Definition 17 (Frequent Translucent End Arcs). *Let $L \in \mathcal{B}((\mathcal{P}(\mathcal{U}_{act}) \times \mathcal{U}_{act})^*)$ be a translucent event log, $\blacksquare \notin \pi_{act}(L)$, and $0 \leq f \leq 1$. The set of frequent translucent end arcs is defined as $E^{L,f}_{End} = \{(a, \blacksquare) \in \pi_{act}(L) \times \{\blacksquare\} \mid End_{freq}(a)^L > f \cdot \max_{c \in \pi_{act}(L)} End_{freq}(c)^L\}$.*

By utilizing the sets of frequent arcs, we create a tfDFG.

Definition 18 (Translucent Frequent Directly-Follows Graph). *Let $L \in \mathcal{B}((\mathcal{P}(\mathcal{U}_{act}) \times \mathcal{U}_{act})^*)$ be a translucent event log, $\blacktriangleright, \blacksquare \notin \pi_{act}(L)$, and $0 \leq f \leq 1$. Furthermore, let $E^{L,f}_{df}$, $E^{L,f}_{par}$, $E^{L,f}_{Start}$, and $E^{L,f}_{End}$ be as defined before. A translucent frequent Directly-Follows Graph (tfDFG) of L and f is a directed graph $G = (V, E)$ where $V = \pi_{act}(L) \cup \{\blacktriangleright, \blacksquare\}$, where $\blacktriangleright$ is the start node, $\blacksquare$ is the end node, and $E \subseteq (E^{L,f}_{Start} \cup E^{L,f}_{End} \cup E^{L,f}_{df} \cup E^{L,f}_{par} \cup \{(\blacktriangleright, \blacksquare)\})$.*

To illustrate the difference between the classic DFG, the tDFG from Beyel and van der Aalst (2024a) and introduced earlier, and the newly introduced tfDFG, consider Fig. 9. While the DFG suffers from only sequential behavior, the tDFG shows a lot of parallel behavior. The tfDFG filters the infrequent relationships out due to the translucent exclusive-choice relationships and reflects the behavior better. The removal allows for the discovery of better process models.



(a) DFG (b) tDFG (c) tfDFG

Fig. 9 Various graphs based on $L = [\langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg} \rangle, \langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{ef}\,\underline{g} \rangle, \langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{ef}\,\underline{g}, \underline{bc}, \underline{c}, \underline{d}, \underline{efg} \rangle, \langle \underline{a}, \underline{bc}, \underline{c}, \underline{d}, \underline{ef}\,\underline{g}, \underline{ef} \rangle]$

Log splitting in filtered setting

Splits on the filtered DFG are only defined for traditional event logs. Thus, we have to extend the splitting definitions. Note that the concurrency split stays the same as in the translucent IM (Beyel and van der Aalst 2024a). Hence, we only redefine the exclusive-choice, sequence, and redo-loop split. These are similar to the splits in IMf (Leemans et al. 2013b).

To split a log according to an exclusive-choice filtered split, each trace is put into a sublog. All events that are not from the activity partition of a sublog are deviating. The split aims to maximize the number of events in a trace that fits the activity partition. Note that the enabled activities are projected on the partition as well.

Definition 19 (Exclusive-Choice Split Filtered). *Let* $L \in \mathcal{B}((\mathcal{P}(\mathcal{U}_{act}) \times \mathcal{U}_{act})^*)$ *be a translucent event log and* $(\times, A_1, \ldots, A_n)$ *be an exclusive-choice cut. Then, the exclusive-choice split filtered is for* $i \in \{1, \ldots, n\}$ *defined as* $L_i = \{\sigma \restriction_{A_i} \in L \mid i = \arg\max_{j \in \{1,\ldots,n\}} |\pi_{act}(\sigma) \restriction_{A_j}|\}$[1].

For example, the log $L = [\langle \underline{a}c, \underline{b}c\rangle, \langle \underline{c}, \underline{c}, \underline{c}\rangle, \langle \underline{a}c, \underline{b}c, \underline{c}\rangle]$ and the cut $(\times, \{a, b\}, \{c\})$ would result in $L_1 = [\langle \underline{a}, \underline{b}\rangle^2]$ and $L_2 = [\langle \underline{c}, \underline{c}, \underline{c}\rangle]$.

The sequence split filtered minimizes the number of events that are deviating. To do so, a function *findSplit* is introduced. The function takes the trace, the activity partition of interest, an index indicating the positing within the trace, and the previous activity partitions, which should be ignored, as input. The function returns the index to which a split point is identified. Each identified segment is then projected on the assigned activity partition.

Definition 20 (Sequence Split Filtered). *Let* $L \in \mathcal{B}((\mathcal{P}(\mathcal{U}_{act}) \times \mathcal{U}_{act})^*)$ *be a translucent event log and* $(\rightarrow, A_1, \ldots, A_n)$ *be a sequence cut. We define* $findSplit(\sigma, A, s, I) = \min \arg\min_{1 \leq j \leq |\sigma|} \mid \{e \in \langle \sigma_s, \ldots, \sigma_j\rangle \mid \pi_{act}(e) \notin I\} \mid - |\{e \in \langle \sigma_s, \ldots, \sigma_j\rangle \mid \pi_{act}(e) \in A\}|$. *We define* $s_1 = 0$. *Also, we define* $e_i = findSplit(\sigma, A_i, s_i, I)$. *Moreover, we define for* $i \in \{2, \ldots, n\}$:

$$s_i = \begin{cases} e_{i-1}, & \text{if } s_{i-1} = e_{i-1} \\ e_{i-1} + 1, & \text{if } s_{i-1} \neq e_{i-1} \end{cases}$$

Furthermore, we define for each $\sigma \in L, s, e \in \{0, \ldots, |\sigma|\}, s \leq e$:

$$getSubSequence(\sigma, s, e) = \begin{cases} \langle\rangle, & \text{if } s = e \\ \langle \sigma_s, \ldots, \sigma_e\rangle, & \text{if } s \neq e \end{cases}$$

Then, the sequence split filtered is for $i \in \{1, \ldots, n\}$ *defined as* $L_i = \{getSubSequence(\sigma, s_i, e_i) \restriction_{A_i} \mid \sigma \in L\}$.

Given the log $L = [\langle \underline{a}c, \underline{b}c, \underline{c}\rangle, \langle \underline{b}c, \underline{a}c, \underline{c}\rangle, \langle \underline{c}, \underline{a}c, \underline{b}c, \underline{c}\rangle]$ and the cut $(\rightarrow, \{a, b\}, \{c\})$, the result is $L_1 = [\langle \underline{a}, \underline{b}\rangle^2, \langle \underline{b}, \underline{a}\rangle]$ and $L_2 = [\langle c\rangle^3]$. The first c of the third trace of L is removed. Also, again, a projection of enabled activities takes place.

[1] arg max returns one $j \in \{1, \ldots, n\}$ even when when the are multiple values that maximize $|\pi_{act}(\sigma) \restriction_{A_j}|$

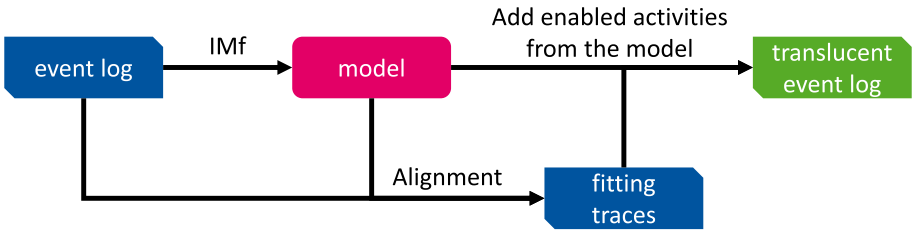


Fig. 10 Overview of creating translucent event logs using the IMf (Leemans et al. 2013b)

The redo-loop split filtered detects deviations when a trace ends with an activity not part of the first partition. If this occurs, an empty trace is added to the log of the first partition.

Definition 21 (Redo-Loop Split Filtered). *Let* $L \in \mathcal{B}((\mathcal{P}(\mathcal{U}_{act}) \times \mathcal{U}_{act})^*)$ *be a translucent event log and* $(\circlearrowleft, A_1, \ldots, A_n)$ *be a redo-loop cut. For* $\sigma \in L$ *we define*

$$first(\sigma) = \begin{cases} \bot, & \text{if } \sigma = \langle\rangle \\ \pi_{act}(\sigma_1), & \text{if } \sigma \neq \langle\rangle \end{cases}$$

Moreover, for $\sigma \in L$ *we define*

$$last(\sigma) = \begin{cases} \bot, & \text{if } \sigma = \langle\rangle \\ \pi_{act}(\sigma_{|\sigma|}), & \text{if } \sigma \neq \langle\rangle \end{cases}$$

The redo-loop split is for $i \in \{1, \ldots, n\}$ *defined as* $L_i = \{\sigma \restriction_{A_i} \mid \sigma' \cdot \sigma \cdot \sigma'' \in L \wedge \forall_{e \in \sigma} \pi_{act}(e) \in A_i \wedge last(\sigma') \notin A_i \wedge first(\sigma'') \notin A_i\}$. *Moreover, for all* $\sigma \in L$, *if* $first(\sigma) \notin A_1$, *then* $L_1 = L_1 \uplus [\langle\rangle]$. *Furthermore, for all* $\sigma \in L$, *if* $last(\sigma) \notin A_1$, *then* $L_1 = L_1 \uplus [\langle\rangle]$. *Also, if* $|L_1| - |\bigcup_{j \in \{2,\ldots,n\}} L_j| \neq 1$, $L_1 = L_1 \uplus [\langle\rangle^{|\bigcup_{j \in \{2,\ldots,n\}} L_j| - |L_1| + 1}]$.

Given a log $L = [\langle \underline{a}b, \underline{b}\rangle]$ and cut $(\times, \{a\}, \{b\})$, the result are sub-logs $L_1 = [\langle a\rangle, \langle\rangle]$ and $L_2 = [\langle b\rangle]$.

As with the IMf (Leemans et al. 2013b), the empty traces fall-through is applied if enough empty traces occur in an event log. If there are not enough empty traces to trigger the fall-through, they are removed from the event log. Consequently, empty traces are never projected on an activity partition.

Evaluation

In this section, we evaluate our proposed method by conducting various experiments. First, we explain our experiment design, including the generation process of a translucent event log. Second, we focus on the results of the experiments related to the sepsis event log (Mannhardt 2016). Third, we focus on the results of the experiments related to the road traffic fine management event log (de Leoni and Mannhardt 2015). We implemented our approaches in Python using PM4Py (Berti et al. 2023)[2].

[2] Our code is provided here: https://github.com/hherbertb/TranslucentActivityRelationships

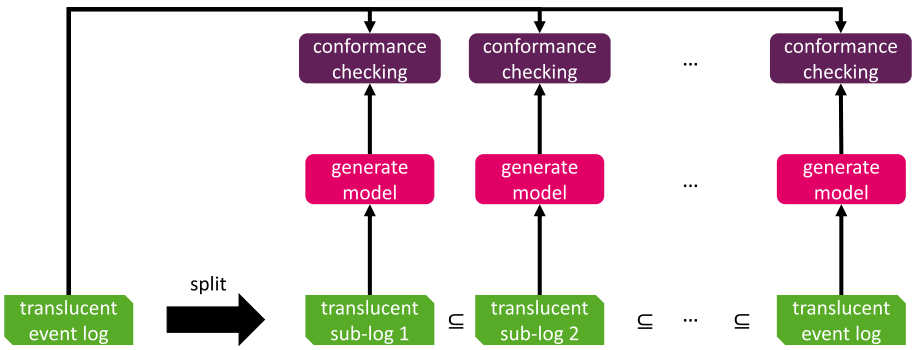


Fig. 11 Overview of our evaluation concept

Experimental setup

In this section, we explain our experimental setup. First, we focus on the generation of translucent event logs. Second, we explain how we measure whether our discovery techniques improve existing techniques.

As discussed in Beyel and van der Aalst (2022) and Beyel et al. (2024), enabled activities can be extracted from screenshots using computer-vision techniques. However, we generate artificial translucent event logs to assess various scenarios and improve data availability and reproducibility. This method follows a similar approach to that presented in Beyel and van der Aalst (2022). An overview of the process is shown in Fig. 10. As shown, a process model is discovered from an existing event log. The model and the log are aligned, and by considering only fitting traces, we incorporate information on enabled activities using the replay state in the model. For further details, we refer to Beyel and van der Aalst (2022). For this, we use Petri nets discovered by the IMf (Leemans et al. 2013b) with different noise thresholds: 40 %, 60 %, and 80 %. We opted not to use the standard IM (Leemans et al. 2013a), as this algorithm generates models that result in translucent event logs with an excessive number of enabled activities per event. The chosen noise thresholds balance the preprocessing time required to create synthetic translucent event logs. A lower threshold would have made the process too time-consuming without providing additional valuable insights. The output of the IMf is a block-structured process model. Consequently, the resulting logs will capture this bias. When discovering process models from the created translucent event logs, the IMf and its translucent variations (IMfto, IMftf, and IMfts) use a threshold of 20 %, 40 %, 60 %, and 80 %. We also investigated whether there are differences if a translucent DFG or classical DFG is used in fall-throughs, but we did not observe any differences within our experiments.

To assess whether translucent activity relationships add value to process discovery, we compare the standard IM and IMf, which do not account for enabled activities, the translucent IM (Beyel and van der Aalst 2024a), i.e., IMto, IMtf, and IMts, and the translucent IMf, i.e., IMfto, IMftf, and IMfts, presented in this work. Additionally, we evaluate whether approaches utilizing information on enabled activities discover similar or improved models with fewer variants compared to the original algorithms. To explore these aspects, we design evaluation scenarios, as outlined in Fig. 11. We split

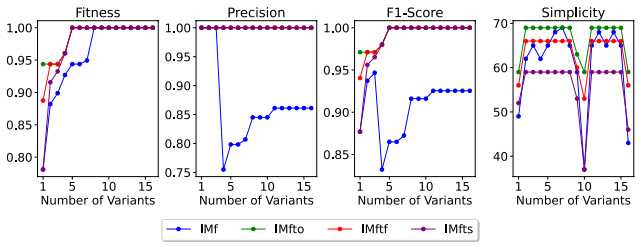
a translucent event log into several sublogs, by focusing on variants of the translucent event log. The first sublog includes all traces from the most frequent variant, the second sublog adds traces from the second most frequent variant, and so on. For each sublog, the IM (Leemans et al. 2013a), IMf (Leemans et al. 2013b), the translucent IM (Beyel and van der Aalst 2024a) (IMto, IMtf, and IMts), and our new approach (IMfto, IMftf, and IMfts) discover process models. To determine whether a more representative process model is discovered with less information, we compute fitness, precision, and F1 scores using the unsplit translucent event log. Fitness scores are calculated using alignments with a standard cost function (Adriansyah 2014). For precision scores, we employ a modified version of the escaping arcs approach (Munoz-Gama and Carmona 2010), which considers both the executed activities and the enabled activities in the log as behavior. Only fitting traces are analyzed. See Beyel and van der Aalst (2024d) for more information. The F1 score is calculated as the harmonic mean of fitness and precision. We also measure simplicity by counting the number of places, transitions, and arcs in each discovered Petri net.

Sepsis

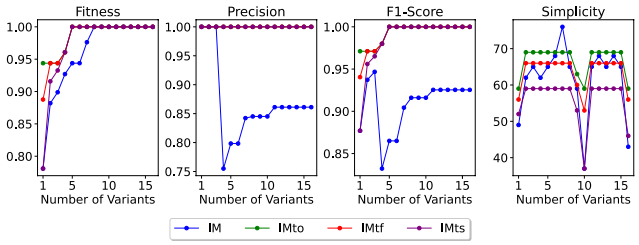
Our modified sepsis event log (Mannhardt 2016) can be found in Beyel and van der Aalst (2024c). The thresholds 60 % and 80 % for the IMf (Leemans et al. 2013b) to discover the model used as input for adding enabled activities resulted in no difference in the performance of the various algorithms. Also, the threshold for discovering process models on the generated logs (see Definitions 14, 15, 16, 17 and 18) had a minimal impact. There is only a difference in performance when comparing the classical IM setting to other values. The results are depicted in Fig. 12. The results indicate that the translucent variants achieve perfect fitness with fewer variants while still providing perfect precision. This highlights that our methods discover process models that can be used to tackle the generalization issue since fewer variants are needed to discover a well-representative process model.

When looking at the translucent discovery algorithms, we can denote that all simultaneously achieve a perfect fitness score. However, the models discovered by IMto and IMfto fit better when fewer variants are considered. Considering simplicity, the translucent variations are more stable and behave similarly for the different variant counts. The final models of the non-translucent discovery algorithms are slightly simpler.

When using a threshold of 40 % for the IMf (Leemans et al. 2013b) for discovering the model that is used to annotate the classic event logs, we observe similar results. The results are depicted in Fig. 13. As we denote, the discovery algorithms using information on enabled activities perform better. We can observe that if the threshold for discovering models (described in Definitions 14, 15, 16, 17 and 18) is 40 % or 20 %, perfect fitness is achieved by including more variants than with a higher threshold. As before, the precision of the models discovered by translucent discovery algorithms is better than those created by discovery algorithms that do not use the information on enabled activities. Simplicity behaves similarly to the previous experiment.



(a) IMf (Leemans et al. 2013b) and its translucent variants IMfto, IMftf, and IMfts presented in this work. There are no differences between the thresholds 20 %, 40 %, 60 %, and 80 % for filtering the graph (see Definitions 14 to 18)



(b) IM (Leemans et al. 2013a) and its translucent variants, IMto, IMtf, and IMts presented in (Beyel and van der Aalst 2024)

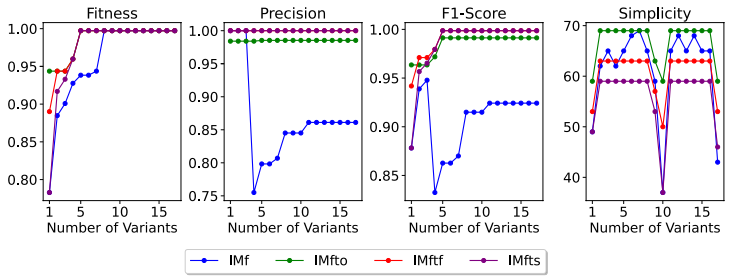
Fig. 12 Results for the algorithms on the modified sepsis event log (Beyel and van der Aalst 2024c). The models used for annotating the event logs were discovered by the IMf (Leemans et al. 2013b) with a noise threshold of 60 % and 80 %

Road traffic fine management

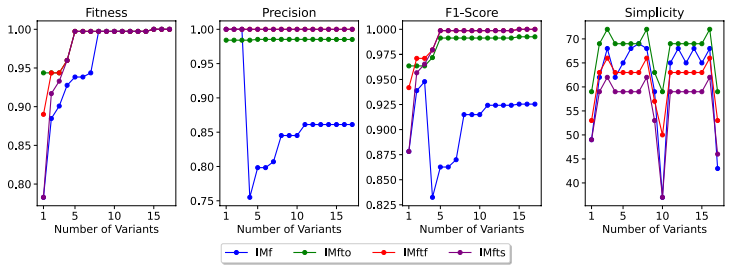
Our modified road traffic fine management event log (de Leoni and Mannhardt 2015) can be found in Beyel and van der Aalst (2024b). When the threshold of the IMf (Leemans et al. 2013b) for discovering the ground truth model is set to 80 % or 60 %, only two variants are returned. Since such a low number gives limited insights, we do not focus on the results of these experiments.

When considering the translucent event log created with a model discovered by the IMf (Leemans et al. 2013b) with a threshold of 40 %, we can observe more differences between the different thresholds for the IMf and its translucent variants, IMfto, IMftf, and IMfts (see Fig. 14). By decreasing the threshold for discovering models from 80 % to 60 % (described in Definitions 14, 15, 16, 17 and 18), we allow for discovering models with perfect fitness for a larger number of variants. By decreasing this threshold to 40 %, we observe that the performance of IMfto and IMftf improves. By further decreasing this threshold to 20 %, we observe that the IMfts is able to discover perfectly fitting models with a larger number of variants. We can also denote that the IMf (Leemans et al. 2013b) performance does not change for the different threshold values. The models discovered by utilizing information on enabled activities are simpler.

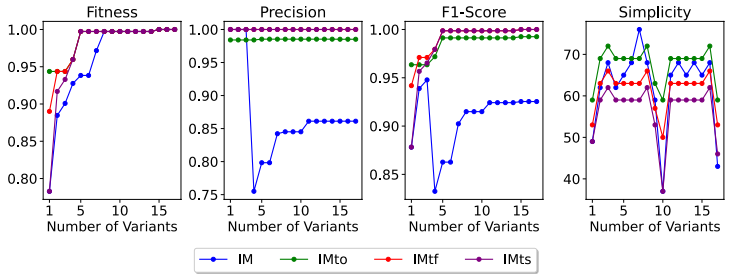
The results for the IM (Leemans et al. 2013a), IMto, IMtf, and IMts (Beyel and van der Aalst 2024a) are showcased in Fig. 15. The performance of the translucent variants is getting closer to each other. Concerning simplicity, the translucent variants of the IM usually find again simpler models. Overall, it seems that IMtf and IMftf are providing the best results.



(a) IMf (Leemans et al. 2013b) and its translucent variants IMfto, IMftf, and IMfts presented in this work for a threshold of 60 % and 80 % when discovering models (see Definitions 14 to 18)



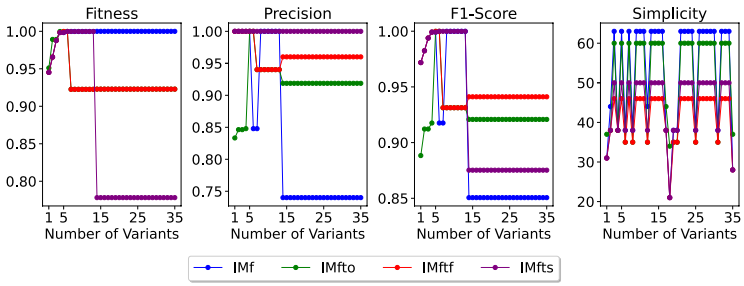
(b) IMf (Leemans et al. 2013b) and its translucent variants IMfto, IMftf, and IMfts presented in this work for a threshold of 20 % and 40 % when discovering models (see Definitions 14 to 18)



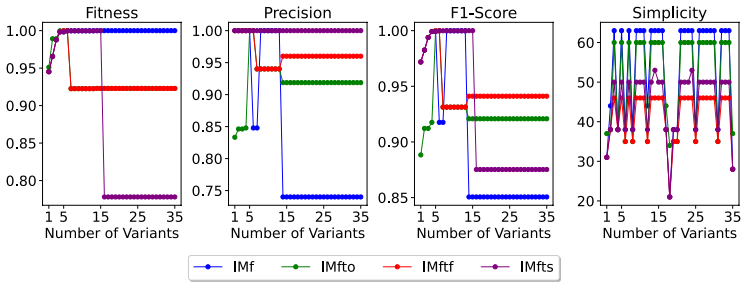
(c) IM (Leemans et al. 2013a) and its translucent variants IMto, IMtf, and IMts presented in (Beyel and van der Aalst 2024)

Fig. 13 Results for the algorithms on the modified sepsis event log (Beyel and van der Aalst 2024c). The model used for annotating the event log was discovered by the IMf (Leemans et al. 2013b) with a noise threshold of 40 %

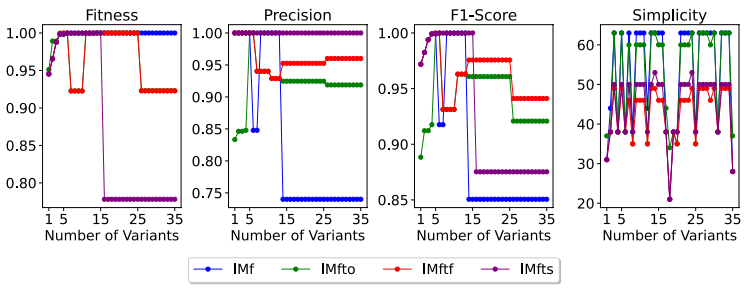
An overview of different process models discovered by the approaches presented in this work is given in Fig. 16. The model discovered by the IMf (see Fig. 16a) allows for a lot of parallel behavior, thus lowering precision. The model created by the IMfto (see Fig. 16b) allows for less parallel behavior but still has a parallel block in a branch. The model generated by the IMftf (see Fig. 16c) replaces the parallel block with a decision. The model discovered by the IMfts (see Fig. 16d) provides no parallelism or choice in the branch in question. Hence, the model has the highest precision score. The models support the scores described in Fig. 14d. Also, the models show the value of enabled activities in process discovery.



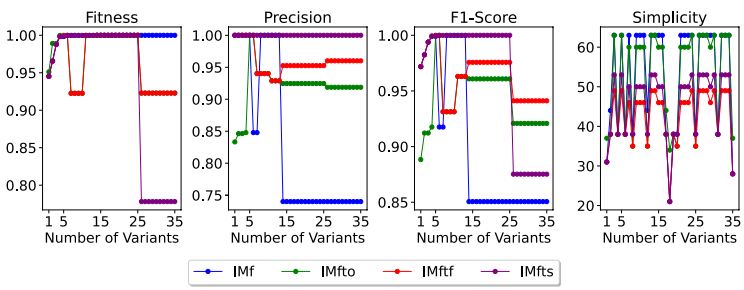
(a) IMf (Leemans et al. 2013b) and its translucent variants with a threshold of 80 %



(b) IMf (Leemans et al. 2013b) and its translucent variants with a threshold of 60 %



(c) IMf (Leemans et al. 2013b) and its translucent variants with a threshold of 40 %



(d) IMf (Leemans et al. 2013b) and its translucent variants with a threshold of 20 %

Fig. 14 Results for the IMf (Leemans et al. 2013a) and IMfto, IMftf, and IMfts presented in this work on the modified road traffic fine management event log (Beyel and van der Aalst 2024b). The model used for annotating the event log was discovered by the IMf (Leemans et al. 2013b) with a noise threshold of 40 %

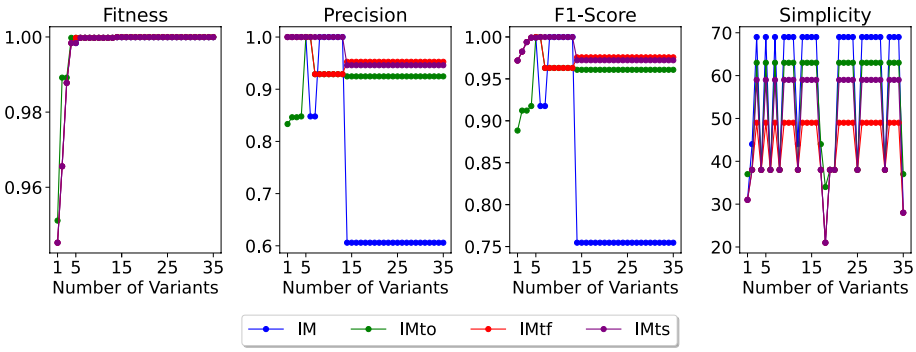
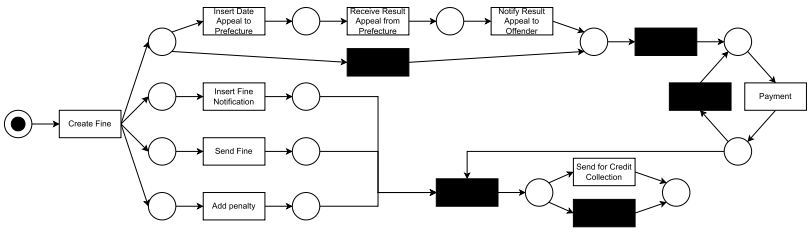
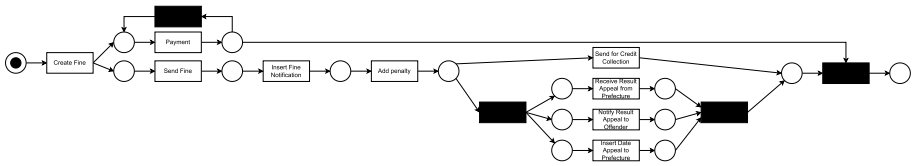


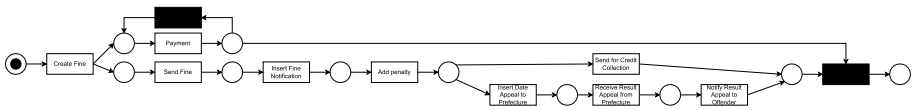
Fig. 15 IM (Leemans et al. 2013b) and its translucent variants (IMto, IMtf, and IMts (Beyel and van der Aalst 2024a)) on the modified road traffic fine management event log (Beyel and van der Aalst 2024b). The model used for annotating the event log was discovered by the IMf (Leemans et al. 2013b) with a noise threshold of 40 %



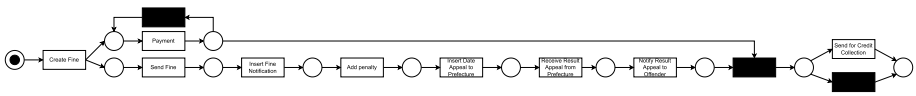
(a) Discovered by the IMf (Leemans et al. 2013b) with a threshold of 20 %



(b) Discovered by the IMfto with a threshold of 20 %



(c) Discovered by the IMftf with a threshold of 20 %



(d) Discovered by the IMfts with a threshold of 20 %

Fig. 16 Models discovered by IMf and its translucent variants (IMfto, IMftf, and IMfts) on the complete modified road traffic fine management event log (Beyel and van der Aalst 2024b). The model used for annotating the event log was discovered by the IMf (Leemans et al. 2013b) with a noise threshold of 40 %

Discussion

In general, working with translucent event data tackles two challenges described in van der Aalst et al. (2011): dealing with complex event logs with diverse characteristics and balancing quality criteria. Especially the generalization issue (Carmona et al. 2018) can be tackled by utilizing these data.

The generalization issue is based on the problem that only a sample from an information system is used to which process mining is applied. Thus, verifying that the model represents all the data is impossible. Using translucent activity relationships, we can show that incorporating them can help to produce more general process models.

In the following, we discuss our work's advantages and disadvantages and our methods' generalization capabilities.

Advantages and disadvantages

We first focus on the advantages. First, our technique allows for discovering more robust process models with fewer cases using translucent activity relationships. This is supported by the higher F1 scores and simplicity scores in the experiments and our drawn conclusion for generalization. Second, within our techniques, we allow for fallbacks to establish work, thus allowing for at least the performance that the IMf (Leemans et al. 2013b) allows for. Third, by including frequencies, we can utilize the translucent exclusive-choice relationship to filter out noise within the data. This is was not possible before.

Next, we focus on the disadvantages. First, we have to have information on the activities that are enabled. As discussed, there are currently two ways of receiving this information: using domain knowledge or extracting this information from external sources such as screenshots (Beyel and van der Aalst 2022; Beyel et al. 2024). However, such logs are not yet available on a larger scale. Second, if we consider using screenshots as a source for a translucent event log in a multi-user environment, issues can happen. If we consider a task management tool, e.g., Jira, all users have to be recorded. If at least one user is not recorded and performs tasks, these tasks can disappear without their recording. Consequently, false conclusions about relationships are created. Third, translucent directly-follows relationships are counted twice since each translucent parallel relationship is also a directly-follows relationship. Consider the event log $L = [\langle \underline{a}b, \underline{b}\rangle^{10}, \langle \underline{a}, \underline{b}c\rangle^{10}, \langle \underline{a}b, \underline{c}\rangle^{15}]$. $df^L_{freq}(a,b) = 20$, $par^L_{freq\_synch}(a,b) = 10$, and $exc^L_{freq\_synch}(a,b) = 15$. As a result, an arc from a to b would be created, although one may argue that the counts are not exclusive to a relationship.

Generalization

Within our experiments, we did not discuss in detail the generalization capabilities of our work. Generalization aims to measure how well a process model can generalize from the behavior recorded in event logs. It is important to note that translucent event logs can show possible behavior besides the executed behavior. As such, techniques that use this information can discover more generalized process models. In the context of our experiments, generalization can be seen when considering our fitness measurements. We discover process models with sublogs and compute the fitness considering the complete translucent event logs. Therefore, we can access desired generalization capabilities (Carmona et al. 2018).

As our experiments have revealed, the classic IM without frequencies (Leemans et al. 2013a) sometimes needed more variants to achieve the same fitness score as its translucent variation. When adding more variants in the discovery process, the precision values get worse for the classic IM. Consequently, the translucent variants of the IM (Beyel and van der Aalst 2024a) can discover more general process models than the classic IM.

Considering frequencies leads to mixture results. In the context of the sepsis event log and its translucent variants (Mannhardt 2016; Beyel and van der Aalst 2024c), we make similar observations in comparison to when no frequencies are considered. One reason is that the sepsis event log is a rather complex event log in which frequencies have a limited impact. Concerning the road traffic fine management event log and its translucent variants (de Leoni and Mannhardt 2015; Beyel and van der Aalst 2024b), we can denote that the IMf (Leemans et al. 2013b) has better fitness values than IMfto, IMftf, and IMfts. Nonetheless, the translucent precision is not as high as that of the translucent variants. The decrease in precision of the IMf is greater than the decrease in fitness of its translucent variants. When considering the discovered models in Fig. 16, it becomes evident that higher fitness is achieved by the parallel block after creating the fine.

Overall, our approaches can discover process models that generalize well. While in some cases, the non-translucent IM and IMf can discover more general models, this may not be the desired generalization.

Conclusion

In this work, we showed how to utilize all three translucent activity relationships within process discovery. We showed how the IMf (Leemans et al. 2013b) can be extended with the partially newly introduced relationships and their frequencies. As showcased in the evaluation, our methods are able to discover more generalizable process models. This is highly interesting when assuming that we cannot use all available data to discover process models. From our proposed methods, it seems promising first to use the information on enabled activities and, if that does not result in a finding, to use the classic setting.

There are also points for future work. As we observed in our evaluation, utilizing information on enabled activities is valuable for process discovery. Thus, extending already existing process-discovery techniques is beneficial. Example techniques involve the ILP Miner (van der Werf et al. 2009), AGNEs Miner (Goedertier et al. 2009), and Staged Process Miner (Nguyen et al. 2017). In addition to the three defined relationships, it is possible to design additional ones, such as those that capture causal relationships. Furthermore, it is feasible to create relationships that address specific challenges in process discovery, such as handling long-term dependencies. Moreover, analyzing activities that are enabled but never executed presents an interesting opportunity, as a log represents only a sample of real-world behavior. As mentioned at the beginning of the work, the GUI example (see Fig. 1) assumes that not multiple users are involved in a task. However, in modern task management software, multiple users can do one task. Thus, our assumption about parallelism and choice may lead to false conclusions in such a scenario. As a result, being aware of this assumption is necessary. When considering the option to receive translucent event logs from user

interactions, the area of Robotic Process Automation (RPA) (Syed et al. 2020) is interesting. Fewer variants of executions might be needed to create a robust software bot that executes tasks from the recorded data using the methods presented in this work. An overview of Robotic Process Mining (RPM) and task mining is given in Leno et al. (2021) and Dumas et al. (2022).

In general, information about enabled activities needs to be incorporated into process discovery, whether by enhancing existing techniques, defining additional relationships, or developing new discovery methods.

Authors' contributions
H.B. was responsible for conceptualization, data curation, methodology, software development, validation, visualization, and writing of the manuscript. W.v.d.A. performed the funding acquisition, provided supervision, and reviewed the manuscript.

Funding
We thank the Alexander von Humboldt (AvH) Stiftung for supporting our research (grant no. 1191945).

Data availability
The data sources are provided within the manuscript. In particular, they can be found here: https://doi.org/10.5281/zenodo.14161548 and https://doi.org/10.5281/zenodo.14161124.

Declarations

Ethics approval and consent to participate
Not applicable.

Competing interests
The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Received: 14 November 2024 Accepted: 2 April 2025
Published online: 09 July 2025

References

Adriansyah A (2014) Aligning observed and modeled behavior. PhD thesis, Mathematics and Computer Science. https://doi.org/10.6100/IR770080
Augusto A, Carmona J, Verbeek E (2022) Advanced process discovery techniques. In: Process Mining Handbook. Springer, pp 76–107. https://doi.org/10.1007/978-3-031-08848-3_3
Augusto A, Conforti R, Dumas M et al (2019a) Automated discovery of process models from event logs: Review and benchmark. IEEE Trans Knowl Data Eng 31(4):686–705. https://doi.org/10.1109/TKDE.2018.2841877
Augusto A, Conforti R, Dumas M et al (2019b) Split miner: automated discovery of accurate and simple business process models from event logs. Knowl Inf Syst 59(2):251–284. https://doi.org/10.1007/s10115-018-1214-x
Augusto A, Dumas M, Rosa ML (2020) Automated discovery of process models with true concurrency and inclusive choices. In: Process Mining Workshops - ICPM. Springer, pp 43–56. https://doi.org/10.1007/978-3-030-72693-5_4
Berti A, van Zelst S, Schuster D (2023) PM4Py: A process mining library for python. Softw Impacts 17:100556. https://doi.org/10.1016/j.simpa.2023.100556
Beyel HH, Manuel S, van der Aalst WMP (2024) ActivityGen: Extracting enabled activities from screenshots. In: ECAI. IOS Press. https://doi.org/10.3233/FAIA240553
Beyel HH, van der Aalst WMP (2022) Creating translucent event logs to improve process discovery. In: Process Mining Workshops - ICPM. Springer, pp 435–447. https://doi.org/10.1007/978-3-031-27815-0_32
Beyel HH, van der Aalst WMP (2024a) Improving process discovery using translucent activity relationships. In: BPM. Springer, pp 146–163. https://doi.org/10.1007/978-3-031-70396-6_9
Beyel HH, van der Aalst WMP (2024b) Translucent Event Logs based on the Road Traffic Fine Management Event Log and the Inductive Miner - infrequent. https://doi.org/10.5281/zenodo.14161124
Beyel HH, van der Aalst WMP (2024c) Translucent Event Logs based on the Sepsis Event Log and the Inductive Miner - infrequent. https://doi.org/10.5281/zenodo.14161548
Beyel HH, van der Aalst WMP (2024d) Translucent precision: Exploiting enabling information to evaluate the quality of process models. In: RCIS. Springer, pp 29–37. https://doi.org/10.1007/978-3-031-59468-7_4
Brunelli R (2009) Template Matching Techniques in Computer Vision: Theory and Practice. Wiley. https://doi.org/10.1002/9780470744055
Carmona J, van Dongen BF, Solti A et al (2018) Conformance Checking - Relating Processes and Models. Springer. https://doi.org/10.1007/978-3-319-99414-7

Cortadella J, Kishinevsky M, Lavagno L et al (1998) Deriving petri nets for finite transition systems. IEEE Trans Comput 47(8):859–882. https://doi.org/10.1109/12.707587
de Leoni M (2022) Foundations of process enhancement. In: Process Mining Handbook. Springer, pp 243–273. https://doi.org/10.1007/978-3-031-08848-3_8
de Leoni MM, Mannhardt F (2015) Road traffic fine management process. https://doi.org/10.4121/UUID:270FD440-1057-4FB9-89A9-B699B47990F5
Dumas M, Rosa ML, Leno V et al (2022) Robotic process mining. In: Process Mining Handbook. Springer, pp 468–491. https://doi.org/10.1007/978-3-031-08848-3_16
Dumas M, Rosa ML, Mendling J et al (2018) Fundamentals of Business Process Management, 2nd edn. Springer. https://doi.org/10.1007/978-3-662-56509-4
Ehrenfeucht A, Rozenberg G (1990a) Partial (set) 2-structures. part I: basic notions and the representation problem. Acta Informatica 27(4):315–342. https://doi.org/10.1007/BF00264611
Ehrenfeucht A, Rozenberg G (1990b) Partial (set) 2-structures. part II: state spaces of concurrent systems. Acta Informatica 27(4):343–368. https://doi.org/10.1007/BF00264612
Goedertier S, Martens D, Vanthienen J et al (2009) Robust process discovery with artificial negative events. J Mach Learn Res 10:1305–1340. https://doi.org/10.5555/1577069.1577113
Leemans SJJ, Fahland D, van der Aalst WMP (2013a) Discovering block-structured process models from event logs - A constructive approach. In: PETRI NETS. Springer, pp 311–329. https://doi.org/10.1007/978-3-642-38697-8_17
Leemans SJJ, Fahland D, van der Aalst WMP (2013b) Discovering block-structured process models from event logs containing infrequent behaviour. In: Business Process Management Workshops. Springer, pp 66–78. https://doi.org/10.1007/978-3-319-06257-0_6
Leno V, Polyvyanyy A, Dumas M et al (2021) Robotic process mining: Vision and challenges. Bus Inf Syst Eng 63(3):301–314. https://doi.org/10.1007/s12599-020-00641-4
Mannhardt F (2016) Sepsis cases - event log. https://doi.org/10.4121/uuid:915d2bfb-7e84-49ad-a286-dc35f063a460
Mauser S, Lorenz R (2009) Variants of the language based synthesis problem for petri nets. In: Ninth International Conference on Application of Concurrency to System Design, ACSD. IEEE Computer Society, pp 89–98. https://doi.org/10.1109/ACSD.2009.19
Munoz-Gama J, Carmona J (2010) A fresh look at precision in process conformance. In: BPM. Springer. https://doi.org/10.1007/978-3-642-15618-2_16
Nguyen H, Dumas M, ter Hofstede AHM et al (2017) Mining business process stages from event logs. In: CAiSE. Springer, pp 577–594. https://doi.org/10.1007/978-3-319-59536-8_36
Reisig W (1985) Petri Nets: An Introduction, EATCS Monographs on Theoretical Computer Science, vol 4. Springer, Heidelberg. https://doi.org/10.1007/978-3-642-69968-9
Solé M, Carmona J (2011) Light region-based techniques for process discovery. Fundam Informaticae 113(3–4):343–376. https://doi.org/10.3233/FI-2011-612
Solé M, Carmona J (2012) Incremental process discovery. Trans Petri Nets Other Model Concurr 5:221–242. https://doi.org/10.1007/978-3-642-29072-5_10
Solé M, Carmona J (2013) Region-based foldings in process discovery. IEEE Trans Knowl Data Eng 25(1):192–205. https://doi.org/10.1109/TKDE.2011.192
Syed R, Suriadi S, Adams M et al (2020) Robotic process automation: Contemporary themes and challenges. Comput Ind 115:103162. https://doi.org/10.1016/j.compind.2019.103162
van der Aalst WMP (2016) Process Mining - Data Science in Action, 2nd edn. Springer. https://doi.org/10.1007/978-3-662-49851-4
van der Aalst WMP (2019) Lucent process models and translucent event logs. Fundam Informaticae 169(1–2):151–177. https://doi.org/10.3233/FI-2019-1842
van der Aalst WMP (2022) Foundations of process discovery. In: Process Mining Handbook. Springer, pp 37–75. https://doi.org/10.1007/978-3-031-08848-3_2
van der Aalst WMP, Adriansyah A, de Medeiros AKA et al (2011) Process mining manifesto. In: BPM Workshops. Springer, pp 169–194. https://doi.org/10.1007/978-3-642-28108-2_19
van der Aalst WMP, Rubin VA, Verbeek HMW et al (2010) Process mining: a two-step approach to balance between underfitting and overfitting. Softw Syst Model 9(1):87–111. https://doi.org/10.1007/s10270-008-0106-z
van der Werf JMEM, van Dongen BF, Hurkens CAJ et al (2009) Process discovery using integer linear programming. Fundam Informaticae 94(3–4):387–412. https://doi.org/10.3233/FI-2009-136
van Detten JN, Schumacher P, Leemans SJJ (2023) An approximate inductive miner. In: ICPM. IEEE, pp 129–136. https://doi.org/10.1109/ICPM60904.2023.10271971

Publisher's Note