Discovering Process Models with Long-Term Dependencies while Providing Guarantees and Filtering Infrequent Behavior Patterns

Lisa L. Mannel*, **Wil M. P. van der Aalst**

Process and Data Science (PADS)

RWTH Aachen University

Aachen, Germany

mannel@pads.rwth-aachen.de, wvdaalst@pads.rwth-aachen.de

Abstract. In process discovery, the goal is to find, for a given event log, the model describing the underlying process. While process models can be represented in a variety of ways, Petri nets form a theoretically well-explored description language and are therefore often used. In this paper, we extend the eST-Miner process discovery algorithm. The eST-Miner computes a set of Petri net places which are considered to be fitting with respect to a certain fraction of the behavior described by the given event log as indicated by a given noise threshold. It evaluates all possible candidate places using token-based replay. The set of replayable traces is determined for each place in isolation, i.e., these sets do not need to be consistent. This allows the algorithm to abstract from infrequent behavioral patterns occurring only in some traces. However, when combining places into a Petri net by connecting them to the corresponding uniquely labeled transitions, the resulting net can replay exactly those traces from the event log that are allowed by the combination of all inserted places. Thus, inserting places one-by-one without considering their combined effect may result in deadlocks and low fitness of the Petri net. In this paper, we explore adaptions of the eST-Miner, that aim to select a subset of places such that the resulting Petri net guarantees a definable minimal fitness while maintaining high precision with respect to the input event log. Furthermore, current place evaluation techniques tend to block the execution of infrequent activity labels. Thus, a refined place fitness metric is introduced and thoroughly investigated. In our experiments we use real and artificial event logs to evaluate and compare the impact of the various place selection strategies and place fitness evaluation metrics on the returned Petri net.

Keywords: Process Discovery, Petri Nets, eST-Miner, long-term dependencies, non-free choice

*Address for correspondence: Process and Data Science (PADS), RWTH Aachen University, Aachen, Germany

1. Introduction and related work

More and more corporations and organizations support their processes using information systems, which record the occurring behavior and represent this data in the form of *event logs*. Each event in such a log has a name identifying the executed activity (activity name), an identifier mapping the event to some execution instance (case id), a timestamp showing when the event was observed, and often extended meta-data of the activity or process instance. Based on the timestamp and case id, we can organize the event log as sequences of activities, also called *traces*, which represent example execution sequences of the process. In the field of *process discovery*, we utilize the event log to identify relations between the activities (e.g., pre-conditions, choices, concurrency), which are then expressed within a process model. While several modeling formalisms exist, this work focuses on modeling processes using Petri nets [1, 2, 3, 4].

Process discovery is non-trivial for various reasons. We cannot assume that the given event log is complete, as some possible behavior might be yet unobserved. Also, real-life event logs often contain noise in the form of incorrectly recorded data or deviant behavior, which is usually not desired to be reflected in the process model. However, correctly classifying behavior as noise is often not trivial. Several quality dimensions have been defined to evaluate process models ([5]) which are partially in conflict with each other and thus it is usually not possible to achieve perfect values for all aspects. A process model with good *fitness* can (mostly) reproduce the behavior contained in the event log, while high *precision* corresponds to not allowing for (much) unobserved behavior. Furthermore, a model with good *generalization* is expected to express behavior possible in the process that generated the event log even if it has not yet been observed. Finally, since process models are often used in contexts that require interpretation by a person, model *simplicity* is of interest. Additionally, an ideal discovery algorithm should be reasonably time and space efficient. Since real-life event logs rarely allow for a model that perfectly satisfies all the different quality criteria, different discovery algorithms focus on different aspects, while neglecting others. As a result, the models returned by these discovery algorithms for a given event log can differ significantly. Note that generally there is no clearly definable best process model but instead this choice depends on what purpose it should serve.

Many existing discovery algorithms abstract from the full information given in a log or generate places heuristically, in order to decrease computation time and complexity of the returned process models. While this is convenient in many settings, the resulting models are often underfitting, in particular when processes are complex. Examples are the Alpha Miner variants [6], the Inductive Mining family [7], genetic algorithms or Heuristic Miner [8]. In contrast to these approaches, which are not able to (reliably) discover complex model structures, algorithms based on region theory [9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19]) discover models whose behavior is the minimal behavior representing the input event log. On the downside, these approaches are known to be rather time-consuming, cannot handle noise, and tend to produce complex, overfitting models which can be hard to interpret. A combination of strategies has been introduced in [20], which aims to circumvent performance issues by limiting the application of region theory to small fragments of a pre-discovered Petri net.

In [21] we introduced the discovery algorithm eST-Miner. This approach aims to combine the capability of *finding complex control-flow structures like long-term dependencies* (non-free choice con-

structs) with an inherent ability to *filter infrequent behavior patterns* while exploiting the token-game to *increase efficiency*. The basic idea is to construct a Petri net without any places and one uniquely labeled transition for each activity in the event log, and then compute and insert a set of fitting places. A place can be considered fitting even if it disagrees with a part of behavior in the event log, which allows to filter infrequent behavior. Efficiency is significantly increased by skipping uninteresting parts of the search space. The approach guarantees that all places considered fitting with respect to the event log are discovered, and can thereby provide some kind of precision guarantee. In fact, in the case where fitting places are defined as feasible places, i.e., the discovered Petri net is required to reflect all behavior in the input event log, it guarantees to return a minimal over-approximation.

The eST-Miner evaluates each candidate place in isolation, i.e., for each place it focuses only on the activities in the event log whose transitions are connected to that place. This allows us filter infrequent behavior patterns in the event log, by requiring each place to be able to replay only parts of the traces in the event log. A candidate place will be accepted and inserted into the returned Petri net, if the event log contains sufficient support for the relation between the activities as defined by the place. Common noise filtering techniques, which are often applied as a preprocessing step before applying a discovery algorithm lose information by simply removing infrequent trace variants or infrequent activities from the event log. In contrast, the eST-Miner can consider all information in the event log to discover relations between activities. In particular, we aim to accurately represent behavior patterns with high support in the event log, while ignoring infrequent deviations.

$$
\begin{aligned}
L = [&\langle \blacktriangleright, x_1, x_2, x_3, y_3, y_2, y_1, \blacksquare \rangle^1, \langle \blacktriangleright, x_3, x_2, x_1, y_1, y_2, y_3, \blacksquare \rangle^1,\\
&\langle \blacktriangleright, x_1, x_2, x_3, y_3, y_1, y_2, \blacksquare \rangle^1, \langle \blacktriangleright, x_3, x_1, x_2, y_1, y_2, y_3, \blacksquare \rangle^1,\\
&\langle \blacktriangleright, x_1, x_2, x_3, y_2, y_3, y_1, \blacksquare \rangle^1, \langle \blacktriangleright, x_2, x_3, x_1, y_1, y_2, y_3, \blacksquare \rangle^1,\\
&\langle \blacktriangleright, x_1, x_2, x_3, y_2, y_1, y_3, \blacksquare \rangle^1, \langle \blacktriangleright, x_2, x_1, x_3, y_1, y_2, y_3, \blacksquare \rangle^1,\\
&\langle \blacktriangleright, x_1, x_2, x_3, y_1, y_3, y_2, \blacksquare \rangle^1, \langle \blacktriangleright, x_1, x_3, x_1, y_1, y_2, y_3, \blacksquare \rangle^1]
\end{aligned}
$$

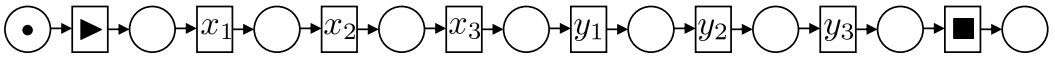


Figure 1. The behavior in event log L corresponds in large parts to the sequential Petri net below. However, in all traces some deviations in activity order occur (marked in red). Since all traces and all activities are equally frequent, it is not possible to filter infrequent behavior patterns and discover the underlying main process structure by simply removing infrequent traces or activities in a preprocessing step. This becomes even more challenging for processes that include concurrency, choice or non-free choice constructs.

Consider the simple example event log and Petri net in Figure 1. The event log exhibits frequent behavioral patterns which are reflected in the behavior of the Petri net. However, in all traces some deviations in activity order occur (marked in red). Often, users do not want a discovery algorithm to return a model including all possible behavior in an event log but rather only the main process structure with noise or deviations filtered out. In this example, it is not possible to discover the main process behavior by simply removing infrequent traces or activities in a preprocessing step. In more complex processes, exhibiting concurrency, choice or non-free choice constructs, this becomes even

more challenging. However, the place-wise perspective of the eST-Miner allows it to ignore deviating parts of a trace not involved with the place currently evaluated. Each place included in the presented Petri net is able to replay a large part of the presented event log and can therefore be discovered when considered in isolation.

The place-wise perspective of the eST-Miner ensures that all occurrences of activities within a log are considered when discovering a model. However, when the set of discovered fitting places is combined in a Petri net, this Petri net allows only for the behavior in the intersection of the behaviors allowed by all inserted places. This intersection may be small or even empty, thus, the Petri net may contain deadlocks or dead parts, resulting in a much lower overall fitness than the fitness of each individual place and an overly complicated model. In extreme cases, the constructed net cannot replay any trace at all, as illustrated by the small example in Figure 2. Here, having inserted places p_6 and p_7 into the Petri net, each of which allows execution of at least 40 traces from the example event log when considered in isolation, together they cause a deadlock and the Petri net discovered cannot fire any transition after the start transition.

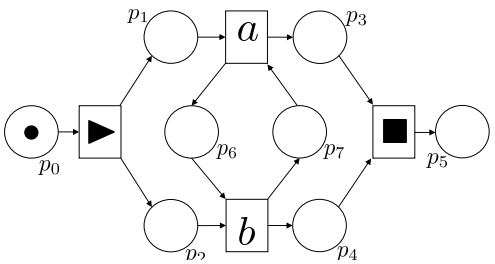


Figure 2. Consider the event log $L = [\langle \blacktriangleright, a, b, \blacksquare \rangle^{40}, \langle \blacktriangleright, b, a, \blacksquare \rangle^{60}]$, where the first trace variant occurs 40 times and the second one 60 times. Considered in isolation, place p_6 allows for the first sequence of activities while place p_7 allows for the second. However, in combination they cause a deadlock in the Petri net.

In this paper, we aim to remedy the issue by investigating strategies of selecting a subset of places which can be combined into a deadlock-free Petri net with definable minimal fitness, while simultaneously striving for high precision and simplicity. Additionally, we introduce a new fitness metric used to evaluate candidate places with the goal to better control how candidate place selection affects the inclusion of infrequent activities into the discovered model. We require the algorithm to maintain its ability to discover and model non-free choice constructs and to provide guarantees as indicated above without over- or underfitting. Furthermore, the time and space consumption should remain reasonable, in particular more scalable than classic region theory approaches.

This paper is a significantly extended and revised version of our work presented in [22]. With respect to this previous work, the main extensions are:

- Section 4 details the root causes of deadlocks and dead parts in the discovered Petri net and introduces our twofold solution approach.
- In Section 5 we introduce a new fitness metric for place evaluation to refine the handling of infrequent activities and thoroughly investigate its properties. In particular, we aim to be able

to maintain the eST-Miner's ability to abstract from deviations and avoid overfitting, while at the same time preventing infrequent activities to cause dead parts in the returned Petri net. We explore the properties of the new metric and show that it can be directly incorporated into the eST-Mining framework. This refines the place candidate evaluation and enables significant improvements of the eST-Miners approach to noise filtering.

- We significantly extend the introduction to the eST-Miner (Section 3), to better motivate our solution approach and provide a basis for the newly introduced fitness metric for place evaluation.
- Section 6 has been partially reworked to clarify the goals and properties of the place selection strategy and correct a mistake related to the adaption functions.
- We refine and extend the experimental evaluation of our previous work and provide an in-depth discussion of the results, in particular with respect to their interpretation in the context of the applied quality metrics. Furthermore, a brief analysis of the algorithm's running time is added.
- We extend and refine the definitions and concepts to make the paper self-contained. Furthermore, Section 4 provides a discussion of the context and assumptions we make, together with detailed examples and motivation for the proposed extensions and refinements. Examples and motivation are extended in other sections as well.

Section 2 provides basic notation and definitions. In Section 3, we briefly review the basics of the standard eST-Miner before providing a detailed problem description and method overview in Section 4. The newly introduced fitness metric and its incorporation into the eST-Miner framework are investigated in Section 5. Section 6 presents our place selection approach, followed by an extensive evaluation in Section 7. Finally, we discuss design choices, open questions and future work in Section 8 before concluding this work with Section 9.

2. Basic notations, event logs, and process models

A set, e.g. $\{a_1, a_2, \dots a_n\}$, does not contain any element more than once. In contrast, a multiset $m\colon A \to \mathbb{N}_0$ over the set A can contain multiples of the same element. We use square brackets to denote multisets with the multiplicity of an element indicated in its exponent, i.e., we write $[a_1^{m(a_1)}, a_2^{m(a_2)}, \dots, a_n^{m(a_n)}]$. For readability, elements with multiplicity 0 and exponents with value 1 are omitted and we write $a \in m$ if $m(a) \geq 1$ holds. The intersection of two sets contains only elements that occur in both sets, i.e., $\{x, y\} \cap \{y, z\} = \{y\}$, while the intersection of two multisets contains each element with its minimum frequency, i.e, $[x, y^2, z] \uplus [y^5, z^2] = [y^2, z]$. Similarly, the union of two sets contains all elements in both sets once, i.e., $\{x, y\} \cup \{y, z\} = \{x, y, z\}$, while the union of two multisets contains all elements with the sum of their frequencies, i.e, $[x, y^2, z] \uplus [y^5, z^2] = [x, y^7, z^3]$. By $\mathbb{P}(X)$ we refer to the power set of the set X, and $\mathbb{M}(X)$ is the set of all multisets over X. We project a multiset m onto a set A by removing all elements not contained in A from m while maintaining other frequencies, e.g., for $A = \{x, y\}$ we have $[x^3, y, z^2]\restriction_A = [x^3, y]$. More formally, $m\restriction_A(a) = m(a)$ if $a \in A$, and $m\restriction_A(a) = 0$ otherwise. In contrast to sets and multisets, where the order of elements is irrelevant, in sequences the elements are given in a certain order, e.g., $\langle x, y, x, y\rangle \neq \langle x, x, y, y\rangle$. We

refer to the i-th element of a sequence σ by $\sigma(i)$. The size of a set, multiset or sequence X, that is $|X|$, is defined to be the number of elements in X. As short notation for the real numbers between 0 and 1 we introduce $\mathbb{R}_0^1 = \{r \in \mathbb{R} \mid 0 \leq r \leq 1\}$.

In the following, we give a standard definition for activities, traces, and logs, except that we require each trace to begin with a designated start activity ($\blacktriangleright$) and end with a designated end activity ($\blacksquare$). Note that it is reasonable to assume a process to have a clear start and end of execution, with each trace describing an end-to-end example run of this process, and that any log can easily be transformed accordingly by adding artificial activities.

Definition 2.1. (Activity, Trace, Event Log)
Let $\mathcal{A}$ be the universe of all possible *activities* (e.g., actions or operations), let $\blacktriangleright \in \mathcal{A}$ be a designated start activity and let $\blacksquare \in \mathcal{A}$ with $\blacksquare \neq \blacktriangleright$ be a designated end activity. A *trace* is a sequence $\sigma = \langle a_1, a_2, \ldots, a_n \rangle$ with $n \geq 2$ such that $a_1 = \blacktriangleright, a_n = \blacksquare$ and $a_i \in \mathcal{A} \backslash \{\blacktriangleright, \blacksquare\}$ for $i \in \{2, 3, \ldots, n-1\}$. Let $\mathcal{T}$ be the universe of all such traces. An *event log* $L \in \mathbb{M}(\mathcal{T})$ is a multiset of traces.

In this paper we extend the eST-Miner algorithm, which, given an event log, returns a Petri net without silent or duplicate transition labels, i.e., a Petri net where each transition can be uniquely identified by its activity label. Therefore, in this paper we can refer to transitions as activities. Furthermore, there is a predefined source place connected to $\blacktriangleright$ marking the start of the process and a predefined sink place connected to $\blacksquare$ marking the end of the process. Besides the two predefined places, we only allow for places connecting transitions that are initially unmarked. These places are uniquely identified by their non-empty sets of input activities I and output activities O (we are not interested in duplicated places). Considering this context, we introduce the following simplified definition for Petri nets.

Definition 2.2. (Places and Petri nets)
A *Petri net* is a pair $N = (A, P)$, where $A \subseteq \mathcal{A}$ is the finite set of activities including start and end, i.e., $\{\blacktriangleright, \blacksquare\} \subseteq A$ and $P' \subseteq \{(I|O) \in \mathbb{P}(A) \times \mathbb{P}(A) \mid I \subseteq A \wedge I \neq \emptyset \wedge O \subseteq A \wedge O \neq \emptyset\}$ is the set of intermediate *places*. We call I the set of *ingoing* activities of a place and O the set of *outgoing* activities. The complete set of places of N is $P = P' \cup \{(\emptyset|\{\blacktriangleright\}), (\{\blacksquare\}|\emptyset)\}$.

Note that if $p = (I|O)$, then $\bullet p = I$ and $p\bullet = O$ using standard notation. To reduce notional overload, we omit set brackets in places, i.e., we write $(\blacktriangleright|\blacksquare)$ instead of $(\{\blacktriangleright\}|\{\blacksquare\})$. Figure 3 shows an example Petri net with places represented by circles and transitions represented by squares.

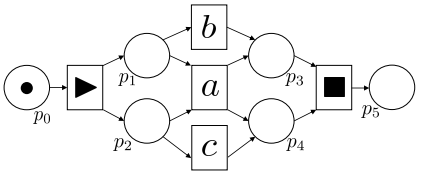


Figure 3. Petri net with $A = \{\blacktriangleright, a, b, c, \blacksquare\}$ and $P = \{(\blacktriangleright|a, b), (\blacktriangleright|a, c), (a, b|\blacksquare), (a, c|\blacksquare), (\emptyset|\blacktriangleright), (\blacksquare|\emptyset)\}$.

The state of a Petri net $N = (A, P)$ is defined by its *marking*. A marking M assigns tokens to places, i.e., $M \colon P \to \mathbb{N}_0$, and can therefore be defined as a multiset of places. The Petri net in

Figure 3 is in marking $[p_0]$ with the corresponding token being represented as a black dot. A transition a is *enabled* in marking M if and only if all places connected to it by an ingoing arc (i.e., all places in the preset of a, $\{(I|O) \subseteq P \mid a \in O\}$) hold at least one token. An enabled transition a can *fire*, thereby changing the marking M of the Petri net to a new marking M' by removing one token from every ingoing place and producing a token in every outgoing place. We write $M \xrightarrow{a} M'$. In the example Petri net only the transition $\blacktriangleright$ can fire initially, consuming the token from its ingoing place p_0 and producing tokens in p_1 and p_2. This enables transitions a, b and c.

Definition 2.3. (Replayable Traces)
Consider a Petri net $N = (A, P)$ and a trace $\sigma = \langle \blacktriangleright, a_1, \ldots, a_n, \blacksquare \rangle$. We say that N can *replay* the trace σ if and only if the transitions corresponding to the activities can be consecutively fired from the marking $M_I = [(\emptyset|\blacktriangleright)]$ to reach the marking $M_F = [(\blacksquare|\emptyset)]$, i.e., there exists a sequence of markings $\langle M_1, \ldots, M_n, M_{n+1} \rangle$ such that $M_I \xrightarrow{\blacktriangleright} M_1 \xrightarrow{a_1} M_2 \ldots M_n \xrightarrow{a_n} M_{n+1} \xrightarrow{\blacksquare} M_F$.

Consider the Petri net in Figure 3. This Petri net can replay the trace $\langle \blacktriangleright, a, \blacksquare \rangle$ as follows: $[p_0] \xrightarrow{\blacktriangleright} [p_1, p_2] \xrightarrow{a} [p_3, p_4] \xrightarrow{\blacksquare} [p_5]$. In contrast, the trace $\langle \blacktriangleright, a, c, \blacksquare \rangle$ is not replayable: after firing $[p_0] \xrightarrow{\blacktriangleright} [p_1, p_2] \xrightarrow{a} [p_3, p_4]$ the next transition to be fired would be c but c is not enabled in $[p_3, p_4]$.

Definition 2.4. (Behavior of a Petri net)
We define the *behavior* of the Petri net $N = (A, P)$ to be the set of traces replayable by N, i.e., *behavior*$(N) = \{\sigma \in \mathcal{T} \mid N \text{ can replay } \sigma\}$.

Note that Definition 2.4 only allows for traces in the behavior of the form $\langle \blacktriangleright, a_1, a_2, \ldots, a_n, \blacksquare \rangle$ (compare Definition 2.1) such that all intermediate places are unmarked at the end of replaying a trace and never have a negative number of tokens. The behavior of the Petri net in Figure 3 is $\{\langle \blacktriangleright, a, \blacksquare \rangle, \langle \blacktriangleright, b, c, \blacksquare \rangle, \langle \blacktriangleright, c, b, \blacksquare \rangle\}$.

3. Introducing the eST-Miner

Several variants and extensions of the eST-Miner have been proposed in the past years. In the following, we introduce the eST-Miner variant used as the basis of this work. For further details, we refer the interested reader to the respective papers.

Method Overview

As input, the algorithm takes a log L over the set of activities $A \subseteq \mathcal{A}$ and a noise threshold $\tau \in \mathbb{R}_0^1$, and returns a Petri net as output. Inspired by language-based regions, the basic strategy of the approach is to begin with a Petri net $N = (A, \{(\emptyset|\blacktriangleright), (\blacksquare|\emptyset)\})$ whose transitions correspond exactly to the activities in L and no intermediate places. From the finite set of unmarked, intermediate candidate places, $\{(I|O) \in \mathbb{P}(A) \times \mathbb{P}(A) \mid I \subseteq A \wedge I \neq \emptyset \wedge O \subseteq A \wedge O \neq \emptyset\}$, the subset of all places *fitting* with respect to L and τ is computed and inserted, where the noise threshold τ allows us to ignore deviations from the main control-flow relations. A candidate place is considered fitting, if it

does not hinder the replay of a significant part of L as indicated by τ. Details on the exact definition and computation of the set of fitting places are provided below.

To facilitate further computations and human readability, *implicit* places are identified and removed [23, 24, 25] from the set of fitting places. A place is implicit if its removal does not increase the number of traces replayable by the Petri net. We have two approaches at our disposal. The first option is to solve optimization problems to identify implicit places based on the structure of the Petri net, as proposed for the original eST-Miner [21]. This approach reliably removes implicit places, but is computationally expensive. The second approach replays the event log to compare the markings of the places and uses this information to compute non-minimal regions, corresponding to implicit places [26]. While it is much faster, for correct and complete implicit place identification certain requirements must be satisfied, e.g. sufficient similarity between the event log and behavior of the Petri net as well as guaranteed inclusion of certain places in the Petri net. Adaptation of this approach to the context of this work is out of scope, therefore, we use the first implicit place removal variant.

We continue with defining the exact meaning of fitting places.

Fitting Places

The eST-Miner considers all possible candidate places based on the activities in the event log. Therefore, in the following, we consider only places whose sets of ingoing and outgoing activities occur in the corresponding event log at least once.

A candidate place can be either *fitting* or *unfitting* with respect to L and τ. We further distinguish *unfitting* places into *underfed* and *overfed* places. These concepts can be used to compute the set of fitting places more efficiently. We introduce them on the level of individual traces first.

Definition 3.1. (Fitting, Underfed and Overfed Places, cf. [27])
Let $p = (I|O) \in \mathbb{P}(\mathcal{A}) \times \mathbb{P}(\mathcal{A})$ be a place, and let σ be a trace. With respect to σ, p is called

- *underfed*, denoted by $\triangledown_\sigma(p)$, if and only if $\exists k \in \{1, 2, \ldots, |\sigma|\}$ *such that* $|\{i \mid i \in \{1, 2, \ldots, k-1\} \wedge \sigma(i) \in I\}| < |\{i \mid i \in \{1, 2, \ldots, k\} \wedge \sigma(i) \in O\}|$
- *overfed*, denoted by $\triangle_\sigma(p)$, if and only if $|\{i \mid i \in \{1, 2, \ldots, |\sigma|\} \wedge \sigma(i) \in I\}| > |\{i \mid i \in \{1, 2, \ldots, |\sigma|\} \wedge \sigma(i) \in O\}|$,
- *fitting*, denoted by $\square_\sigma(p)$, if and only if not $\triangledown_\sigma(p)$ or $\triangle_\sigma(p)$.

A place is underfed with respect to a trace σ if, at some point, replaying σ requires more outgoing transitions of the place to be fired than ingoing transitions have been fired before, implying a lack of tokens in the place. Such a place hinders replay of σ because a transition that should be fired is not enabled. Consider the Petri net in Figure 3 and the trace $\sigma = \langle \blacktriangleright, a, b, \blacksquare \rangle$. The place p_1 is underfed with respect to this trace, because when its outgoing transition b occurs in the trace, not enough ingoing transitions have been fired before to provide the necessary token. Thus, we can conclude that in any Petri net that contains p_1, b is not enabled in the marking reached after firing $\blacktriangleright$ and a.

A place whose ingoing transitions occur more often than its outgoing transitions in a trace σ is called overfed with respect to σ. It hinders replay because it has tokens remaining if the end of replay

is reached, which violates the requirement of ending in marking $[(\blacksquare|\emptyset)]$. The place p_3 is overfed with respect to σ: $\blacktriangleright$ is not connected, a and b each produce a token here, and firing $\blacksquare$ at the end of the trace consumes only one of the two tokens, thus one token is remaining.

Note that a place can be underfed and overfed with respect to a trace at the same time. If a place is neither overfed or underfed with respect to a trace, it is fitting. The place p_2 is fitting with respect to σ. Firing $\blacktriangleright$ produces a token, which is subsequently consumed when a is fired. Neither b or $\blacksquare$ are connected to p_2. Thus, at the end of σ, no token is missing or remaining in p_2.

We can formalize these observations and relate them to the behavior of a Petri net as introduced in Section 2 as follows:

Observation 3.1. (Place Fitness and Firing Sequences)
Given a Petri net $N = (A, P)$ and a trace $\sigma = \langle a_1, a_2, ..., a_n \rangle \in \mathcal{T}$ that is replayable by N, consider a place $p = (I|O)$ with $p \notin P$. The following properties hold for $N' = (A, P \cup \{p\})$:

- $\square_\sigma(p) \Leftrightarrow \sigma \in \textit{behavior}(N')$
- $\triangledown_\sigma(p) \Leftrightarrow \exists k, 0 \leq k \leq n, \textit{ such that } [(\emptyset|\blacktriangleright)] \xrightarrow{a_1} M_1 \xrightarrow{a_2} ... \xrightarrow{a_k} M_k \wedge p \notin M_k \wedge a_{k+1} \in O \Rightarrow \sigma \notin \textit{behavior}(N')$
- $\triangle_\sigma(p) \Leftrightarrow [(\emptyset|\blacktriangleright)] \xrightarrow{a_1} M_1 \xrightarrow{a_2} ... \xrightarrow{a_n} M_n \wedge p \in M_n \Rightarrow \sigma \notin \textit{behavior}(N')$

We define the following functions to succinctly refer to multisets of traces of the event log with respect to which (sets of) places are fitting, underfed or overfed.

Definition 3.2. (Multisets of Fitting/Underfed/Overfed Traces (compare [27]))
Given a set of places $P \subseteq \mathbb{P}(\mathcal{A}) \times \mathbb{P}(\mathcal{A})$ and a an event log $L \in \mathbb{M}(\mathcal{T})$, we define the following functions with respect to a place $p \in P$:

- $\texttt{fitting}_L(p) = L\restriction_{\{\sigma \in L | \square_\sigma(p)\}}$ is the multiset of log traces for which p is fitting
- $\texttt{underfed}_L(p) = L\restriction_{\{\sigma \in L | \triangledown_\sigma(p)\}}$ is the multiset of log traces for which p is underfed
- $\texttt{overfed}_L(p) = L\restriction_{\{\sigma \in L | \triangle_\sigma(p)\}}$ is the multiset of log traces for which p is overfed

We extend these functions to the set of places P as follows:

- $\texttt{fitting}_L(P) = L\restriction_{\{\sigma \in L | \forall p \in P:\, \square_\sigma(p)\}}$.
- $\texttt{underfed}_L(P) = L\restriction_{\{\sigma \in L | \exists p \in P:\, \triangledown_\sigma(p)\}}$.
- $\texttt{overfed}_L(P) = L\restriction_{\{\sigma \in L | \exists p \in P:\, \triangle_\sigma(p)\}}$.

Note that, for a Petri net $N = (A, P)$ and event log L, $\texttt{fitting}_L(P)$ corresponds exactly to the log traces replayable by N, since none of the places in N hinders the replay of those traces.

A fitness metric is used to assign a value to a place, indicating how well it fits the given event log. We require its range to be $\mathbb{R}_0^1$ to facilitate comparison with thresholds and between different metrics.

Definition 3.3. (Measuring Fitness of a Place)
Given an event log $L \in \mathbb{M}(\mathcal{T})$ we define a *fitness metric* to be a function $\mathtt{fm}^L \colon \mathbb{P}(\mathcal{A}) \times \mathbb{P}(\mathcal{A}) \to \mathbb{R}^1_0$, which assigns a fitness score to a place based on L such that 0 represents the lowest achievable fitness, while 1 reflects the highest achievable fitness.

Before introducing concrete fitness metrics, we define the multiset of traces in an event log which contain a given set of activities.

Definition 3.4. (Activated Traces, cf. [27])
Given an event log $L \in \mathbb{M}(\mathcal{T})$ and a set of activities $A \subseteq \mathcal{A}$, we define the multiset of traces *activated* by A as $\mathtt{act}_L(A) = L\restriction_{\{\sigma \in L \mid \exists i \in \mathbb{N} \colon \sigma(i) \in A\}}$.

Two fitness metrics have been proposed so far, in the following called *absolute fitness* and *relative fitness*.

Definition 3.5. (Fitness Metrics, cf. [27])
Let $L \in \mathbb{M}(\mathcal{T})$ be an event log and let $p = (I \mid O)$ be a place. We define two different fitness metrics of the place p with respect to L.

$$\textit{absolute fitness: } \mathtt{fm}^L_{abs}(p) = \frac{|\mathtt{fitting}_L(p)|}{|L|}$$

$$\textit{relative fitness: } \mathtt{fm}^L_{rel}(p) = \frac{|\mathtt{act}_L(I \cup O) \uplus \mathtt{fitting}_L(p)|}{|\mathtt{act}_L(I \cup O)|}$$

Both metrics return values between 0 (low fitness) and 1 (high fitness).

Based on a fitness metric and the noise threshold τ we can lift the notions of fitting and unfitting to the complete event log rather than a single trace.

Definition 3.6. (Fitness with Respect to a Threshold, compare [27])
With respect to an event log $L \in \mathbb{M}(\mathcal{T})$, a threshold $\tau \in \mathbb{R}^1_0$ and a fitness metric $\mathtt{fm}$, we call a place p

- *fitting* if and only if $\mathtt{fm}^L(p) \geq \tau$, and
- *unfitting* if and only if $\mathtt{fm}^L(p) < \tau$.

To increase the efficiency of place candidate evaluation, we need to lift the concepts of underfed and overfed to the log level in a similar way. However, this is done for concrete fitness metrics individually to satisfy certain requirements as explained in the following subsection.

Efficient candidate places evaluation

The eST-Miner uses token-based replay to evaluate each candidate place. The number of possible places is finite but exponential in the number of observed activities. To avoid replaying the log on the exponential number of candidate places and increase efficiency, the eST-Miner aims to skip uninteresting subsets of candidate places, while still guaranteeing to discover all places that are considered

fitting. To this end, it utilizes the following *monotonicity properties* to derive information about candidate places based on a place p already evaluated: If p is underfed with respect to a trace, a candidate place generated only by adding outgoing activities to or removing ingoing activities from p will lack at least as many tokens during replay and will therefore also be underfed for that trace. Vice versa, if p is overfed, a candidate place constructed by adding only ingoing activities or removing outgoing activities will have at least the same number of tokens remaining and is therefore overfed as well. Clearly, this information can be derived without actually evaluating the newly generated places.

Theorem 3.7. (Monotonicity Properties, cf. [27])
Consider two places $p_1 = (I_1 \mid O_1)$ and $p_2 = (I_2 \mid O_2)$. Then the following holds for any event log $L \in \mathbb{M}(\mathcal{T})$:

- $I_1 \supseteq I_2 \wedge O_1 \subseteq O_2 \implies \texttt{underfed}_L(p_1) \subseteq \texttt{underfed}_L(p_2)$
- $I_1 \subseteq I_2 \wedge O_1 \supseteq O_2 \implies \texttt{overfed}_L(p_1) \subseteq \texttt{overfed}_L(p_2)$

The eST-Miner organizes the candidate places as a forest, called the *complete candidate tree*. In this forest, we distinguish between two types of edges: blue edges connect a parent candidate place to a child place constructed from it by adding an outgoing activity, while red edges connect it to a child generated by adding an ingoing activity. A visualization of such a tree-structured candidate space for activities ▶, a, b and ■ is given in Figure 4. If combined with suitable fitness metrics, the monotonicity properties can be used to skip subtrees in the complete candidate tree.

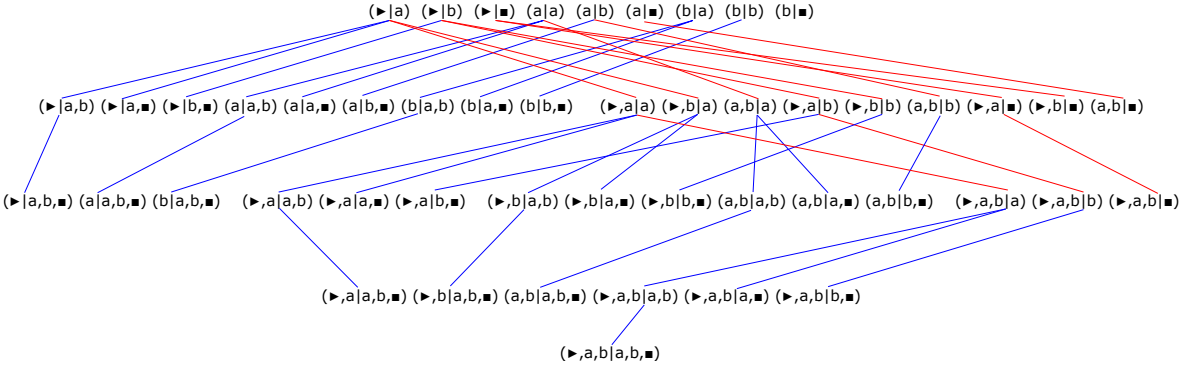


Figure 4. Example of a tree-structured candidate space for the set of activities $\{▶, a, b, ■\}$, with orderings $■ >_i b >_i a >_i ▶$ and $■ >_o b >_o a >_o ▶$.

Definition 3.8. (Complete Candidate Tree)
Let $A \in \mathcal{A}$ be a set of activities and let $>_i, >_o$ be two total orderings on this set of activities. A *complete candidate tree* is a pair $CT = (V, F)$ with

$$V = \{(I|O) \in \mathbb{P}(A) \times \mathbb{P}(A) \mid I \subseteq A \backslash \{■\} \wedge O \subseteq A \backslash \{▶\} \wedge I \neq \emptyset \wedge O \neq \emptyset\}.$$

We have that $F = F_{\text{red}} \cup F_{\text{blue}}$, with

$$\begin{aligned} F_{\text{red}} =& \{((I_1|O_1),(I_2|O_2)) \in V \times V \mid |O_2| = 1 \wedge O_1 = O_2 \\ & \wedge \exists a \in I_1 \colon \big(I_2 \cup \{a\} = I_1 \wedge \forall a' \in I_2 \colon a >_i a'\big)\} \textbf{ (red edges)} \\ F_{\text{blue}} =& \{((I_1|O_1),(I_2|O_2)) \in V \times V \mid I_1 = I_2 \\ & \wedge \exists a \in O_1 \colon \big(O_2 \cup \{a\} = O_1 \wedge \forall a' \in O_2 \colon a >_o a'\big)\} \textbf{ (blue edges)}. \end{aligned}$$

If $((I_1|O_1),(I_2|O_2)) \in F$, we call the candidate $(I_1|O_1)$ the *child* of its *parent* $(I_2|O_2)$.

Note the incremental structure of the trees, i.e., the increase in distance from the roots corresponds to the increase of the number of activities connected to the candidate places. Specifically, for a place $(I|O)$ at depth k we have that $|I| + |O| = k$. Each level of the tree contains all possible places that can be created by connecting the corresponding number of activities (excluding ▶ as an outgoing activity and ■ as an ingoing activity, since such places cannot fit), with the root level consisting of all candidate places that can be build using exactly two transitions. The asymmetry in the definition of red and blue edges (Definition 3.8) ensures that all candidates are reachable from exactly one root by a unique path (using the same rules for both types of edges would result either in some candidates having both, a blue and a red parent, or no parent at all). Thus, candidate traversal following the tree structure guarantees that all candidates are considered exactly once.

Organization of candidates within the same depth and their connections to parent and child candidates is not fixed, but defined by ordering strategies for of ingoing activities ($>_i$) and outgoing activities ($>_o$). With knowledge of these ordering strategies we can deterministically compute the next candidate place to be considered based on the last candidate we evaluated, i.e., based only on the connected activities of the last place. This is very space efficient, since we do not need to keep the whole tree in memory. It also contributes to time efficiency: we do not only avoid evaluation of candidates in skipped subtrees but do not even need to compute and traverse them.

Consider any candidate place $p = (I|O)$ in the complete candidate tree (Definition 3.8). By construction, the tree structure guarantees that for every descendant $p_{blue} = (I_{blue}|O_{blue})$ of p reachable via a path of purely blue edges we have that $I = I_{blue}$ and $O \subseteq O_{blue}$ and thus, by Theorem 3.7, $\texttt{underfed}_L(p) \subseteq \texttt{underfed}_L(p_{blue})$ holds. Respectively, for every descendant p_{red} of p reachable via purely red edges, we have that $\texttt{overfed}_L(p) \subseteq \texttt{overfed}_L(p_{red})$. Combining this property with a suitable fitness metric allows us to cut off subtrees consisting of only unfitting candidate places based on the replay result of the parent candidate place.

In Definition 3.6 we have used the noise threshold τ to lift the concepts of fitting and unfitting places to the log level. To enable skipping of uninteresting subtrees, we have to lift the notions of underfed and overfed places to the log level in a way that is suitable for the applied fitness metric.

Definition 3.9. (Underfed/Overfed Places with Respect to an Event Log, cf. [27])
With respect to an event log $L \in \mathbb{M}(\mathcal{T})$ and a noise threshold $\tau \in \mathbb{R}_0^1$ we define a place $p = (I|O)$ to be *underfed* or *overfed* based on the fitness metric used.

A place unfitting with respect to *absolute fitness* can be classified further as

- *underfed*, denoted by $\bigtriangledown_{abs}^{L,\tau}(p)$, if and only if $\frac{|\mathtt{underfed}_L(p)|}{|L|} > 1 - \tau$,
- *overfed*, denoted by $\bigtriangleup_{abs}^{L,\tau}(p)$, if and only if $\frac{|\mathtt{overfed}_L(p)|}{|L|} > 1 - \tau$.

A place unfitting with respect to *relative fitness* can be classified further as

- *underfed*, denoted by $\bigtriangledown_{rel}^{L,\tau}(p)$, if and only if $\frac{|\mathtt{act}_L(I \cup O) \pitchfork \mathtt{underfed}_L(p)|}{|\mathtt{act}_L(I \cup O)|} > 1 - \tau$,
- *overfed*, denoted by $\bigtriangleup_{rel}^{L,\tau}(p)$, if and only if $\frac{|\mathtt{act}_L(I \cup O) \pitchfork \mathtt{overfed}_L(p)|}{|\mathtt{act}_L(I \cup O)|} > 1 - \tau$.

In Figure 5 we provide an overview illustrating the fitness status a place can have based on Definitions 3.6 and 3.9. A place can either be fitting or unfitting. A place that is unfitting may be further classifiable to be underfed or overfed. A place can be underfed and overfed at the same time, given that sufficiently many traces validate the necessary requirements. It can also be unfitting without being neither underfed nor overfed. Considering the event log $L = [\langle \blacktriangleright, a, a, b, d, \blacksquare \rangle^{60}, \langle \blacktriangleright, a, c, d, d, \blacksquare \rangle^{40}]$ as an example, the following place fitness classifications hold for all fitness metrics discussed in this paper. With respect to L and a noise threshold of $\tau = 0.5$ the place $p_1 = (\blacktriangleright | b)$ is fitting, $p_2 = (c | \blacksquare)$ is underfed, $p_3 = (\blacktriangleright | c)$ is overfed, and $p_4 = (a, d | a)$ is underfed as well as overfed. With respect to L and a noise threshold of $\tau = 0.3$ the place $p_5 = (a | d)$ is unfitting but neither underfed nor overfed.

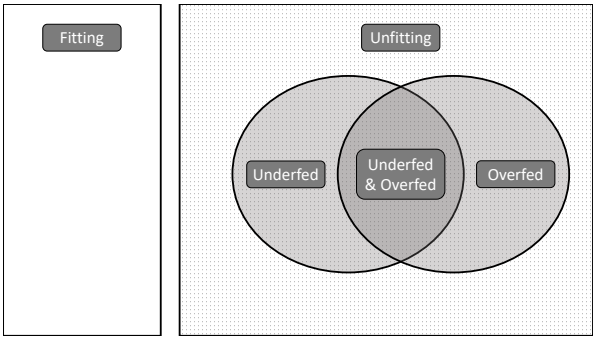


Figure 5. Illustrating the fitness status a candidate place can have with respect to an event log and noise threshold. A place can either be fitting or unfitting. If it is unfitting, it may be underfed, overfed or both, enabling the skipping of derived candidate places. It may also be unfitting (Definition 3.6) without satisfying the threshold to be underfed or overfed (Definition 3.9), in which case no candidates can be skipped based on it.

The prerequisite for the eST-Miner to skip parts of the candidate space is the guarantee, that if a place is determined to be underfed (overfed), the same must hold for all its blue (red) descendants in the complete candidate tree. Due to the construction of the tree and the following lemma this requirement is satisfied for Definition 3.9.

Lemma 3.10. (Monotonicity of Fitness Metrics, cf.[27])
Consider an event log $L \in \mathbb{M}(\mathcal{T})$ and a noise threshold $\tau \in \mathbb{R}_0^1$.

For two places $p = (I|O)$ and $p' = (I|O')$ with $O \subseteq O'$ the following implications hold:

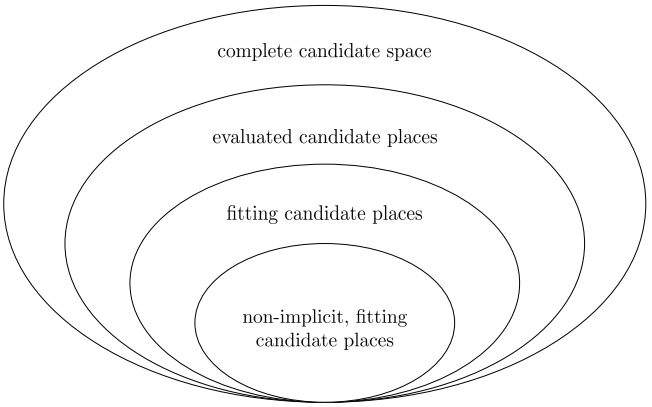
$$\nabla^{L,\tau}_{abs}(p) \implies \nabla^{L,\tau}_{abs}(p'),$$
$$\nabla^{L,\tau}_{rel}(p) \implies \nabla^{L,\tau}_{rel}(p').$$

For two places $p = (I|O)$ and $p'' = (I''|O)$ with $I \subseteq I''$ the following implications hold:

$$\triangle^{L,\tau}_{abs}(p) \implies \triangle^{L,\tau}_{abs}(p''),$$
$$\triangle^{L,\tau}_{rel}(p) \implies \triangle^{L,\tau}_{rel}(p'').$$

Conclusion

The runtime of the eST-Miner strongly depends on the number of candidate places skipped during the search for fitting places. In Figure 6, the subset relations between the sets of places relevant to the eST-Miner framework are visualized. To illustrate the effect of cutting off candidate subtrees, the absolute number of places in these sets is shown for two example applications of the algorithm on two different event logs. While the size of the complete candidate space is the same for both event logs (based on the number of activities), the ordering of the candidates within the tree and properties of the



Set of Places	Teleclaims	RTFM
complete candidate space	16,769,025	16,769,025
evaluated candidates	5,119,151 (69 % skipped)	3,066,180 (82 % skipped)
fitting places	18,139	3,855
non-implicit places	8	4

Figure 6. Comparison of the sets of places considered within the eST-Miner framework. To exemplify the impact of cutting off unfitting subtrees, we choose the well-known `Teleclaims` and `Road Traffic Fine Management` event logs (both have $11 + |\{\blacktriangleright, \blacksquare\}| = 13$ activities) with $\tau = 1.0$ (i.e., requiring all places to be perfectly fitting). The table shows the size relations of the computed sets of places to the complete candidate space (for lexicographical activity orderings $>_i$ and $>_o$ in the complete candidate tree).

event logs themselves result in different sizes for the subset of candidate places to be evaluated. This example emphasizes the performance gain achieved by skipping subtrees.

4. Problem explanation and solution framework

One of the major advantages of the eST-Miner is its ability to ignore infrequent behavior patterns during discovery. This is achieved by computing for each place candidate individually the multiset of log traces it allows to replay. Thus, it can consider and combine information contained in all log traces even if not all of these traces are replayable in the constructed Petri net. This is illustrated by the example in Figure 1: every place in the sequential net allows for a large fraction of the event log to be replayed and the discovered Petri net indeed represents the main behavior contained in the log.

On the downside, its straightforward insertion of fitting places makes the eST-Miner prone to introduce deadlocks into the discovered Petri net. Deadlocking constructs are clearly undesirable, since they may result in a model that does not meet the user's expectations with respect to fitness or at least is unnecessary complicated given the behavior it represents. We use the example event log and corresponding Petri net given in Figure 7 to illustrate the issue of deadlocks, discuss its two causes and motivate our twofold solution approach.

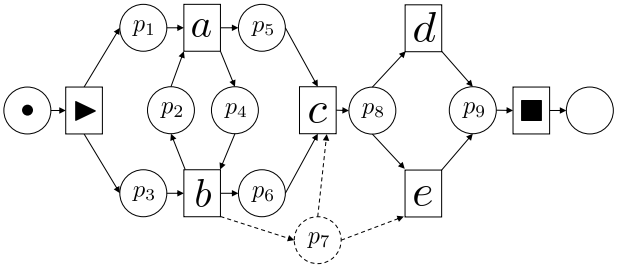


Figure 7. To exemplify the problems addressed in this work we use the Petri net above together with the event log $[\langle \blacktriangleright, a, b, c, d, \blacksquare \rangle^{35}, \langle \blacktriangleright, a, b, c, e, \blacksquare \rangle^{5}, \langle \blacktriangleright, b, a, c, d, \blacksquare \rangle^{55}, \langle \blacktriangleright, b, a, c, e, \blacksquare \rangle^{5}]$. One issue is the insertion of p_7 instead of the more desirable place p_6. Another problem is posed by the interaction between p_2 and p_4.

Deadlocks Related to Place Evaluation

The first reason for deadlocks in the Petri net discovered by the eST-Miner is caused by how the proposed fitness metrics evaluate places that are connected to infrequent activities. Both, the absolute and relative fitness metric, evaluate the fitness of a place based on a fraction of log traces it allows to replay. Places that hinder replay of only a few traces are likely to pass the threshold defined by τ. Places that hinder replay of infrequent activities are hardly penalized by this strategy, since blocking of such activities does not significantly lower the fraction of replayable traces. Therefore, it is likely that returned models include places that block infrequent activities.

Consider the example event log and the place $p_7 = (b|c, e)$ in Figure 7. This place can replay all traces in the event log that do not include the infrequent activity e, i.e., a fraction of 0.9 of all traces.

Inserting this place into the Petri net would block e from being executed completely. However, despite occurring only infrequently, activity e has a clear positioning within the control-flow: whenever it occurs in a trace, it replaces execution of activity d after c and before ■. The place p_6 can perfectly replay the event log (i.e., replays a fraction of 1.0 of all traces) but is removed as implicit when p_7 is added.

In Section 5, we aim to tackle this problem by introducing a new fitness metric to evaluate a place. This strategy maintains a high fitness score for places that adequately represent the control-flow patterns of all their connected activities (e.g., p_6) while places that block too many traces for any of their connected transitions are assigned a low fitness value (e.g., p_7). Additionally, the new metric is defined in such a way that the eST-Miner can still use the evaluation results to improve efficiency by skipping parts of the candidate space.

Deadlocks Related to Combination of Places

The second reason for deadlocks in the Petri net discovered by the eST-Miner is caused by the combination of places that allow for replay of differing parts of the event log. When a candidate place is evaluated to be fitting based on L and τ, it is simply inserted into the Petri net by connecting it to its uniquely labeled ingoing and outgoing transitions. The resulting Petri net can replay exactly the subset of log traces in the intersection of the traces replayable by all inserted places. Independently of the fitness metric used, this may result in Petri nets that can replay a fraction of traces significantly lower than τ, despite each single place satisfying the threshold.

We illustrate the issue using the example in Figure 7. Consider the example event log and the candidate places p_1 to p_6. All of these places can replay at least a fraction of 0.4 traces in the event log and thus would be inserted into the Petri net for any threshold $\tau \leq 0.4$. However, the resulting Petri net would not be able to replay any trace at all: the places $p2$ and $p3$ will block any execution of activities a and b. Two viable solutions exist to obtain a more reasonable model. The first variant, visualized in Figure 8, would be to include neither p_2 nor p_4 and allow a behavior where a and b are executed in parallel.

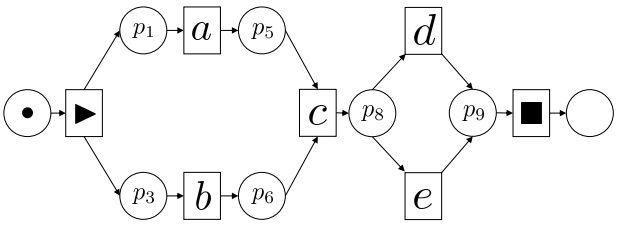


Figure 8. Consider the event log $[\langle \blacktriangleright, a, b, c, d, \blacksquare \rangle^{35}, \langle \blacktriangleright, a, b, c, e, \blacksquare \rangle^{5}, \langle \blacktriangleright, b, a, c, d, \blacksquare \rangle^{55}, \langle \blacktriangleright, b, a, c, e, \blacksquare \rangle^{5}]$. The Petri net above is able to perfectly replay all traces in the event log.

Alternatively, one can focus on the most frequent behavior and insert only p_2, which would result in the removal of the then implicit places p_1 and p_6. This second option is illustrated in Figure 9.

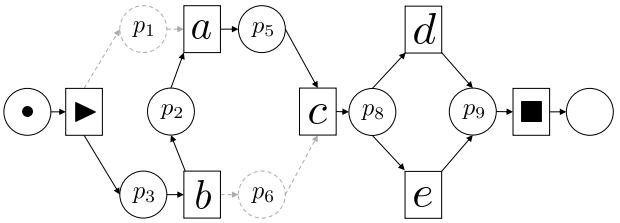


Figure 9. Consider the event log $[\langle \blacktriangleright, a, b, c, d, \blacksquare \rangle^{35}, \langle \blacktriangleright, a, b, c, e, \blacksquare \rangle^{5}, \langle \blacktriangleright, b, a, c, d, \blacksquare \rangle^{55}, \langle \blacktriangleright, b, a, c, e, \blacksquare \rangle^{5}]$. The Petri net above (implicit places marked with gray, dashed lines) is able to replay the two trace variants $\langle \blacktriangleright, b, a, c, d, \blacksquare \rangle^{55}$ and $\langle \blacktriangleright, b, a, c, e, \blacksquare \rangle^{5}$, which constitute a fraction of 0.6 of the input event log.

To avoid the issue of deadlocks caused by combining places, in Section 6 we propose to extend the eST-Mining framework with a place selection strategy that inserts a maximal subset of all fitting places such that the resulting Petri net can replay at least a fraction of τ traces in the event log and does not contain transitions that can never be fired. With respect to the example above, this corresponds to the second solution (Figure 9), which maximizes precision while still satisfying the given fitness threshold.

In the following we give an overview of the algorithmic framework including the proposed extensions.

Algorithmic Framework

This work introduces an algorithmic framework extending the eST-Miner with the goal to address the two different causes for deadlocks described previously. The fitness metric explored in Section 5 refines the evaluation of places. In particular, it enhances the handling of places connected to rarely occurring activities (e.g., p_7 in Figure 7). To prevent deadlocks stemming from combinations of places (e.g., p_2 and p_4 in Figure 7), we replace the straightforward place insertion of the eST-Miner with a strategy preventing such structures. This new place insertion strategy is detailed in Section 6. The goal is to select a subset of fitting places such that the returned Petri net is guaranteed to replay at least a fraction of τ traces in the event log. At the same time, we aim to maximize simplicity and precision of the model.

The proposed algorithm extends the eST-Miner framework by adding and adapting certain submethods. An overview of the framework, including the positioning of our proposed extensions, is given in Figure 10 and described in the following. Details on the extensions will be provided in the corresponding sections.

Initialization: As the only preprocessing steps we add artificial start and end activities ($\blacktriangleright$ and $\blacksquare$) to all traces in the given event log and initialize the output model N as a Petri net with one labeled transition for each activity in the input event log (including $\blacktriangleright$ and $\blacksquare$) and no places except for a marked start place $(\emptyset|\blacktriangleright)$ and a final place $(\blacksquare|\emptyset)$.

Place Insertion: The eST-Miner traverses the complete candidate tree (*Candidate Traversal*) to generate place candidates, which are evaluated one by one, possibly allowing to exploit monotonicity properties to skip uninteresting subtrees. Different traversal strategies of the tree are possible, e.g.

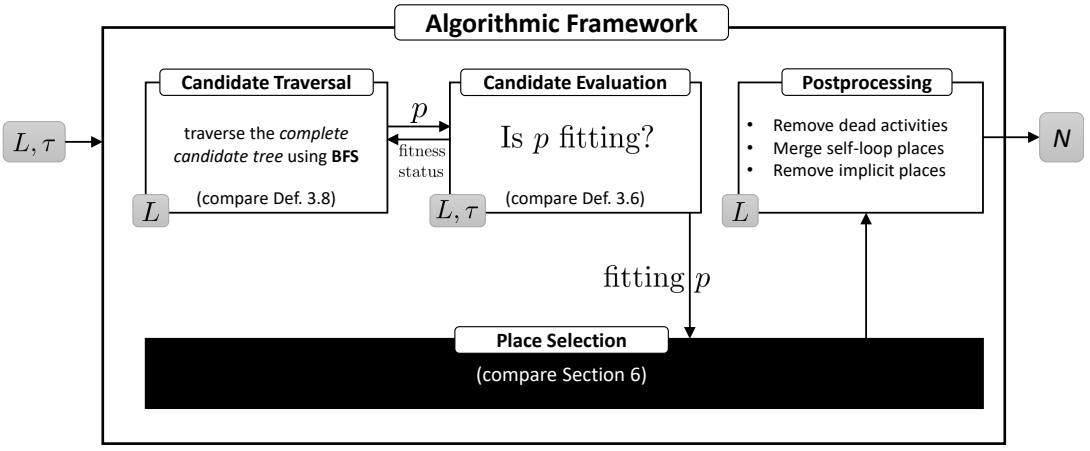


Figure 10. High-level overview of the proposed algorithmic framework. The newly introduced place selection submethod (represented as a black box) is described in detail in Section 6.

Depth-First-Search, Breadth-First-Search or guided by heuristics. It is also possible to limit the depth of tree traversal, improving computation time while losing the option to discover any place below that depth. The extension proposed in this paper, with details provided in Section 6, requires a Breath-First-Search traversal strategy and allows limiting the traversal depth.

Every traversed place candidate is evaluated one by one (*Candidate Evaluation*) to measure how fitting they are with respect to the event log L and the noise threshold τ. We can use absolute fitness, relative fitness or the new metric introduced in Section 5.

The new place selection subroutine (*Place Selection*) following place evaluation is introduced in detail in Section 6. The selected places are added to the Petri net by connecting them to their corresponding uniquely labeled transitions.

Model Simplification: After the selected fitting places have been added, we apply a set of simplification steps (*Postprocessing*). First, the proposed place selection strategy may result in certain activities to be no longer included in the replayable part of the event log. The corresponding transitions are removed from the Petri net (details are provided in Section 6).

Second, all pairs of places of the form $(I \cup A_1 | O \cup A_1)$ and $(I \cup A_2 | O \cup A_2)$ are merged, i.e., replaced by a new place $(I \cup A_1 \cup A_2 | O \cup A_1 \cup A_2)$. This merging of places with self-loops (with A_1 and A_2 being the sets of self-looping transitions) simplifies the Petri net without impacting its behavior. Note that this is necessary only when the tree traversal depth is limited, resulting in the two smaller places being discovered while the larger place connected to more transitions is cut off.

As the final postprocessing step we remove implicit places before returning the resulting Petri net.

Using the Extensions: The newly introduced place selection strategy focuses on guaranteeing minimal fitness of the complete returned Petri net. However, as illustrated before, places that allow for replay of most of the event log may still have a devastating effect on the fitness of single activities, to the degree of blocking their execution completely. Consequently, the proposed fitness metric and the proposed selection technique complement each other but can also be applied independently.

When applying the place selection strategy with relative or absolute fitness, we expect most infrequent activities to be blocked (dead) and consequently be removed from the returned Petri net. This can be desired by users who want to obtain a model reflecting only the most important behavior or subprocess without applying manual preprocessing. However, since removing activities or traces from the event log that do not satisfy a certain frequency threshold is trivial in commonly used tools, it can be reasonable to expect a user to have applied such preprocessing. For users who have applied filtering such that the event log contains only activities and traces they are actually interested in, an ideal model represents the control-flow of all the activities remaining in the event log. We expect that the new fitness metric allows us to continue to abstract from infrequent behavioral patterns without outright removal of infrequent activities or complete trace variants.

Both scenarios are investigated and discussed in our evaluation.

5. A new fitness metric

To prevent the introduction of deadlocks as explained in Section 4, in this section we introduce *aggregated fitness* as a new fitness metric. It refines the fitness evaluation of places with the goal to improve the handling of rarely occurring activities: places simply preventing such actvities from being fired will obtain a low fitness score.

Definition 5.1. (Aggregated Fitness)
Let $L \in \mathbb{M}(\mathcal{T})$ be an event log and let $p = (I|O)$ be a place. We define the fitness metric *aggregated fitness* as

$$\mathtt{fm}^{L}_{agg}(p) = \mathtt{min}_{a \in (I \cup O)} \left(\frac{|\mathtt{act}_L(\{a\}) \pitchfork \mathtt{fitting}_L(p)|}{|\mathtt{act}_L(\{a\})|} \right).$$

Aggregated fitness as introduced in Definition 5.1 returns the minimal fitness computed for each individual transition connected to the place. To maintain the eST-Miners efficiency we require a fitness metric to allow for cutting off subtrees by exploiting the monotonicity properties of the places (Definition 3.7), i.e., the metric has to guarantee that there are no fitting places in the skipped subtrees of the complete candidate tree. In the following, we show that the proposed *aggregated fitness* can satisfy this requirement.

To enable exploitation of the monotonicity results for skipping candidate subtrees while employing the noise threshold τ to filter infrequent behavior patterns, we extend the notions of underfed and overfed (compare Definition 3.1) from single traces to the whole event log in a suitable way:

Definition 5.2. (Aggregated Fitness: Underfed/Overfed Places with Respect to an Event Log)
Consider an event log $L \in \mathbb{M}(\mathcal{T})$, a noise threshold $\tau \in \mathbb{R}^1_0$ and a place $p = (I|O)$. If p is unfitting with respect to L and τ and *aggregated fitness*, it can be classified further as

- *underfed*, denoted by $\bigtriangledown^{L,\tau}_{agg}(p)$, if and only if $\exists a \in I \cup O : \frac{|\mathtt{act}_L(\{a\}) \pitchfork \mathtt{underfed}_L(p)|}{|\mathtt{act}_L(\{a\})|} > 1 - \tau$,
- *overfed*, denoted by $\bigtriangleup^{L,\tau}_{agg}(p)$, if and only if $\exists a \in I \cup O : \frac{|\mathtt{act}_L(\{a\}) \pitchfork \mathtt{overfed}_L(p)|}{|\mathtt{act}_L(\{a\})|} > 1 - \tau$.

As we have seen before for absolute and relative fitness, Definition 5.2 allows for aggregated fitness that a place is underfed and overfed at the same time, given that sufficiently many traces validate the necessary requirements, or is unfitting without being neither underfed nor overfed.

To skip parts of the candidate space without missing fitting places, we need to guarantee, that if a place is determined to be underfed (overfed), the same is guaranteed to be true for its blue (red) descendants in the complete candidate tree.

Lemma 5.3. (Monotonicity of Aggregated Fitness)
Consider an event log $L \in \mathbb{M}(\mathcal{T})$ and a noise threshold $\tau \in \mathbb{R}_0^1$. For three places $p = (I|O)$, $p' = (I|O')$ with $O \subseteq O'$ and $p'' = (I''|O)$ with $I \subseteq I''$the following implications hold:

$$\begin{aligned} \bigtriangledown_{agg}^{L,\tau}(p) &\implies \bigtriangledown_{agg}^{L,\tau}(p'), \\ \bigtriangleup_{agg}^{L,\tau}(p) &\implies \bigtriangleup_{agg}^{L,\tau}(p''). \end{aligned}$$

Proof:
First, assume that $\bigtriangledown_{agg}^{L,\tau}(p)$. Then, by definition, there exists some activity $a \in I \cup O$ such that $\frac{|\mathtt{act}_L(\{a\}) \oplus \mathtt{underfed}_L(p)|}{|\mathtt{act}_L(\{a\})|} > 1 - \tau$. Because of $O \subseteq O'$ we know that $a \in I \cup O'$. Also, by Theorem 3.7 we have that $\mathtt{underfed}_L(p) \subseteq \mathtt{underfed}_L(p')$. Together, this implies that $\frac{|\mathtt{act}_L(\{a\}) \oplus \mathtt{underfed}_L(p')|}{|\mathtt{act}_L(\{a\})|} \geq \frac{|\mathtt{act}_L(\{a\}) \oplus \mathtt{underfed}_L(p)|}{|\mathtt{act}_L(\{a\})|} > 1 - \tau$, and thus by definition $\bigtriangledown_{agg}^{L,\tau}(p')$.

Now, assume that $\bigtriangleup_{agg}^{L,\tau}(p)$. Then, by definition, there exists an activity $a \in I \cup O$ such that $\frac{|\mathtt{act}_L(\{a\}) \oplus \mathtt{overfed}_L(p)|}{|\mathtt{act}_L(\{a\})|} > 1 - \tau$. Because of $I \subseteq I''$ we know that $a \in I'' \cup O$. By Theorem 3.7 we have that $\mathtt{overfed}_L(p) \subseteq \mathtt{overfed}_L(p'')$. Together, this implies that $\frac{|\mathtt{act}_L(\{a\}) \oplus \mathtt{overfed}_L(p'')|}{|\mathtt{act}_L(\{a\})|} \geq \frac{|\mathtt{act}_L(\{a\}) \oplus \mathtt{overfed}_L(p)|}{|\mathtt{act}_L(\{a\})|} > 1 - \tau$, and thus by definition $\bigtriangleup_{agg}^{L,\tau}(p'')$. □

We have shown that the monotonicity properties of the newly introduced *aggregated fitness* enable the same efficiency optimizations by skipping candidate subtrees as the absolute and relative fitness. Thus, this metric can be easily integrated into the eST-Miners framework. To compare the impact on the place fitness evaluation, we continue to investigate relationships between these fitness metrics.

Lemma 5.4. (Relationships Between Fitness Metrics)
Let $L \in \mathbb{M}(\mathcal{T})$ be an event log and let $p = (I|O)$ be a place. Then the following holds:

$$\mathtt{fm}_{rel}^{L}(p) \leq \mathtt{fm}_{abs}^{L}(p).$$

Proof:
With respect to the place p, we can partition the event log L into the following disjunct multisets of traces:

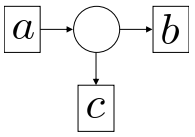
$L_{\checkmark}^{\square} = \mathtt{fitting}_L(p) \uplus \mathtt{act}_L(I \cup O)$, *the multiset of fitting, activated traces.*
$L_{\checkmark}^{\boxtimes} = (\mathtt{underfed}_L(p) \uplus \mathtt{overfed}_L(p)) \uplus \mathtt{act}_L(I \cup O)$, *the multiset of unfitting, activated traces.*
$L_{\times} = L \backslash \mathtt{act}_L(I \cup O)$, *the multiset of non-activated traces.*

Note that $|L^{\square}_{\checkmark}| + |L^{\boxtimes}_{\checkmark}| > 0$, since we consider only activities contained in L. In the following, we use these sets to rewrite absolute and relative fitness and show the claimed relation between them:

$$
\begin{aligned}
& \mathtt{fm}^{L}_{rel}(p) \leq \mathtt{fm}^{L}_{abs}(p) \\
\Leftrightarrow \quad & \frac{|L^{\square}_{\checkmark}|}{|L^{\square}_{\checkmark}| + |L^{\boxtimes}_{\checkmark}|} \leq \frac{|L^{\square}_{\checkmark}| + |L_{\times}|}{|L^{\square}_{\checkmark}| + |L^{\boxtimes}_{\checkmark}| + |L_{\times}|} \\
\Leftrightarrow \quad & |L^{\square}_{\checkmark}| \cdot \left(|L^{\square}_{\checkmark}| + |L^{\boxtimes}_{\checkmark}| + |L_{\times}|\right) \leq \left(|L^{\square}_{\checkmark}| + |L_{\times}|\right) \cdot \left(|L^{\square}_{\checkmark}| + |L^{\boxtimes}_{\checkmark}|\right) \\
\Leftrightarrow \quad & |L^{\square}_{\checkmark}|^2 + |L^{\square}_{\checkmark}| \cdot |L^{\boxtimes}_{\checkmark}| + |L^{\square}_{\checkmark}| \cdot |L_{\times}| \leq |L^{\square}_{\checkmark}|^2 + |L_{\times}| \cdot |L^{\square}_{\checkmark}| + |L^{\square}_{\checkmark}| \cdot L^{\boxtimes}_{\checkmark}| + |L_{\times}| \cdot |L^{\boxtimes}_{\checkmark}| \\
\Leftrightarrow \quad & 0 \leq |L_{\times}| \cdot |L^{\boxtimes}_{\checkmark}|
\end{aligned}
$$

□

We have shown that relative fitness is at least as strict as absolute fitness. Therefore, using relative fitness instead of absolute fitness during candidate place evaluation guarantees the discovered places to be at least as fitting. Unfortunately, no similar relation holds between $\mathtt{fm}^{L}_{agg}(p)$ and $\mathtt{fm}^{L}_{rel}(p)$ as illustrated by the example in Figure 11. Consider the given place p and event log L_1: we have that $\mathtt{fm}^{L_1}_{agg}(p) = \mathtt{min}(\frac{90}{90}, \frac{0}{10}) = 0$, which is strictly smaller than $\mathtt{fm}^{L_1}_{rel}(p) = \frac{90}{100}$. However, considering the same place p and the event log L_2, we have that $\mathtt{fm}^{L_2}_{agg}(p) = \mathtt{min}(\frac{33}{33}, \frac{33}{66}, \frac{33}{66}) = \frac{1}{2}$ is strictly larger than $\mathtt{fm}^{L_2}_{rel}(p) = \frac{33}{99} = \frac{1}{3}$.



$$
\begin{aligned}
L_1 &= [\langle a, b\rangle^{90}, \langle x, y\rangle^{20}, \langle c\rangle^{10}] \\
L_2 &= [\langle a, b, a, c\rangle^{33}, \langle x\rangle^{1}, \langle b\rangle^{33}, \langle c\rangle^{33}]
\end{aligned}
$$

Figure 11. Traces in L_1 and L_2 which are unfitting with respect to the place $p = (a|b, c)$ are marked in red, while traces not activated are colored gray. For L_1 we have that $\mathtt{fm}^{L_1}_{agg}(p) < \mathtt{fm}^{L_1}_{rel}(p)$, while for L_2 we have that $\mathtt{fm}^{L_2}_{agg}(p) > \mathtt{fm}^{L_2}_{rel}(p)$.

To obtain a metric that guarantees to satisfy a given fitness threshold τ with respect to all three fitness metrics discussed, we define *combined fitness* as a final fitness metric. Combined fitness simply takes the minimum of all three fitness metrics and therefore inherits their monotonicity properties, i.e., when using it to skip parts of the candidate space we maintain the guarantee to not skip any fitting place candidate. More specifically, if a place is fitting with respect to combined fitness, then it is also fitting with respect to each individual fitness metric, and if a place is underfed (overfed) with respect to at least one of the individual metrics then the descendants in the corresponding subtree are also underfed (overfed) with respect to that fitness metric.

Definition 5.5. (Combined Fitness of a Place)
Let $L \in \mathbb{M}(\mathcal{T})$ be an event log and let $p = (I|O)$ be a place. We define the *combined fitness* of the place p with respect to L as

$$\mathtt{fm}^{L}_{comb}(p) = \mathtt{min}(\mathtt{fm}^{L}_{abs}(p), \mathtt{fm}^{L}_{rel}(p), \mathtt{fm}^{L}_{agg}(p)).$$

Combined fitness is rather restrictive, mostly due to the newly introduced aggregated fitness. A more forgiving aggregation function than the minimum, e.g., such as average, median or harmonic mean, would be interesting to investigate as an alternative. Unfortunately, such aggregations would not allow for the skipping of candidate subtrees in the eST-Miners complete candidate tree, since they do not provide the same monotoncity properties. For example, consider the place $p = (a|b, c)$ and the event log $L = [\langle \blacktriangleright, a, b, b, \blacksquare \rangle^1, \langle \blacktriangleright, a, c, \blacksquare \rangle^1, \langle \blacktriangleright, a, d, \blacksquare \rangle^1]$. The aggregated fitness of p as defined is $\mathtt{fm}^L_{agg}(p) = \mathtt{min}(\frac{2}{3}, \frac{0}{1}, \frac{1}{1}) = 0$ and p would be considered underfed because of b, allowing us to skip its blue subtree. When taking the average instead of the minimum, the score would be $\mathtt{average}(\frac{2}{3}, \frac{0}{1}, \frac{1}{1}) \approx 0.55$. Consider now the place $p' = (a|b, c, d)$ constructed from p by adding the outgoing activity d. The aggregated fitness of p' is $\mathtt{fm}^L_{agg} = \mathtt{min}(\frac{2}{3}, \frac{0}{1}, \frac{1}{1}, \frac{1}{1}) = 0$ and p' would also be underfed, as expected. However, when taking the average instead of minimum, the score would be $\mathtt{average}(\frac{2}{3}, \frac{0}{1}, \frac{1}{1}, \frac{1}{1}) \approx 0.66$. This example clearly illustrates that the monotonicity properties necessary to skip subtrees without losing guarantees do not hold for such aggregation functions.

The *combined fitness* is the strictest fitness metric introduced in this work and therefore guarantees that any (set of) places satisfying a threshold τ with respect to combined fitness also satisfies τ with respect to absolute, relative and aggregated fitness. We can still choose to be less restrictive by lowering the threshold τ. Note that due to Lemma 5.4 we do not need to compute absolute fitness in the implementation of combined fitness. In Section 7 we apply the proposed eST-Miner variant with *combined fitness* as well as *relative fitness* as a fitness metric to various event lots to evaluate their impact on real-life data.

6. Place selection

The eST-Miner evaluates all candidate places and discovers a set of places fitting the input event log based on the noise threshold τ and the chosen fitness metric. As illustrated by the simple example in Figure 7, simply adding all fitting places to a Petri net indiscriminately may result in deadlocks and unnecessary complexity. In this section, we propose an approach aiming to mitigate this problem by selecting a suitable subset of places.

To motivate our strategy, we first discuss a more complex example. Consider the event log L and the set of places p_1 to p_8 in Figure 12. For each of the three fitness metrics introduced in Definitions 3.5 and 5.1, all places in this (incomplete) subset of candidate places are *fitting* with respect to L and $\tau = 0.75$. Inserting all of these places results in the given Petri net N, which can replay only the first trace variant in L, corresponding to 60 % of the traces. The introductory example in Figure 7 illustrates, that the fraction of replayable traces may even decrease to 0. Such a result is undesirable, since it is unnecessarily complex with respect to the behavior it represents, not free of dead parts and likely to disappoint user expectations with respect to fitness. In the following, we explore strategies to return a deadlock free Petri net that is guaranteed to replay at least a fraction of τ traces in the event log by inserting only a selection of the discovered fitting places.

Consider the set of all fitting places discovered during the place candidate evaluation of the eST-Miner. Selecting an adequate subset of these places, such that also the resulting Petri net as a whole satisfies the noise threshold τ, is challenging for a variety of reasons. While trivial solutions exist, such

ID	Traces in L	p_1 $(\blacktriangleright \mid a)$	p_2 $(a \mid c)$	p_3 $(a \mid b)$	p_4 $(c \mid e)$	p_5 $(b \mid e)$	p_6 $(e \mid \blacksquare)$	p_7 $(b \mid c, d)$	p_8 $(d, e \mid \blacksquare)$	N
1	$\langle \blacktriangleright, a, b, c, e, \blacksquare \rangle^{60}$	✓	✓	✓	✓	✓	✓	✓	✓	✓
2	$\langle \blacktriangleright, a, b, d, \blacksquare \rangle^{20}$	✓	×	✓	✓	×	×	✓	✓	×
3	$\langle \blacktriangleright, a, c, b, e, \blacksquare \rangle^{15}$	✓	✓	✓	✓	✓	✓	×	✓	×
4	$\langle \blacktriangleright, a, b, d, e, \blacksquare \rangle^{5}$	✓	×	✓	×	✓	✓	✓	×	×

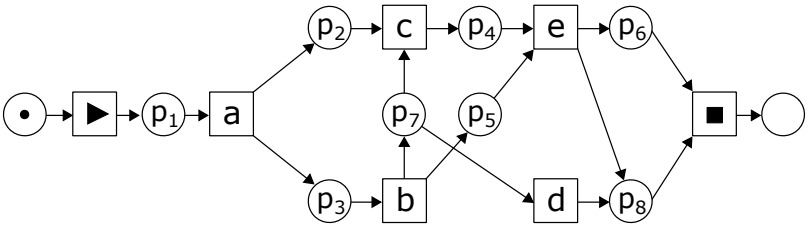


Figure 12. The table indicates for each of the given trace variants and candidate places whether the place can replay that trace variant. The Petri net N is created by inserting all these places and can replay only the first trace variant, i.e., $0.6 \cdot |L| = 60$ traces.

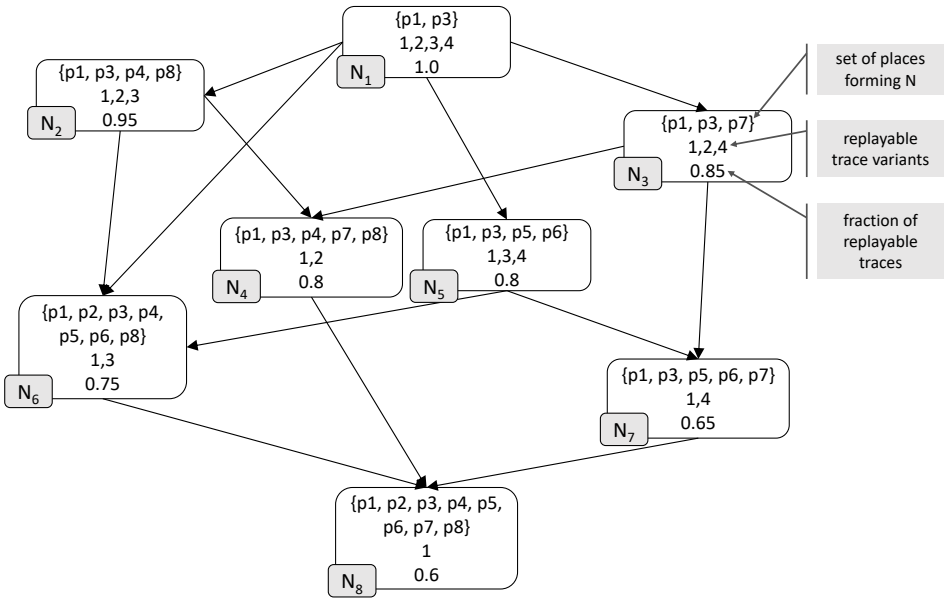


Figure 13. Consider the set of places given in Figure 12. This figure shows all possible combinations of these places such that adding any other place to the corresponding Petri net would decrease the number of replayable log traces. Each set of places, i.e., Petri net, is annotated with the list of trace variants it can replay and the corresponding fraction of log traces. Note that N_8 corresponds to the Petri net shown in Figure 12.

as not inserting any place at all, defining what the best solution would be is not straightforward, since several maximal subsets of places satisfying this requirement may exist. These subsets may differ, for example, in size, fraction of replayable traces, place complexity (number of connected activities) or subjective 'interestingness' measures. Figure 13 illustrates all maximal sets of places that can be built from the example places given in Figure 12. These sets are maximal in the sense that adding any of the other places would decrease the number of replayable log traces. Depending on the choice of the minimal fitness threshold τ, the solution considered optimal by the user is unclear.

Furthermore, even if we have somehow obtained a notion of optimality, first collecting all fitting places and then computing an optimal solution can quickly become unfeasible, both in terms of time complexity and memory requirements. This is due to the huge number of fitting but potentially implicit places discovered by the eST-Miner. Unfortunately, knowledge of which places are contained in the Petri net is required to identify implicit places reliably.

To circumvent the issue of time and space complexity, we combine the eST-Miners sequential place evaluation procedure with a guided greedy place selection approach, which is described in detail in Subsections 6.1 and 6.2. In the absence of a clear notion of optimality, we propose and investigate several heuristic selection strategies and evaluate their impact on different quality aspects of the returned Petri net. In this paper, we consider fitness, precision, and simplicity as desirable properties.

A model with high fitness can express most of the behavior seen in the event log. High precision means that the model does not allow for a lot of behavior not seen in the event log. Simplicity refers to the general readability and understandability of the model and is therefore inherently subjective. In this work, we approximate simplicity by assuming that fewer arcs indicate a simpler model. While generalization (avoiding overfitting with respect to the sample process executions given in the event log) is desirable, additional information would be required to evaluate it, which is why we consider it outside the scope of this work. The concrete metrics used in this work to evaluate quality will be briefly discussed in Section 7. For further details, we refer the reader to [5, 28].

6.1. Place classification

When making the decision to insert a place into the model, this reduces the possible choices we can make later on: the place constrains the behavior of the model and only places with a sufficiently large intersection of replayable traces can be added to the model at a later point. Consider the example place combinations in Figure 13 with a fitness threshold of $\tau = 0.75$ and assume that the model already contains the places p_1 and p_3. If the next fitting place we discover is p_7, and we immediately insert it into the Petri net, we can no longer discover a Petri net including, for example, p_6 without violating our fitness constraint. Such choices may prevent us from discovering a more desirable solution. Therefore, we aim to capture the main behavior of the log by using heuristics to postpone, or even disallow, the addition of very restrictive places.

To this end, we introduce a new parameter δ which is our main tool to guide the choice of places while balancing fitness, precision, and simplicity. This δ specifies the largest acceptable reduction in replayable traces when adding a place to the model. Optionally, δ can be adapted for each place individually using an adaption function `adapt` to favor certain places over others, according to the user's preferences. Favored places can be added earlier, despite being rather restrictive, while other

places will be added only if they do not constrain the behavior too much. Such adaption strategies are discussed in Section 6.3.

Definition 6.1 formalizes the use of τ, δ and $\texttt{adapt}$ to decide for a newly discovered fitting place p, whether the algorithm should *add* it to the selected set of places P, *keep* it for later re-evaluation or *discard* it forever.

Definition 6.1. (Place Classification Using τ, δ and $\texttt{adapt}$)
Consider an event log $L \in \mathbb{M}(\mathcal{T})$ over the set of activities $A \in \mathcal{A}$, a set of places $P \subseteq \mathbb{P}(A) \times \mathbb{P}(A)$, and a place $p \in \mathbb{P}(A) \times \mathbb{P}(A)$. We use parameters $\tau \in \mathbb{R}_0^1$ and $\delta \in \mathbb{R}_0^1$, and a function $\texttt{adapt}\colon \mathbb{R}_0^1 \times (\mathbb{P}(A) \times \mathbb{P}(A)) \to \mathbb{R}_0^1$ to categorize p as follows:

$$\texttt{keep}_{L,\tau}(P,p) = |\texttt{fitting}_L(P) \uplus \texttt{fitting}_L(p)| \geq \tau \cdot |L|$$
$$\texttt{add}_{L,\tau,\delta}(P,p) = \texttt{keep}_{L,\tau}(P,p) \wedge |\texttt{fitting}_L(P)| - |\texttt{fitting}_L(P) \uplus \texttt{fitting}_L(p)| \leq \texttt{adapt}(\delta,p) \cdot |L|$$

If $\texttt{keep}_{L,\tau}(P,p)$ does not hold, p will be discarded.

In the following subsection, we give an overview of the complete approach.

6.2. Selection framework

An overview of our approach, indicating inputs, outputs and use of parameters, is given in Figure 14. Since we consider the simplicity of the model to be a desirable property, we set the eST-Miner to

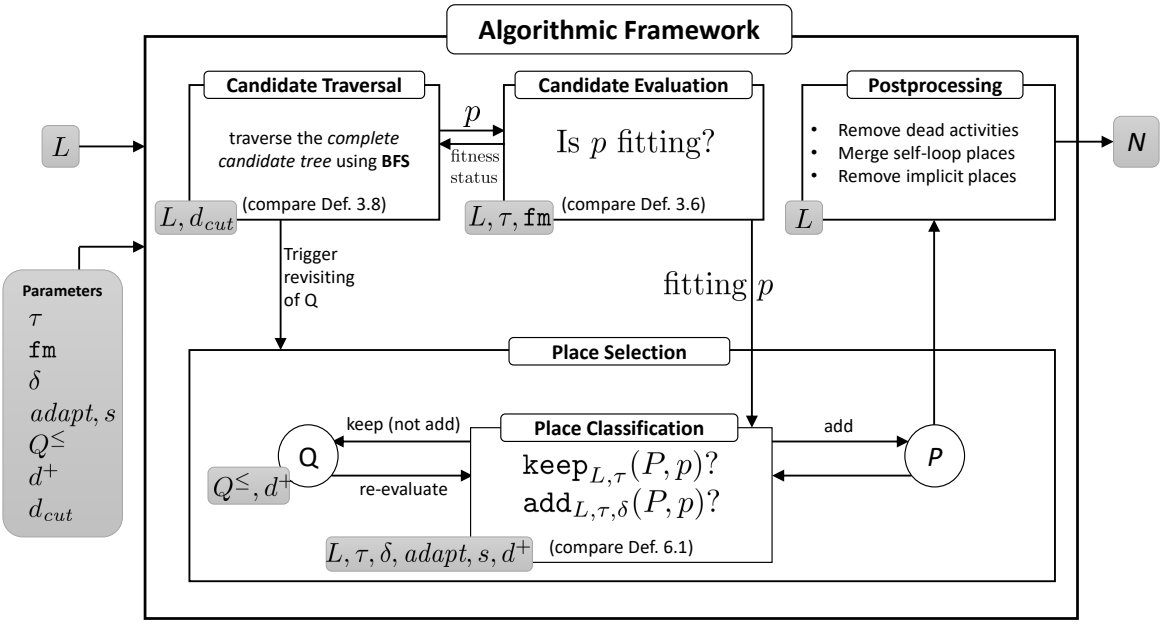


Figure 14. Overview of the presented approach, including input, output, and parameter use.

traverse the *complete candidate tree* using BFS rather than DFS. Thus, places with few connected activities are evaluated first and can therefore be inserted into the model at an earlier stage. Furthermore, we limit the traversal depth to places with d_{cut} activities. This can not only significantly improve the running time but also balances precision and fitness on the one hand and simplicity and generalization on the other hand by preventing the algorithm to discover complex places located at deeper levels of the complete candidate tree. While some complex behaviors may need such places to be expressed exactly, inserting them into the Petri net is usually devastating to readability and in practical applications their constraints can often be sufficiently approximated by much simpler places.

After the eST-Miner framework evaluates a candidate place p to be fitting with respect to a threshold τ, we use the *classification functions* given in Definition 6.1 to decide whether the place should immediately be added to the output Petri net, discarded forever or kept for re-evaluation. In the latter case it is added to a queue Q of potential places which is sorted according to how interesting a place is. In our case, we sort first by place simplicity (few transitions are better) and second by the number of replayable log traces. Optionally, the length of Q can be limited to $Q^{\leq}$ with the least interesting places being dropped if necessary, thus trading an improvement in time and space complexity for potentially lowered model quality.

Whenever the BFS candidate traversal reaches a new level in the *complete candidate tree*, we revisit the potential places queue Q and re-evaluate its places using the classification functions before proceeding with the traversal of more complex places. This makes sense to promote simplicity in particular together with the sigmoid delta adaption function proposed in Section 6.3, which gives preference to places less complex than indicated by the current tree level. After reaching the lowest tree level, the approach continues to iterate over the potential places queue repeatedly d^+ times. This can be relevant for delta adaption functions depending on place complexity and current tree depth, as exemplified in Section 6.3: with each iteration the current tree depth parameter of the adaption functions is incremented (hence the term *artificial tree depth*) allowing for gradually increased leniency also for the most complex places evaluated.

Finally, the resulting Petri net $N = (A, P)$ may contain dead parts: activities which occur only in the subset of log traces that are no longer replayable by N are not guaranteed to be executable at all. Therefore, as a final step, we detect and remove all activities that do not occur in $\texttt{fitting}_L(P)$ together with their connected arcs. Before returning this Petri net as final output, the eST-Miner framework removes implicit places, and merges self-looping places when applicable (see Section 4).

The approach returns a Petri net N satisfying the following guarantees.

Theorem 6.2. (Guarantees)
Given an event log $L \in \mathbb{M}(\mathcal{T})$ over activities A, parameters $\tau \in \mathbb{R}_0^1, \delta \in \mathbb{R}_0^1, s \in \mathbb{N}, Q^{\leq} \in \mathbb{N}, d^+ \in \mathbb{N}, d_{cut} \in \mathbb{N}$ and an adaption function $\texttt{adapt}\colon \mathbb{R}_0^1 \times (\mathbb{P}(A) \times \mathbb{P}(A)) \to \mathbb{R}_0^1$, the eST-Miner extended with the place selection strategy given in Definition 6.1 computes a Petri net $N = (A', P)$ with $A' \subseteq A$, such that N can replay at least $\tau \cdot |L|$ traces from L and every transition in A' can be fired at least once.

Proof:
The algorithm initializes the Petri net $N_0 = (A, \{(\emptyset|\blacktriangleright), (\blacksquare|\emptyset)\})$ with one transition for each activity in L. There is no place constraining the behavior of N_0 except for $\{(\emptyset|\blacktriangleright)$ and $(\blacksquare|\emptyset)\}$, which allow

for ▶ and ■ to be fired exactly once each. According to our trace definition (Definition 2.1), these activities do occur exactly once in each trace. Thus, N_0 can replay at least $\tau \cdot |L|$. The method then iteratively adds places. According to Definition 6.1 a place p can be added to a Petri net $N_1 = (A_1, P_1)$ only if $\texttt{add}_{L,\tau,\delta}(P_1, p))$ holds, which requires $\texttt{keep}_{L,\tau}(P_1, p)$ to hold. This requirement ensures that $|\texttt{fitting}_L(P_1) \uplus \texttt{fitting}_L(p)| \geq \tau \cdot |L|$, i.e., the Petri net with the place p added, $N_2 = (A_1, P_1 \cup \{p\})$ can replay at least $\tau \cdot |L|$ traces from L.

Since the requirement must hold for all added places, no further transitions are added and every transition that is not part of the replayable traces is removed, the final returned Petri net $N = (A', P)$ can replay at least $\tau \cdot |L|$ traces from L, $A' \subseteq A$ holds and every transition can be fired at least once (when replaying a trace including the corresponding activity). □

Furthermore, if the length of Q is not limited, and thus a place p is discarded only if it does not satisfy $\texttt{keep}_{L,A,\tau}(P, p)$, the set of places P is maximal in the sense that no place from the set of evaluated candidate places can be added without violating the fitness constraints imposed by the chosen heuristics.

6.3. Selection strategies

As illustrated by the example place combinations in Figure 13, the order of places added can have a significant impact on the selected subset of places and, thus, the behavior of the returned Petri net. The presented framework allows for a wide range of heuristic functions, optimizing the place selection individually towards a variety of possible user interests. Thus, obviously, the examples presented in the following are by far not exhaustive and entirely different choices are possible, but they can serve as a starting point for an investigation of the impact and suitability of our approach.

The *sigmoid* delta adaption function aims to promote fitness and simplicity. The *constant* and *no delta* delta adaption functions are introduced to be used as a baseline in our experiments, towards which the effect of the sigmoid delta adaption function can be compared.

No Delta

As a baseline to compare to, we introduce a function that ignores the parameter δ and simply adds every fitting place to the Petri net as soon as it is discovered. Within the framework, this can be formalized to

$$\texttt{adapt}_{\textit{noDelta}}(\delta, p) = 1.$$

Constant Delta

Trivially, we can choose not to adapt delta at all. We simply add every fitting, non-discarded place that does not reduce the replayable traces from the log by a fraction of more than delta. Formally, this resembles the identity function:

$$\texttt{adapt}_{\textit{constant}}(\delta, p) = \delta$$

Sigmoid Delta Adaption

While optimizing towards fitness as well as simplicity, we can balance the two forces in different ways based on the details of the adaption function. In this work investigate the *sigmoid* delta adaption as defined in the following and visualized in Figure 15.

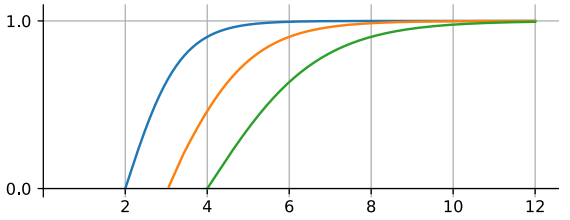


Figure 15. Example behavior of the delta adaption modifier $mod^{s,d}_{sigmoid}(\delta)$ for three places with $2, 3$, and 4 activities, respectively. The x-axis indicates the current tree depth d, with $d_{max} = 12$, while the y-axis indicates the modifier to be multiplied with δ.

Given a set of activities A, the maximum depth of the complete candidate tree is $d_{max} = 2|A|$. Furthermore, let $d \in [2, 3, \ldots, d_{max}]$ be the current depth of the candidate tree traversal. We call $s \in \mathbb{N}\backslash\{0\}$ the *steepness modifier*.

Consider a place $p = (I|O)$. We define the *sigmoid delta adaption function* as follows:

$$\begin{aligned}\texttt{adapt}^{s,d}_{sigmoid}(\delta, (I|O)) &= \delta \cdot mod^{s,d}_{sigmoid}((I|O)) \\ &= \delta \cdot \left(\frac{2}{1 + e^{\left((-1)\cdot \frac{s}{(|I|+|O|)} \cdot (d-(|I|+|O|))\right)}} - 1 \right)\end{aligned}$$

The adaption function multiplies a modifier with the parameter δ. Figure 15 illustrates the behavior of this modifier for three example places of varying complexity. The *sigmoid* delta adaption is designed to prefer simple places. When a place originates from the currently traversed level of the *complete candidate tree*, i.e., it is among the most complex places currently available, the function will evaluate to 0, meaning that only a perfectly fitting place can be added. The simpler the evaluated place is compared to the current tree level, the larger the result of the function and the more unfitting traces are allowed, with δ marking the maximal returnable value. The modifier grows fast in the beginning, but stagnates towards the end, preferring the simpler places more strongly, while the more complex places are (roughly) equally undesirable. The steepness modifier s controls the intensity of the growth.

7. Experimentation and evaluation

We performed several experiments where we run the proposed algorithm with a wide variation of combinations of possible parameter settings on several event logs with different properties. The focus of this paper is on avoiding deadlocks and dead parts in the returned models, which is why the main focus of this evaluation is on the quality of the discovered models. In the end of this section we will

briefly discuss the performance, in particular showing that the extension proposed in this work does not add significantly to the running time of the eST-Miner.

7.1. Experimental setup

Table 1 provides an overview of the event logs used in our experimentation. `Sepsis` has a relatively high number of different trace variants, all of which have comparable frequencies, with the most frequent trace variant making up only 3.33 % of the event log. Activities are repeated often within a

Table 1. List of logs used for the evaluation. The upper part lists real-life logs while the lower part shows artificial logs. Logs are referred to by their abbreviations.

Log Name	**Abbreviation**	**Activities**	**Trace Variants**	**Reference**
Sepsis	`Sepsis`	16	846	[29]
Road Traffic Fine Management	`RTFM`	11	231	[30]
Teleclaims	`Teleclaims`	11	12	[5]
Order-Handling	`Orders`	8	9	[31]

Table 2. Overview of the parameter settings used in our experimentation. The combinations result in 6300 runs for each event log. The value ranges were chosen based on a smaller set of preliminary experiments, aiming to investigate a wide range of parameter settings on the one hand, while on the other hand avoiding unnecessary complexity resulting from variation without notable impact. For example, for our inputs no places were discarded for $Q^{\leq} \geq 10000$. For d^{+} we chose a very low and a very high value to evaluate whether it had any impact at all. Finally, for the chosen event logs $d_{cut} = 5$ has shown to be sufficient to find complex structures with the standard eST-Miner, i.e., increasing the traversed tree depth increases computation time but has no strong impact on model quality.

Parameter	**Used Values**	**Purpose**
τ	0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9	Defines the minimal fraction of log traces that every place, as well as the final Petri net, must be able to replay.
δ	0.05, 0.1, 0.15, 0.2, 0.25	Used to define the allowed reduction in log traces replayable by N when adding a place.
`fm`	$\texttt{fm}_{rel}$, $\texttt{fm}_{comb}$	Defines the fitness metric to be used.
`adapt`	$\texttt{adapt}_{noDelta}$, $\texttt{adapt}_{constant}$, $\texttt{adapt}_{sigmoid}$	The delta adaption function used to guide the heuristics.
s	1, 2, 3, 4, 5	The steepness of the increase of the adaption function (relevant for $\texttt{adapt}_{sigmoid}$ only).
$Q^{\leq}$	100, 1000, 10000	The maximal number of places stored in Q.
d^{+}	0, 10	Artificial tree depth to re-evaluate places in Q after end of tree traversal (relevant for $\texttt{adapt}_{sigmoid}$ only).
d_{cut}	5	Stop candidate traversal after the specified tree level.

trace, which must lead to looping behavior within a Petri net with uniquely labeled transitions. `RTFM` is rather large, with a moderate variety of trace variants and activities. Both for variants and activities some are very frequent while others are quite infrequent. `Teleclaims` is an established artificial log useful for testing discovery of various control-flow structures. With `Orders` we can demonstrate the algorithm's ability to discover complex control flow structures, as well as the option to abstract from rare behavior.

For each event log we perform 6300 runs of the algorithm with varying combinations of the different parameters, as specified in Table 2. Note that we keep the order of place candidate traversal fixed for all runs.

To investigate the qualitative impact of the proposed heuristics, we need to fix the order of candidate evaluation to prevent effects due to different evaluation orders. For easy reproducibility, we use a lexicographical ordering based on activity names. The purpose is to focus on the effect of the different parameters, and possibly derive which of them are the most relevant for the discovery of certain models and whether certain (combinations of) settings are preferable.

Our experiments with combined fitness have shown that only for the `Sepsis` log there are place candidates that score lower on relative fitness than aggregate fitness. However, this happened for very few parameter combinations and then only for at most 2 candidate places. Thus, for the logs evaluated, we can conclude that the choice between using $\texttt{fm}_{agg}$ and $\texttt{fm}_{comb}$ does not have a significant impact.

Evaluation Metrics

Several approaches exist to measure model quality with respect to an event log. In the following, we give a brief overview of the techniques applied in the context of this evaluation.

We use alignment-based fitness to measure how well the behavior in the log is represented by the model. Alignments take an event log and compute the minimal number of insertions and deletions needed to make the traces fitting with respect to a given model, then normalize this value using the worst-case edit distance. Consider the example Petri net in Figure 16 and the trace $\langle \blacktriangleright, a, b, c, \blacksquare \rangle$. This trace can be aligned to the Petri net by, for example, removing b and c or by removing a. Removing a needs 1 edit operation, which is the optimum in this case. The worst-case edit distance would require us to remove every activity from the trace (5) and insert an activity for every transition that needs to be fired to obtain the shortest path through the model (3). This results in an alignment-based fitness of $1 - \frac{1}{5+3}$ for the example trace. For details, we refer the reader to [32].

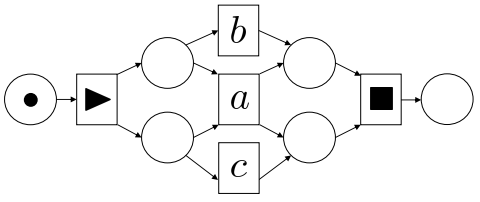


Figure 16. An example model used to illustrate the applied quality metrics.

Evaluating whether the model is sufficiently restrictive, i.e., does not allow for (much) behavior outside of the process behavior is more challenging. The event log represents a set of example process executions and cannot be expected to be complete, making it hard to reliably evaluate process-related precision. Another problem is the potentially infinite language of the model which can make a straightforward comparison infeasible. Therefore, metrics that approximate precision are commonly used. We choose a precision metric based on escaping edges. This metric compares the number of activities enabled in the model with the number of activities actually executed during the replay of the event log. Consider the example Petri net in Figure 16 and the traces $\langle \blacktriangleright, a, \blacksquare \rangle$ and $\langle \blacktriangleright, c, b, \blacksquare \rangle$. We replay all prefixes of traces in the event log and take note of the frequency of that prefix, the number of enabled transitions at the end of the prefix and the number of transitions fired after the prefix. In our example, for both traces in the beginning only $\blacktriangleright$ is enabled and consequently fired ($2 \cdot 1$ enabled, $2 \cdot 1$ fired). This enables three activities a, b and c, of which we fire a in the first trace and c in the second trace ($2 \cdot 3$ enabled, $2 \cdot 2$ executed). Now, for the first trace $\blacksquare$ is the only one executed and fired ($1 \cdot 1$ enabled, $1 \cdot 1$ fired). For the second trace, after firing c only b is enabled and fired ($1 \cdot 1$ enabled, $1 \cdot 1$ fired) followed by the final activity, $\blacksquare$, being enabled and fired ($1 \cdot 1$ enabled, $1 \cdot 1$ fired). This results in a precision of $\frac{2 \cdot 1 + 2 \cdot 2 + 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1}{2 \cdot 1 + 2 \cdot 3 + 1 \cdot 1 + 1 \cdot 1 + 1 \cdot 1} = \frac{9}{11}$, i.e., the number of transitions actually fired during replay divided by the number of enabled transitions. For details on escaping edges precision we refer the reader to [33].

Recall, that for most real-life logs not all quality aspects can be perfectly satisfied at the same time (e.g., high fitness often entails low precision and the other way around). They should be balanced according to the user's needs, i.e., the choice of the best model depends on its purpose. In our evaluation, we consider two target audiences. Some users prefer to directly apply a discovery algorithm to the unmodified log, expecting the algorithm to filter infrequent activities automatically and return a model that focuses on the main behavior in the event log. In the absence of a clear use-case, it is common to score process models based on the harmonic mean of fitness and precision (F_1-Score), which ensures that the resulting aggregated quality score reflects both metrics, i.e., a low value in one of the metrics cannot be obscured by a high value in the other. Without further information, we assume that such a user to be interested in obtaining a model with a high F_1-Score.

Definition 7.1. (F_1-Score)
Given an event log $L \in \mathbb{M}(\mathcal{T})$ and a Petri net N with a fitness score of $\texttt{fitness}_L(N)$ and a precision score of $\texttt{precision}_L(N)$, we define the F_1-score as

$$F_1(L, N) = \begin{cases} 0, & \textit{if } \texttt{fitness}_L(N) = 0 \vee \texttt{precision}_L(N) = 0 \\ \dfrac{2}{\dfrac{1}{\texttt{fitness}_L(N)} + \dfrac{1}{\texttt{precision}_L(N)}}, & \textit{otherwise} \end{cases}$$

In general, users applying our algorithm have the option to perform some basic preprocessing beforehand. In particular, they may use available functionality [34] to remove infrequent activities and infrequent trace variants they are not interested in. In such cases, all infrequent activities and traces remaining in the event log can be considered to be of interest and should be reflected in the discovered model, while still abstracting from exceptional behavior patters. Without further information,

we assume that such a user is interested in obtaining a model that scores high in fitness and precision but ideally also contains all activities from the event log. Therefore, we introduce the the metric of *activity-coverage* and define the HM-Score as the harmonic mean of fitness, precision and activity-coverage.

Definition 7.2. (Activity-Coverage)
Let $L \in \mathbb{M}(\mathcal{T})$ be an event log over the set of activities A and let $N = (A', P)$ be a Petri net with $A' \subseteq A$, i.e., A' is the subset of log activities included in the Petri net N. We define *activity-coverage* of N with respect to L as

$$\texttt{activity-coverage}_L(N) = \frac{|A'|}{|A|}$$

A value of 1 indicates that all activities in the event log are also part of the Petri net, while a value of 0 indicates that no activity in the event log is part of the Petri net. With respect to the event log $L = [\langle \blacktriangleright, a, \blacksquare \rangle, \langle \blacktriangleright, c, b, \blacksquare \rangle, \langle \blacktriangleright, d, e, d, e, a, \blacksquare \rangle]$, the example Petri net in Figure 16 achieves an activity-coverage of $\frac{3}{5}$.

Definition 7.3. (HM-Score)
Given an event log $L \in \mathbb{M}(\mathcal{T})$ and a Petri net N with a fitness score of $\texttt{fitness}_L(N)$ and a precision score of $\texttt{precision}_L(N)$, we define the HM-score as

$$\texttt{HM}(L, N) = \begin{cases} 0, & \textit{if } \texttt{fitness}_L(N) = 0 \vee \texttt{precision}_L(N) = 0 \vee \texttt{activity-coverage} = 0 \\ \dfrac{3}{\dfrac{1}{\texttt{fitness}_L(N)} + \dfrac{1}{\texttt{precision}_L(N)} + \dfrac{1}{\texttt{activity-coverage}_L(N)}}, & \textit{otherwise} \end{cases}$$

The simplicity of a model is highly subjective and a variety of factors may contribute to the readability of a model. Not all of these can be easily represented by numbers, e.g., the way a Petri net is plotted. Even though several metrics have been suggested for it, they are restricted to only some aspects contributing to model understandability and no generally agreed upon solution has become the standard yet. Therefore, we forgo an extensive evaluation of simplicity and focus on a straightforward metric based on the average number of arcs per transition. Not only is this metric closely related to the strategy of the presented approach to prefer places that have few arcs, but also existing research confirms the general relevance of this aspect [35].

Definition 7.4. (Simplicity)
Given a Petri net $N = (A, P)$, we define simplicity as the average number of arcs per transitions, i.e.,

$$\texttt{simplicity}(N) = \frac{\sum_{(I|O) \in P} |I| + |O|}{|A|}.$$

The example Petri net in Figure 16 achieves a simplicity of $\frac{14}{5} \approx 2.8$. On the downside, this measure of simplicity does not map to the interval between 0 and 1 and is therefore not directly comparable

with the other quality metrics (and thus not included in the aggregated score). On the upside, it returns an objective value that allows for further subjective interpretation as the reader sees fit.

7.2. Qualitative analysis

Some interdependencies between the model quality aspects are to be expected and confirmed by our results. Removing a transition from a Petri net reduces the behavior of the net and therefore has a negative effect on fitness and a positive effect on precision. It is important to keep in mind the exact metrics used to measure fitness and precision to avoid misinterpretation of the results: while alignments are quite forgiving with respect to missing infrequent transitions (they simply assign a penalty whenever the corresponding activity occurs in the event log), escaping edges based precision is sensitive with respect to transitions that are frequently enabled without being fired (e.g. in the case of parallelism).

One of our major goals in this work is to be able to avoid the removal of transitions based on infrequency. By achieving this goal, we obtain models that contain transitions which are far more often enabled than fired. Such models score significantly worse with respect to precision than models without those infrequent transitions, while fitness remains comparable. Therefore, in addition to evaluating models using the HM-Score, we focus on models with perfect activity-coverage by evaluating them separately from the complete set of results. Finally, we include some representative models to support the interpretation of the number-based evaluation. However, one needs to keep in mind that the choice of the best model always depends on the user's needs and models scoring high in the context of this general evaluation do not necessarily represent the best model for every application.

Overview of Quality Results

In Figure 17, an overview of the quality results of the 6300 models generated for each log is given. Fitness and simplicity remain rather stable, with fitness being generally high and simplicity values clustering between 2 and 3 arcs per transition on average, which we consider a good value. On the other hand, precision and activity-coverage, and by extension the harmonic means HM and F_1, vary a lot for the discovered models. This clearly indicates that the choice of parameters has a strong impact on these quality aspects.

While we discovered only 5 unique models, i.e., models with different behavior, for Orders, there were 29 unique models found for RTFM, 34 for Teleclaims and 160 for Sepsis. The quality results and frequencies of a selected subset of the discovered Petri nets are given in Table 3. Additionally, we provide the same results for the models discovered by the Inductive Miner infrequent (IMf) with default settings as implemented in ProM [34] and the models discovered by the eST-Miner with $\tau = 1.0$ (comparable to region theory results). Our approach can discover models with HM and F_1 scores that clearly outperform IMf with default settings as well as eST-Miner with $\tau = 1.0$ on the two real-life event logs. For the two artificial event logs results are comparable. A detailed comparison of these models follows at the end of this section.

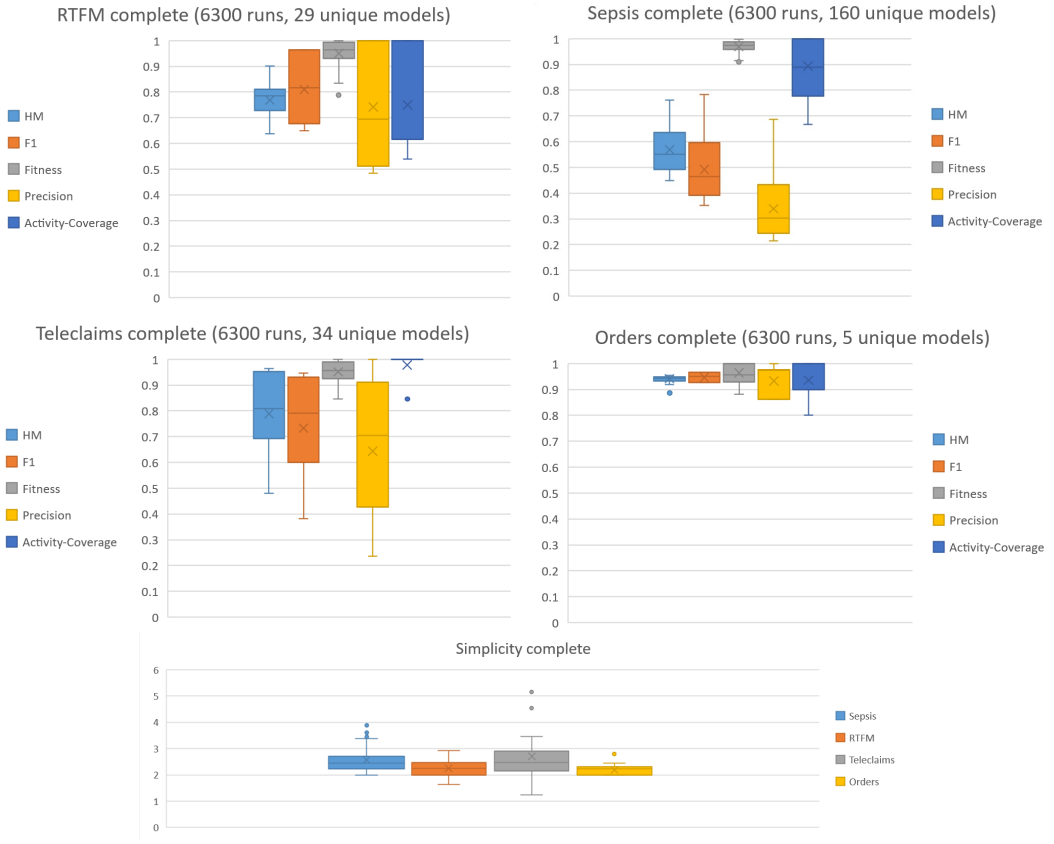


Figure 17. Overview of model quality results for all 6300 runs (complete) with varying parameters but fixed candidate traversal order.

Table 3. Overview of the qualitative results of selected models discovered during our experimentation. For each log we show the quality results of the model with the maximal HM value, the maximal F_1 value, and the model with the maximal HM out of the subset of models with perfect activity-coverage (aCov), as well as the frequency with which this model was discovered. Additionally, we provide scores for the Inductive Miner infrequent (IMf, default settings) and the eST-Miner with $\tau = 1.0$. Green background marks comparatively large values. All of these models are also visualized in Figures 20 to 23 at the end of the section.

	RTFM					Sepsis					Teleclaims					Orders				
	For Comparison		Selection of Discovered Models			For Comparison		Selection of Discovered Models			For Comparison		Selection of Discovered Models			For Comparison		Selection of Discovered Models		
Metric	Imf (default)	eST (τ=1.0)	Max. HM	Max. F1	HM (aCov)	Imf (default)	eST (τ=1.0)	Max. HM	Max. F1	HM (aCov)	Imf (default)	eST (τ=1.0)	Max. HM	Max. F1	HM (aCov)	Imf (default)	eST (τ=1.0)	Max. HM	Max. F1	HM (aCov)
HM	0.7300	0.7385	0.9018	0.8108	0.9018	0.4154	0.4212	0.7620	0.7620	0.5854	0.9659	0.6917	0.9640	0.9640	0.9640	0.9550	0.9536	0.9536	0.9432	0.9536
F1	0.6432	0.6531	0.8596	0.9638	0.8596	0.3215	0.3266	0.7836	0.7836	0.4849	0.9496	0.5993	0.9469	0.9469	0.9469	0.9340	0.9319	0.9319	0.9664	0.9319
Fitness	0.9820	1.0000	0.8747	0.9301	0.8747	0.9060	1.0000	0.9115	0.9115	0.9679	0.9490	1.0000	0.9244	0.9244	0.9244	0.9600	1.0000	1.0000	0.9562	1.0000
Precision	0.4782	0.4849	0.8451	1.0000	0.8451	0.1954	0.1954	0.6871	0.6871	0.3235	0.9503	0.4279	0.9706	0.9706	0.9706	0.9094	0.8725	0.8725	0.9768	0.8725
Activity-Coverage	1.0000	1.0000	0.8451	0.6154	0.8451	1.0000	1.0000	0.7222	0.7222	1.0000	1.0000	1.0000	0.9231	0.9231	0.9231	1.0000	1.0000	1.0000	1.0000	1.0000
Simplicity	2.3333	2.1538	2.7692	2.0000	2.7692	2.5833	2.1111	2.4615	2.4615	3.4444	2.1333	3.2308	2.0000	2.0000	2.0000	2.2000	2.9000	2.8000	2.2222	2.8000
Frequency	-	-	150	1590	150	-	-	72	72	180	-	-	504	504	504	-	-	44	2252	44

Discussion of Dead Transitions

Since we are interested in abstracting from infrequent behavioral patterns without outright removal of infrequent activities or complete trace variants, we take a closer look at the discovery of models that include all activities from the event log. During our experimentation, we discovered 2 different such models for `Orders`, 31 for `Teleclaims`, 29 for `Sepsis` and 9 for `RTFM`.

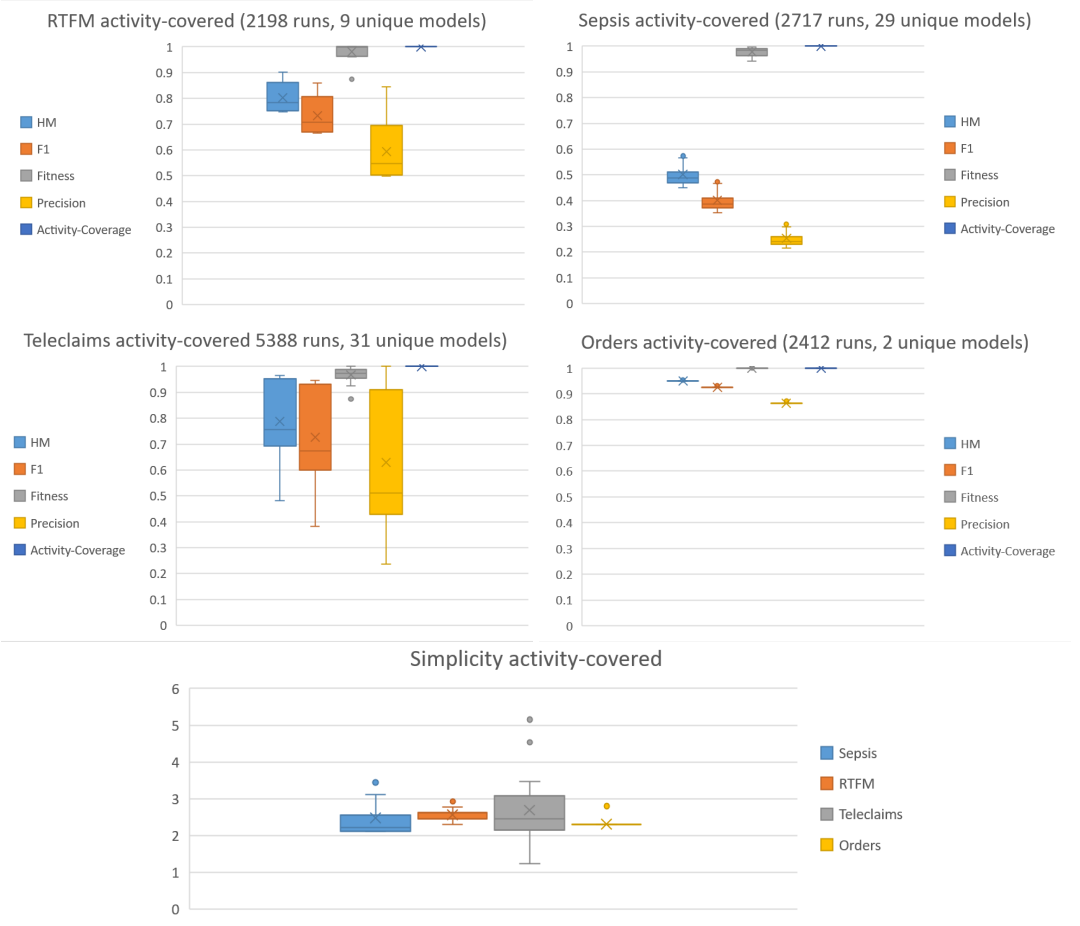


Figure 18. Overview of quality results for all models discovered in the 6300 runs that include all activities observed in the corresponding event log.

In Figure 18, we show an overview of the quality metrics restricted to the runs that resulted in models that include all activities. The general tendencies remain similar to the results shown for the models discovered for the complete set of runs. Fitness is consistently high, with less lower scoring outliers than we have seen for the complete set of models. This is to be expected, since these models do contain all log activities and our proposed approach guarantees that they can all be fired at least once. Consequently, precision is comparable for the artificial logs which do not include a lot of noise

or diverse behavior, or lower for the real-life logs, which exhibit a significantly higher variance in behavior. Notably, the number of models and variance in precision have decreased for `Sepsis` and `Orders` - apparently, the parameter combinations allowing for the discovery of models without dead transitions do result in models scoring similarly in quality.

Figure 19. Comparison of the number of dead transitions of the models discovered using *relative fitness* and using *combined fitness* with the proposed algorithm.

In Section 5, we introduced *combined fitness* with the goal of preventing the uncontrolled deletion of infrequent activities due to their execution being blocked accidentally. In Figure 19, we compare the impact of using combined fitness and of using relative fitness on the number of dead transitions in the discovered models. In particular for the two real-life event logs, `Sepsis` and `RTFM`, a clear effect is visible: when using combined fitness is it much more likely to discover models which include all or most of the observed activities. This observation is confirmed by the data in Table 4 where we compare the number of runs that result in models without dead transitions for the runs using combined

Table 4. Out of the 6300 runs we performed in our experiments, 3150 were performed using *combined fitness* and 3150 using *relative fitness*. For each event log this table gives an overview about the number of runs that resulted in models without dead transition, as well as how many different such models were discovered.

	combined fitness (3150 runs)		**relative fitness (3150 runs)**	
	#runs	**#unique models**	**#runs**	**#unique models**
`RTFM`	1946	8	252	1
`Sepsis`	2479	26	238	3
`Teleclaims`	2688	18	2700	13
`Orders`	1206	1	1206	1

fitness and the runs using relative fitness. While the trend is clearly visible for the two real-life event logs, for `Teleclaims` and `Orders` the majority of parameter combinations result in models without dead transitions, independently of the fitness metric chosen. Similar tendencies can be observed for the number of different models without dead transitions; especially for the two real-life logs, most such models are discovered using combined fitness.

Another interesting aspect can be observed for the `RTFM` and `Teleclaims` event logs: models with certain numbers of dead transitions ($1, 2, 4$ for `RTFM`, 1 for `Teleclaims`) are never discovered. This can be explained by groups of transitions which are closely coupled in behavior and (almost) always occur together in the same traces. They are removed from the model together once these traces are no longer replayable.

Discussion of Selected Models

In Figures 20, 21, 22 and 23 we present a selection of models for each log. For comparison, we present the models discovered by IMf (default settings) and the model discovered by the eST-Miner with $\tau = 1.0$. From the many models discovered during the experimentation with our approach, we show the models scoring highest with respect to `HM` (the harmonic mean of fitness, precision,

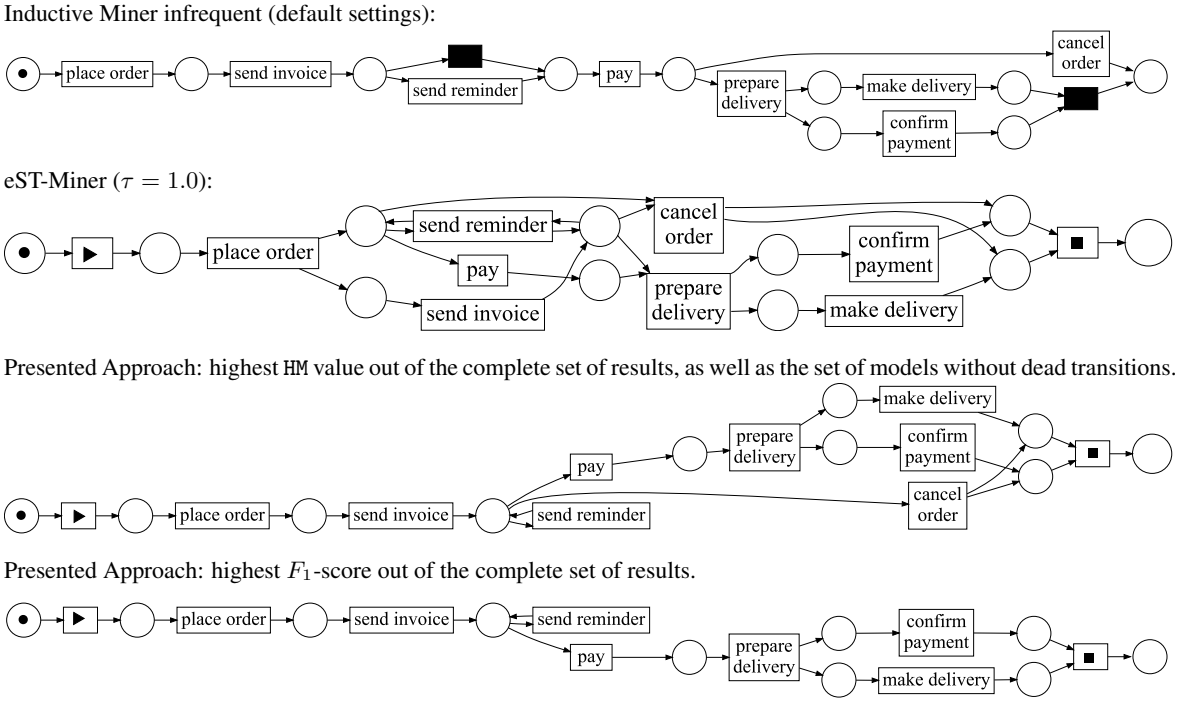
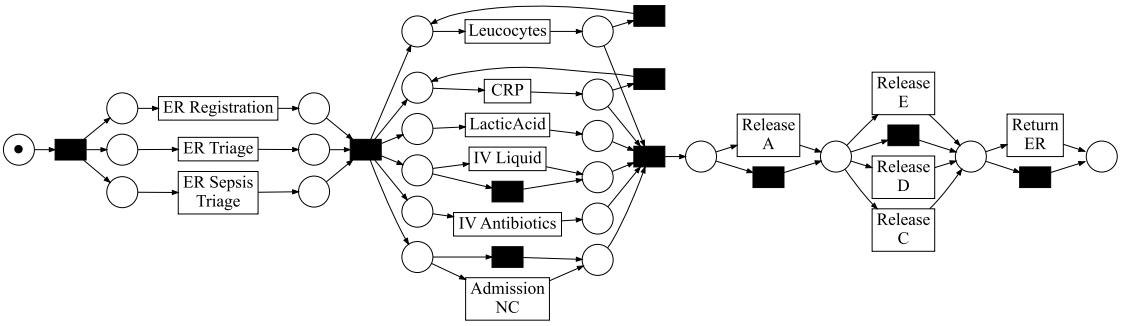
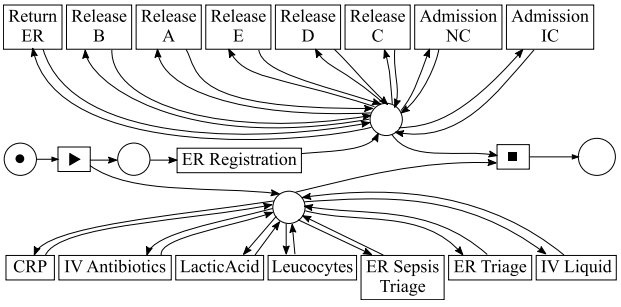


Figure 20. The Petri nets discovered based on the `Orders` log using the Inductive Miner infrequent (default settings), the eST-Miner with $\tau = 1.0$, and a subset of interesting models discovered using the presented approach.

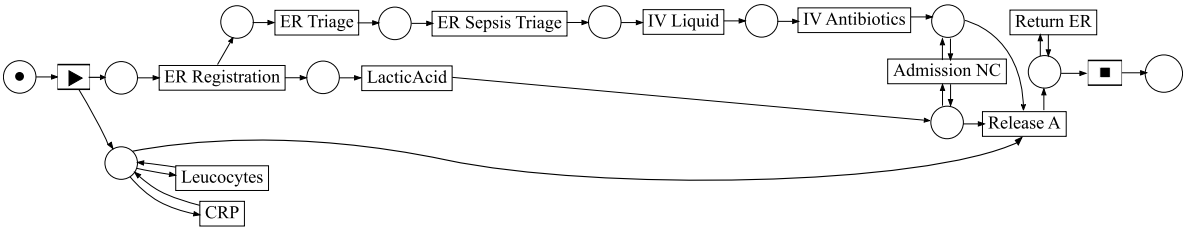
Inductive Miner infrequent (default settings):



eST-Miner ($\tau = 1.0$):



Presented Approach: highest HM value and highest F_1-score out of the complete set of results.



Presented Approach: highest HM value out of the set of models without dead transitions.

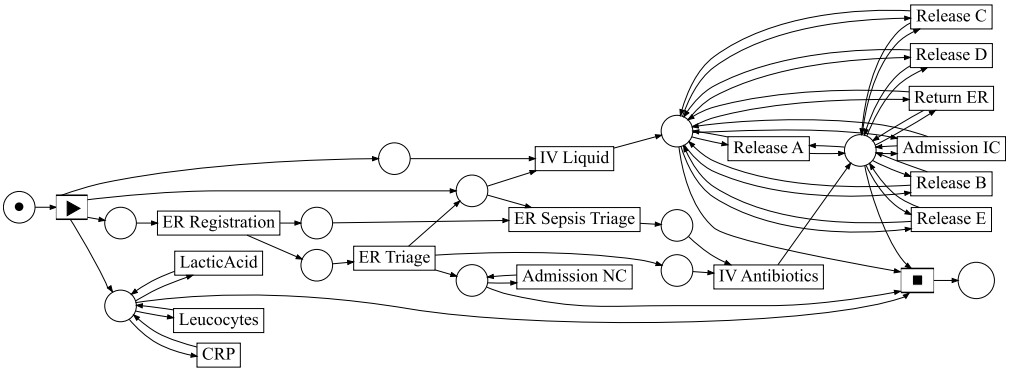


Figure 21. The Petri nets discovered based on the Sepsis log using the Inductive Miner infrequent (default settings), the eST-Miner with $\tau = 1.0$, and a subset of interesting models discovered using the presented approach.

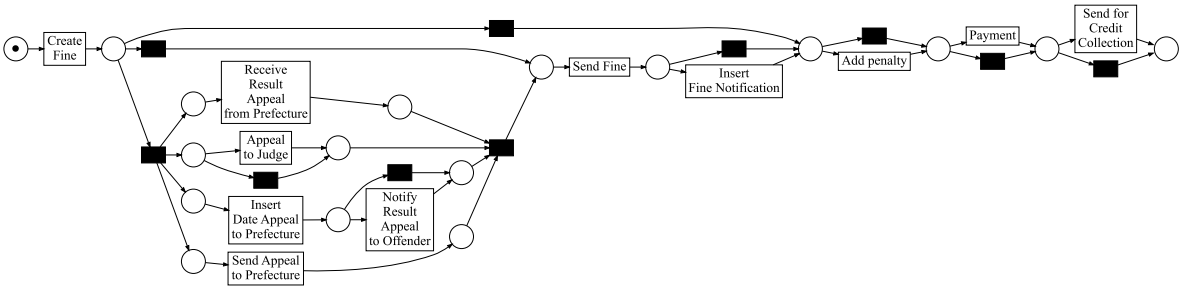
activity-coverage) as well as with respect to F_1 (harmonic mean of fitness and precision only). Furthermore, we present the models with the highest HM and F_1 values out of the set of models without dead transitions. Of course, while these models score highest with respect to the general quality metrics used in this evaluation, other models may still be considered to be more suitable for specific contexts or applications.

All models shown in Figure 20 were discovered for the Orders log. For this rather simple event log, all models achieve relatively high scores with respect to the quality metrics. However, some notable differences in the expressed behavior can be observed in particular with respect to the activities *send invoice, send reminder, pay* and *cancel order*. According to the event log, in most cases the execution of *send invoice* is eventually followed either by *pay* (and then delivery) or by *cancel order*, but never both. In rare cases, payment occurs before sending the invoice. After sending the invoice, reminders can be sent repeatedly, until payment is received or the order is canceled. This behavior is fully expressed only by the model discovered using the eST-Miner with $\tau = 1.0$, which is comparable to results produced by region-based approaches. Since payment before sending the invoice is rare, users may prefer the other models which focus on behavior where payment arrives after sending the invoice. The model discovered by IMf further deviates from the log by not allowing for repeated reminders (occurring in 25 % of the traces), and enabling the cancellation of orders after payment. The model with the highest HM value is the same for both, the complete set of models as well as the set without dead transitions. It includes all activities observed in the event log, in contrast to the model with the highest F_1-Score, which does not contain the activity *cancel order* (occurring in 13.03 % of traces) at all, resulting in slightly lower fitness but increased precision.

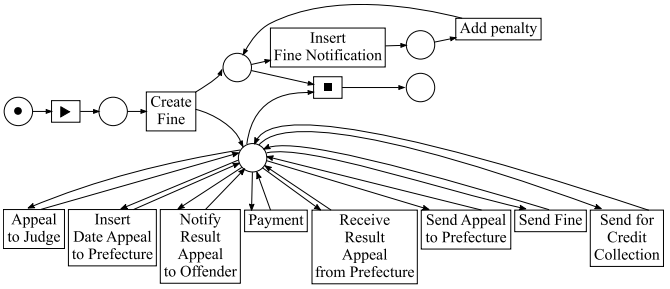
The Sepsis event log exhibits many repetitions of activities and a comparatively high control-flow variance, with 846 trace variants in 1050 traces, the most frequent of which occurs only 35 times. Thus, the discovery of a model with simultaneously high fitness and precision is challenging. Figure 21 presents a selection of discovered Petri nets. The IMf manages to discover groups of activities that occur in sequence, however, within these groups the activities are in parallel and mostly skippable, resulting in a very low precision. The eST-Miner with $\tau = 1.0$ illustrates a disadvantage of requiring perfect fitness: the resulting model allows for nearly all possible behaviors. For the model with highest HM value out of all models without dead transitions, this problem becomes less severe. Finally, the model with highest HM and F_1 values out of all discovered models manages to capture the main behavior hidden in the traces while ignoring infrequent activity behavior, achieving comparatively high precision at the cost of not representing all activities.

Figure 22 shows Petri nets discovered from the RTFM log. Considering the models discovered by IMf and eST-Miner with $\tau = 1.0$, we observe the same general tendencies as for the previous logs. For the model with the highest F_1-score discovered by our approach, we note that several activities are missing, meaning that they are not part of any replayable trace from the event log. The reason can be found by investigation of this particular event log, which describes two very distinct sub-processes, the more frequent of which consists of the activities still contained in the model. The activities of the infrequent sub-process related to appeals have been removed, allowing to focus on the main process. The model with the highest HM includes all activities from the event log. It includes the main process which is also expressed by the model with highest F_1, with additional self-loops that

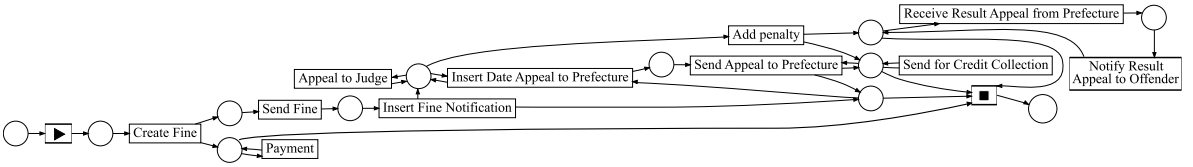
Inductive Miner infrequent (default settings):



eST-Miner ($\tau = 1.0$):



Presented Approach: highest HM value out of the complete set of results, as well as the set of models without dead transitions.



Presented Approach: highest F_1-score out of the complete set of results.

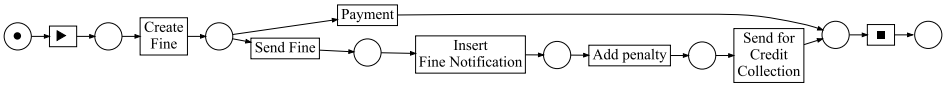
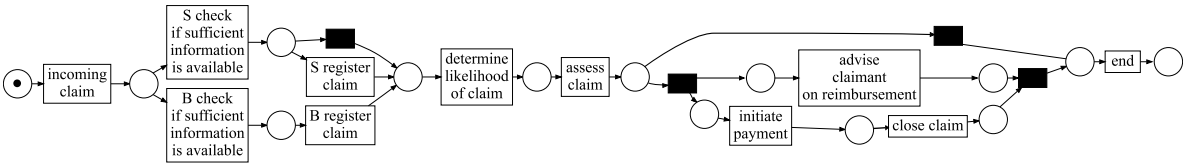


Figure 22. The Petri nets discovered based on the RTFM log using the Inductive Miner infrequent (default settings), the eST-Miner with $\tau = 1.0$, and a subset of interesting models discovered using the presented approach.

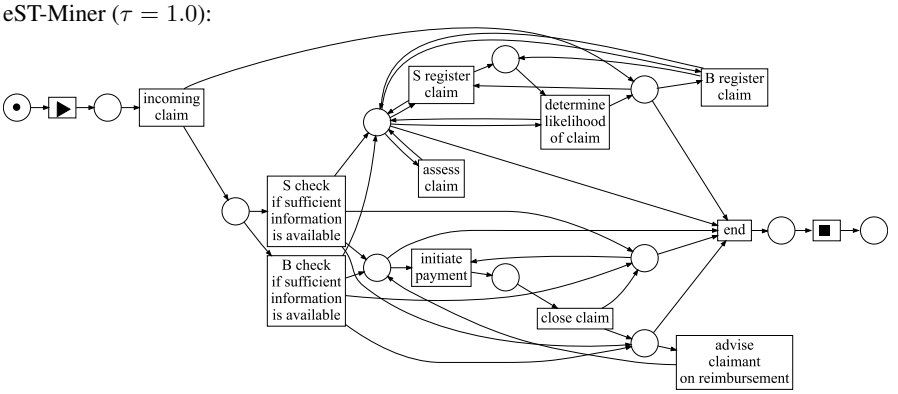
model optionality of those activities. Additionally, the control-flow of the appeal-related subprocess is included. Here, self-loops are not only used to model skippable activities but also to enforce a certain order on the events. Despite the limitations of using only uniquely labeled transitions, the positioning of the infrequent activities within the control-flow is precise and reflects their behavioral patterns in the event log.

A set of process models discovered from the Teleclaims log is presented in Figure 23. For this event log, the same model scores highest with respect to HM and F_1 for the complete set of discovered models as well as for the set of models without dead transitions. This model and the model discovered

Inductive Miner infrequent (default settings):



eST-Miner ($\tau = 1.0$):



Presented Approach: highest HM value and highest F_1-score out of the complete set of results, as well as the set of models without dead transitions.

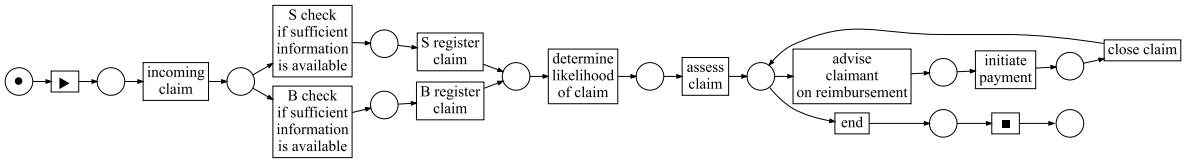


Figure 23. The Petri nets discovered based on the Teleclaims log using the Inductive Miner infrequent (default settings), the eST-Miner with $\tau = 1.0$, and and a subset of interesting models discovered using the presented approach.

by IMf express similar behavior, with the main difference being the representation of skippable activities: with all transitions being uniquely labeled, our approach has to rely on loop constructs rather than silent activities. The eST-Miner with $\tau = 1.0$ does not abstract from infrequent behavior, which in this case results in a perfectly fitting but quite complex model.

Our results confirm the expectation that even minor gains in fitness are usually accompanied by a major drop in precision. The models with the best F_1-Score are usually those with the highest precision value. From Figures 20 to 23 we can observe that these models seem to focus on the main process behavior, giving a clear representation of the control-flow of the main activities. However, they are likely not to incorporate infrequent activities. This is clearly illustrated by the Orders and RTFM event logs. Here, infrequent but potentially vital paths in the control-flow (e.g. cancellation of an order or appeal against a fine) are revealed when the discovery algorithm abstracts from deviating behavioral patterns without ignoring infrequent activities themselves. Our results show that the presented approach is able to return models anywhere on the scale balancing fitness, precision and activity-coverage, based on the choice of parameters.

7.3. Impact of parameter choices

While the quality results discussed earlier clearly indicate that our approach is able to discover models balancing fitness, precision and activity-coverage while maintaining reasonable simplicity, the choice of parameters has a significant impact. Therefore, we investigate this further. We used decision tree analysis to search for parameter settings that would result in the highest quality models as indicated by the HM-Score. Since, for some event logs, the best model according to this metric was discovered very infrequently, we extended the notion to the top 5% of runs. However, in other logs the best model was discovered in significantly more than 5% of the runs. This discrepancy resulted in an imbalance of the absolute number of runs resulting in the highest-scoring model(s). Additionally, we performed the same analysis approach for the set of models without dead activities, again looking for the (at least) 5% of runs which resulted in models with the highest HM value.

The results of this analysis are shown in Figure 24, where each line represents a set of parameter combinations that leads to the discovery of the best model(s). Information on the frequency and entropy of the corresponding leaf in the decision tree is included as well.

For Sepsis the overall best scoring model requires the lowest investigated value of τ, that is 0.3. Additionally, δ should either be ignored completely by using the $\texttt{adapt}_{noDelta}$ adaption function or set to the largest value. Slightly lower values of δ paired with the less constraining relative fitness can be combined with $\texttt{adapt}_{constant}$, i.e., always using the maximum value for δ. All in all, the parameters indicate that rather restricting places with comparatively low fitness must be accepted to discover models with higher quality for the Sepsis log. This makes sense in the context of the large variety of traces without clear main behavior, i.e., all trace variants have similarly low frequency. However, the large HM value of the models discovered with those parameter settings is mostly based on scoring high precision, which is also due to 5 activities not being included. Choosing $\tau = 0.7$ or larger together with relative fitness balances fitness and precision such that the highest-scoring model that includes all log activities can be discovered.

For the RTFM event log the use of combined fitness is a prerequisite to discovering the best-scoring models. In contrast to Sepsis, larger values of τ seem important: a choice $\tau = 0.5$ or $\tau = 0.8$ while ignoring δ ($\texttt{adapt}_{noDelta}$) results in a high-quality model. Alternatively, $\tau = 0.8$ can be combined with large values of δ. In this case, the highest scoring model in general and with full activity-coverage coincide.

For the Teleclaims event log the set of best-scoring models coincides with the set of best-scoring models without dead transitions. A τ value of at most 0.4 is a prerequisite. Combining a $\delta \geq 0.15$ with the constant adaption function results in a high-quality model. Gradually adapting δ using sigmoid adaption requires larger values of δ and high steepness s.

The set of best-scoring models in general and best-scoring models with perfect activity-coverage is the same for the Orders event log. To discover such models, one can combined combined fitness either with the sigmoid adaption function or with any other adaption function and low values for τ.

For the four event logs investigated in this paper, the most important parameter seems to be τ. This is not surprising, since τ has a direct impact on which places are available for addition to the Petri net. Furthermore, the combined fitness metric plays an important role in finding the best models for all

Best 5% of		Leaf Entropy	Frequency	τ	Adaption Strategy	δ	fm	s
Sepsis	Complete (514)	0	300	0.3	$\texttt{adapt}_{noDelta}$			
		0.469	20		$\texttt{adapt}_{constant}$ $\texttt{adapt}_{sigmoid}$	0.25		
		0	60		$\texttt{adapt}_{constant}$	[0.15, 0.2]	$\texttt{fm}_{rel}$	
	Perfect activity-coverage (180)	0	180	[0.7,0,9]			$\texttt{fm}_{rel}$	
RTFM	Complete (435)	0	150	0.5	$\texttt{adapt}_{noDelta}$		$\texttt{fm}_{comb}$	
		0	150	0.8	$\texttt{adapt}_{noDelta}$		$\texttt{fm}_{comb}$	
		0.811	180	0.8	$\texttt{adapt}_{constant}$ $\texttt{adapt}_{sigmoid}$	[0.15,0.25]	$\texttt{fm}_{comb}$	
	Perfect activity-coverage (435)	0	150	0.5	$\texttt{adapt}_{noDelta}$		$\texttt{fm}_{comb}$	
		0	150	0.8	$\texttt{adapt}_{noDelta}$		$\texttt{fm}_{comb}$	
		0.811	180	0.8	$\texttt{adapt}_{constant}$ $\texttt{adapt}_{sigmoid}$	[0.15,0.25]	$\texttt{fm}_{comb}$	
Teleclaims	Complete (504)	0	360	[0.3,0.4]	$\texttt{adapt}_{constant}$	[0.15,0.25]		
		0	96		$\texttt{adapt}_{sigmoid}$	[0.2,0.25]		[4,5]
	Perfect activity-coverage (504)	0	360	[0.3,0.4]	$\texttt{adapt}_{constant}$	[0.15,0.25]		
		0	96		$\texttt{adapt}_{sigmoid}$	[0.2,0.25]		[4,5]
Orders	Complete (2412)	0	1092		$\texttt{adapt}_{sigmoid}$		$\texttt{fm}_{comb}$	
		0	356	[0.3,0.4]	$\texttt{adapt}_{constant}$ $\texttt{adapt}_{noDelta}$		$\texttt{fm}_{comb}$	
	Perfect activity-coverage (2412)	0	1092		$\texttt{adapt}_{sigmoid}$		$\texttt{fm}_{comb}$	
		0	356	[0.3,0.4]	$\texttt{adapt}_{constant}$ $\texttt{adapt}_{noDelta}$		$\texttt{fm}_{comb}$	

Figure 24. Overview of the parameter choices resulting in the discovery of the models with the top 0.05 fraction of the HM value, once for all discovered models and once for the models with perfect activity-coverage. For each log, we indicate how often such models have been discovered in our experimentation. Each line refers to a set of parameter combinations, with the frequency and entropy of the corresponding leaf node in the decision tree. For each parameter that our decision tree analysis has revealed to be impactful, the possible values are indicated. Empty cells indicate that the corresponding parameter was insignificant for reaching the leaf. Leaves which did not include a significant number of positive instances are not included, resulting in a slight discrepancy between the number of such models discovered and the sum of leaf frequencies.

event logs but `Sepsis`. The constant and sigmoid adaption functions usually appear in combination with certain choices for δ, which makes sense since δ is limiting the range of the adaption strategies, which include the use of s. Generally, higher values for delta seem to be favored to find good models.

Notably, the artificial tree depth d^+ as well as $Q^{\leq}$ seem to have had no major impact on the discovery of any of the examined models (and therefore do not appear in Figure 24). For the potential places queue $Q^{\leq}$ this indicates that either the length limit was not reached or no important places were dropped. For d^+ we can conclude that in the scope of our experiments it did not result in significant complex places being added.

Some dependencies on log features are expected, and seem to be confirmed by the results in Figure 24. For the `RTFM` log, which has a few very dominant trace variants, we seem to generally achieve good results for rather high values of τ. In contrast, for the `Sepsis` log, which has a high variety of traces, a low τ-value seems mandatory to achieve high scores. Most likely, the large variety of fitting places allows for obtaining high precision, while our heuristics seems to successfully ensure the focus on the main behavioral patterns.

Indeed, the results from the `Sepsis` log, seem to confirm our algorithms ability to discover the main behavior hidden in an event log even in the absence of clear main trace variants: for a low value of τ, e.g. $\tau = 0.3$, the fraction of log traces replayable by the returned Petri net is close to 0.3, however, the alignment-based fitness reliably remains above 0.9, indicating that most of the traces are close to being replayable. We can conclude that the returned model successfully expresses the core behavior of the process.

To summarize, the results clearly show that high-quality models balancing the different quality aspects can be discovered. There is a significant variance in some of the metrics, particularly precision, indicating that the settings of the algorithm have a notable impact. Our preliminary investigation shows that, based on the event log, certain parameter choices are likely to result in high-quality models. Choosing combined fitness significantly increases the likelihood of the discovered model to contain all or most of the log activities. Our experimentation and analysis give a first indication about which parameters have a more notable impact and whether certain settings are more suitable for logs with certain properties. However, we investigated only four event logs and clearly further experimentation needs to be performed to explore to which degree a generalization of our results is possible. Note that the impact of the candidate traversal order has not been investigated yet, and may allow for further improvements.

7.4. Running time analysis

In general, the worst-case running time of the eST-Miner is exponential in the number of log activities A, since $\mathcal{O}((2^{|A|})^2)$ candidate places may have to be evaluated (compare [21]). However, limiting the tree depth to a *fixed* value $2 \leq k \leq |A|$, as was done in the experiments performed in the context of this work, can improve the worst-case running time. In the complete candidate tree the depth of a place coincides with the number of activities connected to that place, i.e., for a place $(I|O)$ at depth k we have that $|I| + |O| = k$. The number of candidate places at depth k corresponds to the number of all possible subsets of A of size k times all possibilities to split them over the sets of ingoing and outgoing transitions (for simplicity, we omit the insignificant border case of empty sets). For a fixed

traversal depth of k, in the worst-case we visit all candidates on all levels from 2 to k. Thus, for a fixed traversal depth k the number of place candidates visited in the complete candidate tree is bounded by

$$\sum_{i=2}^{k} \left(\binom{|A|}{i} \cdot 2^i \right) \leq k \cdot \left(\binom{|A|}{k} \cdot 2^k \right)$$
$$\in \mathcal{O}(k) \cdot \mathcal{O}(|A|^k) \cdot \mathcal{O}(2^k) \subseteq \mathcal{O}(|A|^k) \quad (\textit{for fixed } k)$$

We conclude that for a fixed traversal depth of k the number of place candidates evaluated in the worst-case becomes polynomial in the number of activities.

In our experimentation, we chose a fixed tree traversal depth of 5 to make a large-scale experimentation more feasible. We tracked the running times of the various components of the proposed eST-Miner variant with the goal to verify that our newly introduced place selection subroutine does not significantly decrease the algorithm's performance. In Figure 25, a summary of the running times of the different subroutines is shown on the left. Clearly, the fitness evaluation, i.e., the replay of the event log on each evaluated place candidate, makes up the bulk of the algorithms running time. The time needed for removing implicit places varies a lot over the runs (low values of τ tend to result in more places being inserted and thus more time needed for implicit place removal) but remains insignificant in most cases. Most importantly, in the context of this work, the time spend on place selection is even less than the time spend on computing the place candidates (tree traversal), as can be seen on the right-hand side of Figure 25 (note that the scale is different).

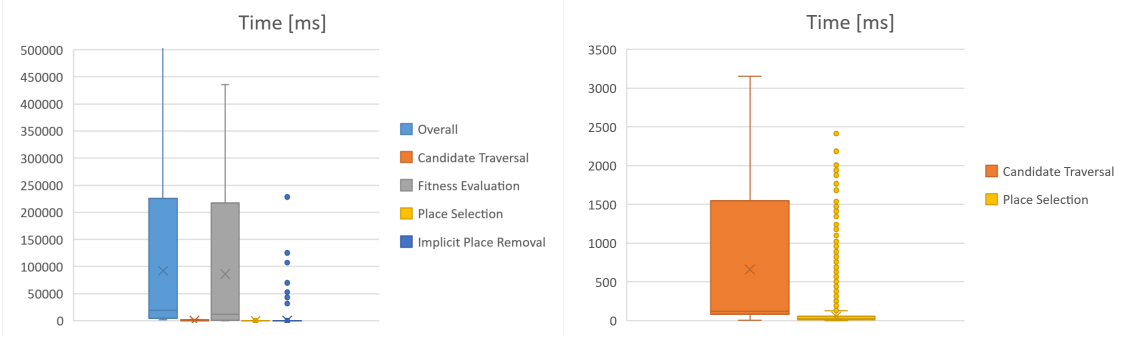


Figure 25. Overview of the running times achieved in our experiments on the left, with a smaller scale visualization of the shorter running times on the right. The newly added place selection subroutine does not add significantly to the overall running time of the algorithm.

8. Discussion and future work

The algorithm presented in this work is an extension of the eST-Miner. The goal is to provide fitness guarantees on the returned Petri net and to abstract from infrequent behavioral patterns without categorically removing infrequent activities while preserving the advantages of the algorithm, such as its ability to reliably discover complex control-flow structures (e.g. long-term dependencies) and guarantee of returning a maximal set of places. The latter are closely related to achieving high precision.

The algorithmic framework we proposed allows for a lot of flexibility in a variety of components and only a subset of the possible options has been explored in this work. On the one hand we can imagine several extensions and refinements of the strategies proposed so far. On the other hand, a strategy to reduce the load of decision making on the user, such as a recommender system or simplified interface, becomes more pressing the more complex the extensions to the algorithm grow.

In Section 5, we introduced a new fitness metric usable within the eST-Mining framework with the goal of avoiding the discovery of places that block infrequent activities. Aggregated fitness does not allow for any place to restrict an activity more than what is allowed according to the noise threshold, which guarantees that no activity can be blocked (dead) (if $\tau > 0$). Thus, activities get removed from the Petri net only if all the traces that contain them happen to be no longer replayable. This strategy has the intended positive effect, i.e., our evaluation clearly shows that we succeed in the sense that the models discovered using combined fitness are much more likely to contain all activities from the event log. However, the decision tree analysis shows that we do not necessarily find the best-scoring models using combined fitness, in particular with respect to precision. While inclusion of infrequent but well-defined behavior has been the goal, achieving it results not only in the representation of infrequent but interesting behavioral patterns by the returned process models, but also activities without a well-defined control-flow remain part of the discovered model. The eST-Miner framework connects these activities using the most constraining places allowed, which in the worst-case result in an activity that is enabled by ▶, may loop, and gets disabled by ■. An example for such structures can be seen in Figure 21 for the activities *LacticAcid, Leucocytes* and *CRP*. Obviously, such structures have a strong negative impact on precision, even though they seem reasonable in the context of the goals and constraints. Less constraining fitness metrics allow for more restrictive places to be added, which may remove such activities or connect them in a more restrictive way, resulting in higher precision values. Based on the assumption of our user being interested in all activities given in the event log, an investigation of a suitable adaption of aggregated fitness or a postprocessing step for more restrictive re-connection of such activities are promising future work. Fitness metrics diverging significantly from what has been proposed so far may be useful depending on the quality aspects a potential user is interested in. However, to utilize the performance optimization enabled by the eST-Miners complete candidate tree, they must satisfy the monotonicity properties discussed in Section 5. Straightforward relaxations of aggregated fitness such as the average, mean or harmonic mean of the individual fitness of a place's transitions do not satisfy this requirement.

Another interesting topic to investigate is the removal of activities that are no longer part of the replayable event log. In particular, when using combined fitness, this straightforward approach to guarantee deadlock-freeness of the returned model is unnecessarily strict: the removed activity may not be the reason for the trace being unfitting and may not be included in a deadlock at all. Future work includes the investigation of alternative approaches for deadlock detection and prevention that may keep such activities as part of the model.

We proposed and evaluated several delta adaption strategies. Unfortunately, no reliable conclusions can be drawn from the experiments performed so far. In any case, improvements or variations of the adaption strategies are likely possible. In fact, there is a multitude of options for delta adaption functions which do not necessarily need to be based on the parameters of place complexity, but may incorporate any information available at the place level. Examples include all kinds of relations be-

tween or constraints on the connected activities, token behavior and/or attributes of related activities or replayable traces. It would be particularly interesting to investigate to which degree the approach can be used to prioritize non-standard quality aspects, for example related to user interests such as compliance or performance.

There is room for improvement concerning the time performance of the algorithm as well. When an activity is determined to be no longer part of the replayable event log and is therefore removed, all subtrees in the complete candidate tree including this activity may be cut off without impacting the returned model. Furthermore, in this work, we apply the standard ILP-based approach to implicit place removal, which identifies implicit places based on the structure of the Petri net. Unfortunately, this approach becomes very time-consuming for larger sets of places. The replay-based approach introduced in [26] is much faster and can identify most implicit places. However, in the standard eST-Mining framework it does not verify whether all places required to make a currently evaluated place implicit are indeed present in the net, which is incompatible with the use of combined fitness in the framework proposed in this work. An adaption of this implicit place removal strategy may contribute to the overall performance of the proposed eST-Miner variant. Related to performance, we have shown that the number of candidate places becomes polynomial when limiting the depth of a tree. Tighter bounds on the degree of the polynomial would be of interest.

Finally, with a working solution to select subsets of fitting places such that high quality models without dead parts can be discovered, the remaining major limitation of the eST-Miner is its current inability to include silent or duplicate transition labels in the discovered Petri nets. With their added expressiveness, even better results with respect to fitness and, in particular, precision could be achieved.

9. Conclusion

In this paper, we proposed various extensions to the eST-Miner. We introduced a new fitness metric, aggregated fitness, investigated its properties and showed that it can be incorporated into the eST-Mining framework. The goal of this metric is to improve the eST-Miner's place evaluation to avoid the discovery of places that prevent infrequent activities from being executed categorically, while maintaining its ability to abstract from infrequent behavioral patterns. Furthermore, we propose a framework for place selection with the goal of guaranteeing that the discovered Petri net is free of deadlocks and satisfies a user-definable minimal fitness constraint. The approach employs heuristics to efficiently select a suitable subset of the discovered fitting places, while aiming towards high precision and simplicity. The algorithm is capable of discovering complex control-flow structures such as non-local dependencies. Furthermore, it is able to abstract from infrequent behavioral patters in the event log without simply filtering out infrequent activities or trace variants and to provide guarantees without over- or underfitting.

Our experiments, using four different event logs, clearly show that not only is it possible to discover high-quality models using the introduced approach, but also the heuristics applied have a significant impact on the obtained Petri net. Based on the parameter settings, models with a very different focus with respect to fitness, precision and the handling of infrequent behavior can be discovered.

Some parameters have a stronger effect than others and some parameter choices seem to be more suitable for logs with certain properties, which should be verified by further experimentation. A theoretical analysis of the running time, as well as an experimental overview of the time needed for the various subroutines of the algorithm, was presented, followed by a discussion of design decisions, open questions and future work.

Besides the aspects discussed in detail in Section 8, future work includes further experimentation to explore the generalization of the presented preliminary results, as well as the impact of the candidate place traversal order and its interaction with the heuristics used. The dead transitions removed from the model because they are no longer part of the replayable event log give rise to further possible extensions of the eST-Miner. When detected early on, they can be used to identify and cut off candidate subtrees consisting of dead places to improve the running time. Further investigation into the cause of their removal may lead to better noise handling strategies to improve the quality of discovered models. Finally, it would be interesting to investigate whether the presented place selection strategies can be adapted to improve other algorithms as well.

Acknowledgments:

We very much appreciate the time and effort of the reviewers, whose comments and suggestions contributed to improve the quality of this work.
Special thanks go to Tobias Brockhoff for his support with conducting the presented experiments.
The authors gratefully acknowledge the financial support by the Federal Ministry of Education and Research (BMBF) for the joint project Bridging AI (grant no. 16DHBKI023).
We thank the Alexander von Humboldt (AvH) Stiftung for supporting our research.

SPONSORED BY THE

Federal Ministry
of Education
and Research

References

[1] Reisig W. Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies. Springer Berlin Heidelberg, 2013. ISBN:9783642332784.

[2] Desel J, Oberweis A, Reisig W, Rozenberg G. Petri nets and business process management. *Saarbrücken: Geschäftsstelle Schloss Dagstuhl*, 1998.

[3] Desel J, Esparza J. Free-Choice Petri Nets. Cambridge Tracts in Theoretical Computer Science. Cambridge University Press, 1995. ISBN:9780521465199.

[4] Berthelot G. Transformations and Decompositions of Nets. In: Petri Nets: Central Models and Their Properties. Springer, Berlin, Heidelberg, 1987 pp. 359–376.

[5] van der Aalst W. Process Mining: Data Science in Action. Springer, Heidelberg, 2 edition, 2016. doi:10.1007/978-3-662-49851-4.

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

[7] Leemans S, Fahland D, van der Aalst W. Discovering Block-Structured Process Models from Event Logs - A Constructive Approach. *Application and Theory of Petri Nets and Concurrency*, 2013. Lecture Notes in Computer Science, vol 7927. doi:10.1007/978-3-642-38697-8_17.

[8] Weijters AJMM, Ribeiro JTS. Flexible Heuristics Miner (FHM). In: Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2011, part of the IEEE Symposium Series on Computational Intelligence 2011, April 11-15, 2011, Paris, France. IEEE, 2011 pp. 310–317. doi:10.1109/CIDM.2011.5949453.

[9] Badouel E, Bernardinello L, Darondeau P. Petri Net Synthesis. Text in Theoretical Computer Science, EATCS Series. Springer, 2015. doi:10.1007/978-3-662-47967-4.

[10] Lorenz R, Mauser S, Juhás G. How to Synthesize Nets from Languages: A Survey. In: Proceedings of the 39th Conference on Winter Simulation: 40 Years! The Best is Yet to Come, WSC '07. IEEE Press, Piscataway, NJ, USA, 2007 pp. 637–647. doi:10.1109/WSC.2007.4419657.

[11] Bergenthum R, Desel J, Lorenz R, Mauser S. Process mining based on regions of languages. *Proc. 5th Int. Conf. on Business Process Management*, 2007. pp. 375–383. doi:10.1007/978-3-540-75183-0_27.

[12] van der Werf JM, van Dongen B, Hurkens C, Serebrenik A. Process Discovery Using Integer Linear Programming. In: Applications and Theory of Petri Nets. Springer, Berlin, Heidelberg, 2008. doi:10.1007/978-3-540-68746-7_24.

[13] van Zelst S, van Dongen B, van der Aalst W. Avoiding Over-Fitting in ILP-Based Process Discovery. In: Business Process Management. Springer International Publishing, Cham, 2015 pp. 163–171. doi:10.1007/978-3-319-23063-4_10.

[14] van Zelst S, van Dongen B, van der Aalst W. ILP-Based Process Discovery Using Hybrid Regions. In: ATAED@Petri Nets/ACSD. 2015.

[15] Carmona J, Cortadella J, Kishinevsky M. A region-based algorithm for discovering Petri nets from event logs. In: Business Process Management. Springer, 2008 p. 358–373.

[16] Darondeau P. Deriving unbounded Petri nets from formal languages. In: CONCUR'98 Concurrency Theory. Springer, Berlin, Heidelberg. 1998 pp. 533–548. ISBN:978-3-540-68455-8.

[17] Bergenthum R, Desel J, Lorenz R, Mauser S. Synthesis of Petri Nets from Finite Partial Languages. *Fundam. Informaticae*, 2008. **88**(4):437–468.

[18] Ehrenfeucht A, Rozenberg G. Partial (set) 2-structures. *Acta Informatica*, 1990. **27**(4):343–368.

[19] Carmona J, Cortadella J, Kishinevsky M, Kondratyev A, Lavagno L, Yakovlev A. A Symbolic Algorithm for the Synthesis of Bounded Petri Nets. In: van Hee KM, Valk R (eds.), Applications and Theory of Petri Nets. Springer Berlin Heidelberg, Berlin, Heidelberg. 2008 pp. 92–111. ISBN:978-3-540-68746-7.

[20] Kalenkova A, Carmona J, Polyvyanyy A, La Rosa M. Automated Repair of Process Models Using Non-Local Constraints. In: Application and Theory of Petri Nets and Concurrency. Springer-Verlag, Berlin, Heidelberg. 2020 p. 280–300. ISBN:978-3-030-51830-1.

[21] Mannel LL, van der Aalst WMP. Finding Complex Process-Structures by Exploiting the Token-Game. In: Donatelli S, Haar S (eds.), Application and Theory of Petri Nets and Concurrency - 40th International Conference, PETRI NETS 2019, Aachen, Germany, June 23-28, 2019, Proceedings, volume 11522 of *Lecture Notes in Computer Science*. Springer, 2019 pp. 258–278. doi:10.1007/978-3-030-21571-2_15.

[22] Mannel LL, van der Aalst WMP. Discovering Process Models with Long-Term Dependencies While Providing Guarantees and Handling Infrequent Behavior. In: Bernardinello L, Petrucci L (eds.), Application and Theory of Petri Nets and Concurrency - 43rd International Conference, PETRI NETS 2022, Bergen, Norway, June 19-24, 2022, Proceedings, volume 13288 of *Lecture Notes in Computer Science*. Springer, 2022 pp. 303–324. doi:10.1007/978-3-031-06653-5_16.

[23] Garcia-Valles F, Colom J. Implicit places in net systems. *Proceedings 8th International Workshop on Petri Nets and Performance Models*, 1999. pp. 104–113. doi:10.1109/PNPM.1999.796557.

[24] Berthomieu B, Botlan DL, Dal-Zilio S. Petri Net Reductions for Counting Markings. In: Gallardo M, Merino P (eds.), Model Checking Software - 25th International Symposium, SPIN 2018, Malaga, Spain, June 20-22, 2018, Proceedings, volume 10869 of *LNCS*. Springer, 2018 pp. 65–84.

[25] Colom J, Silva M. Improving the linearly based characterization of P/T nets. In: Advances in Petri Nets 1990. Springer, Berlin, Heidelberg, 1991 pp. 113–145. doi:10.1007/3-540-53863-1_23.

[26] Mannel LL, Bergenthum R, van der Aalst WMP. Removing Implicit Places Using Regions for Process Discovery. In: Proceedings of the International Workshop on Algorithms & Theories for the Analysis of Event Data (ATAED) 2020, volume 2625. CEUR-WS.org pp. 20–32.

[27] van der Aalst WMP. Discovering the "Glue" Connecting Activities - Exploiting Monotonicity to Learn Places Faster. In: It's All About Coordination - Essays to Celebrate the Lifelong Scientific Achievements of Farhad Arbab. 2018 pp. 1–20. doi:10.1007/978-3-319-90089-6_1.

[28] Carmona J, van Dongen B, Solti A, Weidlich M. Conformance Checking - Relating Processes and Models. Springer, Cham, 2018. doi:10.1007/978-3-319-99414-7.

[29] Mannhardt, F. Sepsis Cases - Event Log, 2016. doi:10.4121/UUID:915D2BFB-7E84-49AD-A286-DC35F063A460.

[30] De Leoni, M, Mannhardt, F. Road Traffic Fine Management Process, 2015.

[31] van der Aalst WMP. Spreadsheets for BPM. *Business Process Management Journal*, 2010. **24**:105–127.

[32] Adriansyah A. Aligning observed and modeled behavior. Ph.D. thesis, Mathematics and Computer Science, 2014. doi:10.6100/IR770080.

[33] Munoz-Gama J, Carmona J. A Fresh Look at Precision in Process Conformance. In: BPM, volume 6336. 2010 pp. 211–226. ISBN:978-3-642-15617-5.

[34] van Dongen B, de Medeiros A, Verbeek H, Weijters A, van der Aalst W. The ProM Framework: A New Era in Process Mining Tool Support. In: Applications and Theory of Petri Nets 2005. Springer, Berlin, Heidelberg, 2005 pp. 444–454. doi:10.1007/11494744_25.

[35] Mendling J, Reijers HA, Cardoso J. What Makes Process Models Understandable? In: Alonso G, Dadam P, Rosemann M (eds.), Business Process Management. Springer Berlin Heidelberg, 2007 pp. 48–63. doi:10.1007/978-3-540-75183-0_4.

[36] Tax N, Sidorova N, van der Aalst WMP. Discovering more precise process models from event logs by filtering out chaotic activities. *J. Intell. Inf. Syst.*, 2019. **52**(1):107–139. doi:10.1007/s10844-018-0507-6.