Discovering Process Models With Long-Term Dependencies While Providing Guarantees and Handling Infrequent Behavior

Lisa L. Mannel✉ and Wil M. P. van der Aalst

Process and Data Science (PADS), RWTH Aachen University
{mannel, 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. In this paper, we present an extension of the eST-Miner process discovery algorithm. This approach computes a set of places which are considered to be fitting with respect to a user-definable fraction of the behavior described by the given event log, by evaluating all possible candidate places using token-based replay. The set of replayable traces is determined for each place in isolation, i.e., they do not need to be consistent. When combining these places into a Petri net by connecting them to the corresponding transitions, which are uniquely labeled for each activity in the event log, the resulting net can replay exactly those traces that can be replayed by each of the inserted places. Thus, inserting places without further checks may results in deadlocks and thus low fitness of the Petri net. In this paper, we explore a variant of the eST-Miner, that aims to select a subset of the discovered places such that the resulting Petri net guarantees a definable minimal fitness while maintaining high precision with respect to the input event log. Various place selection strategies are proposed and their impact on the returned Petri net is evaluated by experiments using both real and artificial event logs.

Keywords: Process Discovery, Petri Nets, eST-Miner

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 identification mapping the event to some execution instance (case id), a time stamp showing when the event was observed, and often extended meta-data of the activity or process instance. 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, for example a Petri net [1–4]. This 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

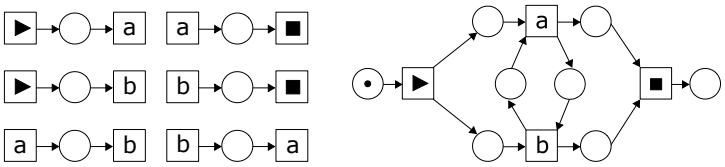


Fig. 1. Consider the event log $L = [\langle \blacktriangleright, a, b, \blacksquare \rangle^{40}, \langle \blacktriangleright, b, a, \blacksquare \rangle^{60}]$ and the set of candidate places on the left. Assuming, that we set the eST-Miner to accept all places that can replay at least $35\ \%$ of the event log, it would add all those places and return the Petri net on the right. Although each individual place has at least 40 fitting traces, the whole model cannot replay any trace.

is not desired to be reflected in the process model. Correctly classifying behavior as noise can be hard to impossible. An ideal process model can reproduce all behavior contained in an event log, while not allowing for unobserved behavior. It should represent all dependencies between events and at the same time be simple enough to be understandable by a human interpreter. Computation should be fast and robust to noise. Usually, it is impossible to fulfill all these requirements at the same time. Thus, different algorithms focus on different quality criteria, while neglecting others. As a result, the models returned for a given event log can differ significantly.

Many existing discovery algorithms abstract from the full information given in a log and/or generate places heuristically, in order to decrease computation time and complexity of the returned process models. While this is convenient in many applied settings, the resulting models are often underfitting, in particular when processes are complex. Examples are the Alpha Miner variants ([5]), the Inductive Mining family ([6]), genetic algorithms or Heuristic Miner. In contrast to these approaches, which are not able to (reliably) discover complex model structures, algorithms based on region theory [7–17]) 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 [18], which aims to circumvent performance issues by limiting the application of region theory to small fragments of a pre-descovered Petri net.

In [19] we introduced the discovery algorithm eST-Miner. This approach aims to combine the capability of finding complex control-flow structures like longterm-dependencies with an inherent ability to handle low-frequent behavior while exploiting the token-game to increase efficiency. The basic idea is to evaluate all possible places, defined by all possible combinations of uniquely labeled transitions, to discover a set of fitting ones. Efficiency is significantly increased by skipping uninteresting parts of the search space. This may decrease computation time immensely compared to the brute-force approach evaluating every single candidate place, while still providing guarantees with regard to fitness and precision.

While traditional region-theory uses a global perspective to find a set of feasible places, the eST-Miner evaluates each place separately, that is from a local perspective. This allows us to easily enforce all kinds of constraints definable on the place level, e.g., constraints on the number or type of connected transitions, token throughput or similar.

In particular, we are able to filter infrequent behavior locally, by requiring each place to be able to replay only a certain fraction of the traces in the event log. A candidate place will be accepted, if the event log contains sufficient support for the relation between the activities as defined by the place. In contrast to common noise filtering techniques which loose information by removing infrequent trace variants or infrequent activities from the event log, this approach can also consider infrequent information to discover relations between activities.

The local perspective of the eST-Miner ensures that all occurrences of activities within a log can contribute to the discovered model. However, when a set of discovered fitting places is combined into a Petri net, this Petri net allows only for the behavior in the intersection of the behaviors allowed by all inserted places. Thus, the Petri net may include 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 Fig. 1. Assuming we decide to add places that replay only a fraction of 0.35 of the traces, the Petri net discovered for the given event log cannot fire any transition after the start transition.

In this paper, we aim to remedy this issue by selecting a subset of the discovered places which can be combined into a Petri net with definable minimal fitness, while simultaneously striving for high precision and simplicity, without loosing the desirable properties of the eST-Miner. Thus, we require the algorithm to maintain its ability to discover and model non-local dependencies, to deal with infrequent behavior and to provide guarantees without over- or underfitting. Additionally, the time and space consumption should remain reasonable, in particular more scalable than classic region theory approaches.

Sec. 2 provides basic notation and definitions. In Sec. 3, we briefly review the basics of the standard eST-Miner. Our new concepts are introduced in Sections 4 and 5, and their experimental evaluation is presented in Sec. 6. Finally, Sec. 7 concludes this work by summarizing our findings and suggesting possibilities for future work.

2 Basic Notations, Event Logs, and Process Models

A set, e.g. $\{a, b, c\}$, does not contain any element more than once, while a multiset, e.g. $[a, a, b, a] = [a^3, b]$, may contain multiples of the same element. 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, $[x, y^2, z] \uplus [y^5, z^2] = [y^2, z]$. 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 this set. 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., $\langle a, b, a, b\rangle \neq \langle a, a, b, b\rangle$. The size of a set, multiset or sequence X, that is $|X|$, is defined to be the number of elements in X.

We define activities, traces, and logs as usual, except that we require each trace to begin with a designated start activity ($\blacktriangleright$) and end with a designated end activity ($\blacksquare$). Note that this is a reasonable assumption in the context of processes, and that any log can easily be transformed accordingly.

Definition 1 (Activity, Trace, 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 $\blacksquare \in \mathcal{A}$ be a designated end activity. A* trace *is a sequence containing $\blacktriangleright$ as the first element, $\blacksquare$ as the last element and in-between elements of $\mathcal{A} \setminus \{\blacktriangleright, \blacksquare\}$. Let $\mathcal{T}$ be the set of all such traces. A* log *$L \in \mathbb{M}(\mathcal{T})$ is a multiset of traces.*

In this paper, we use an alternative definition for Petri nets. We only allow for places connecting transitions, called here activities, that are initially empty (without tokens), because we allow only for traces starting with $\blacktriangleright$ and ending with $\blacksquare$. These places are uniquely identified by the non-empty sets of input activities I and output activities O. Each activity corresponds to exactly one uniquely labeled transition, therefore, this paper refers to transitions as activities.

Definition 2 (Petri nets). *A Petri net is a pair $N = (A, P)$, where $A \subseteq \mathcal{A}$ is the set of activities including start and end ($\{\blacktriangleright, \blacksquare\} \subseteq A$) and $P \subseteq \{(I|O) \mid I \subseteq A \wedge I \neq \emptyset \wedge O \subseteq A \wedge O \neq \emptyset\}$ is the set of places. We call I the set of* ingoing *activities of a place and O the set of* outgoing *activities.*

Note that if $p = (I|O)$, then $\bullet p = I$ and $p\bullet = O$ using standard notation.

A place is *fitting* if it can replay (parts of) the event log without missing or remaining tokens. Otherwise, it is *unfitting*.

Definition 3 (Fitting and Unfitting Places, compare [20]). *Let $N = (A, P)$ be a Petri net, let $p = (I|O) \in P$ be a place, and let σ be a trace. With respect to the given trace σ, p is called*

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

We extend these notions to the whole log using the noise parameter: with respect to a log L and parameter $\tau \in [0, 1]$, p is called fitting, *denoted by $\square_L^\tau(p)$, if and only if $|\{\sigma \in L \mid \square_\sigma(p)\}|/|L| \geq \tau$, and* unfitting *otherwise.*

Definition 4 (Behavior of a Petri net). *We define the* behavior *of the Petri net (A, P) to be the set of all fitting traces, that is $\{\sigma \in \mathcal{T} \mid \forall p \in P \colon \square_\sigma(p)\}$.*

Note that we only allow for behaviors of the form $\langle \blacktriangleright, a_1, a_2, \ldots a_n, \blacksquare \rangle$ (Def. 1) such that places are empty at the end of the trace and never have a negative number of tokens.

We are often interested in the traces of the event log which are replayable by certain (sets of) places.

Definition 5 (Multisets of Fitting Traces). *For an event log L and a place p, the multiset of log traces replayable by p is*

$$\mathit{fit}(L, p) = [\sigma \in L \mid \square_\sigma(p)].$$

For an event log L and a Petri net $N = (A, P)$, the multiset of log traces replayable by N is the intersection of all log traces replayable by the places in P, i.e.,

$$\mathit{fit}(L, N) = [\sigma \in L \mid \forall p \in P \colon \square_\sigma(p)] = \bigoplus_{p \in P} \mathit{fit}(L, p).$$

3 Introducing the eST-Miner

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

As input, the algorithm takes a log L and a parameter $\tau \in [0, 1]$, and returns a Petri net as output. A place is considered *fitting*, if it allows to replay at least a fraction τ of traces in the event log. Inspired by language-based regions, the basic strategy of the approach is to begin with a Petri net whose transitions correspond exactly to the activities used in the given log. From the finite set of unmarked, intermediate places, the subset of all fitting places is computed and inserted. To facilitate further computations and human readability, *implicit* places are identified and removed [21–23]. A place is implicit if its removal does not increase the behavior of the Petri net. Implicit places can be detected based on the structure of the Petri net as proposed for the first eST-Miner variant [19], or by using the faster replay-based implicit place removal strategy introduced in [24]. The latter one is applied in the experimentation of this paper.

The algorithm uses token-based replay to evaluate the candidate places. To avoid replaying the log on the exponential number of candidates (i.e., all pairs of subsets of activities, $(2^{|A|}-1)^2$), it organizes the potential places as a set of trees, such that certain properties hold. When traversing the trees, these properties allow to cut off subtrees, and thus candidates, based on the replay result of their parent [19]. This greatly increases efficiency, while still guaranteeing that all fitting places are found.

An example of such a tree-structured candidate space is shown in Fig. 2. Note the incremental structure of the trees, i.e., the increase in distance from the roots corresponds to the increase of input (red edges) and output (blue edges) activities. However, the organization of candidates within the same depth and their connections to other candidates is not fixed, but defined by the order of ingoing activities ($>_i$) and outgoing activities ($>_o$).

Definition 6 (Complete Candidate Tree). *Let 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 $CT = (N, F)$ with $N = \{(I|O) \mid I \subseteq A \backslash \{\blacksquare\} \wedge O \subseteq A \backslash \{\blacktriangleright\} \wedge I \neq \emptyset \wedge O \neq \emptyset\}$. We have that $F = F_{red} \cup F_{blue}$, with*

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

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

The purpose of the tree structured candidate space is to enable skipping of sets of uninteresting candidates to improve time and space efficiency. The runtime of the eST-Miner strongly depends on the number of candidate places skipped during the search for fitting places.

When a candidate place is evaluated to be fitting, i.e., it can replay a fraction of τ traces in the event log, existing variants of the eST-Miner simply insert the place into

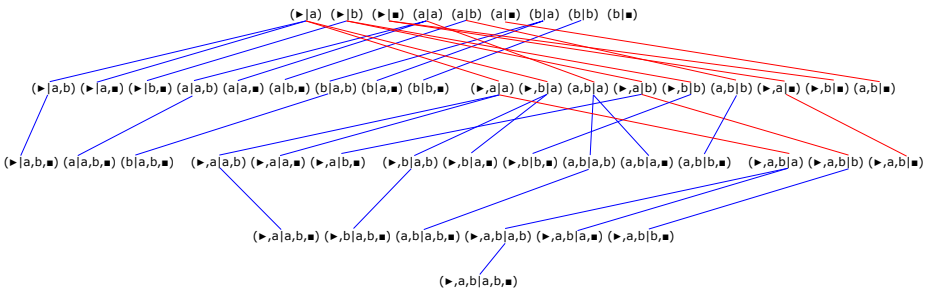


Fig. 2. Example of a tree-structured candidate space for the set of activities $\{\blacktriangleright, a, b, \blacksquare\}$, with orderings $\blacksquare >_i b >_i a >_i \blacktriangleright$ and $\blacksquare >_o b >_o a >_o \blacktriangleright$.

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

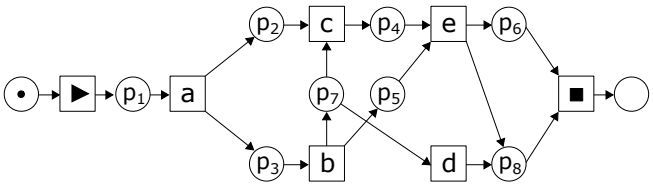


Fig. 3. The table indicates for each of the given trace variants and candidate places whether the place can replay that trace variant. The Petri net N is created by inserting all places which can replay at least $0.75 \cdot |L| = 75$ traces. However, N can replay only the first trace variant, i.e., $0.6 \cdot |L| = 60$ traces.

the Petri net by connecting it to its uniquely labeled ingoing and outgoing transitions. Consider the example event log and subset of candidate places in Fig. 3. Of the (incomplete) subset of candidate places, the places p_1 to p_8 are fitting the event log for $\tau = 0.75$. Inserting these places results in the given Petri net N, which can replay only the first trace variant corresponding to 60 % of traces. The introductory example in Fig. 1 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. This work explores strategies of maintaining the fitness threshold τ as a minimal fitness threshold of the returned, deadlock-free Petri net by inserting only a selection of the discovered fitting places.

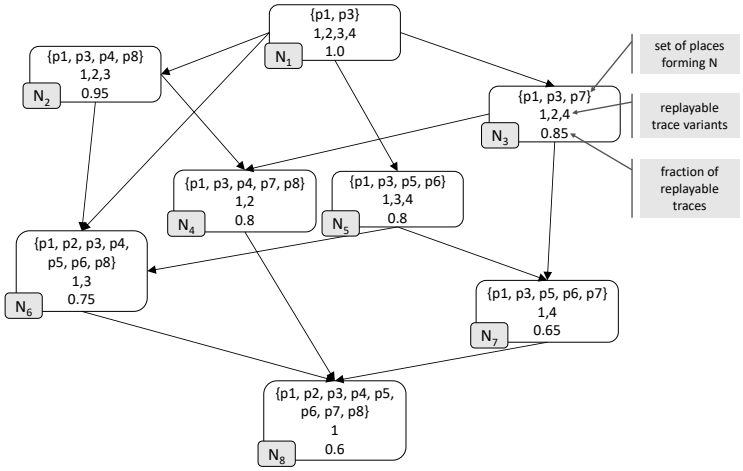


Fig. 4. Consider the set of places given in Fig. 3. This figure shows all possible combinations of these places such that adding any other place to the corresponding Petri net would decrease the number of replayable log traces. Each set of places, i.e., Petri net, is annotated with the list of trace variants it can replay and the corresponding fraction of log traces. Note, that N_8 corresponds to the Petri net shown in Fig. 3.

4 Place Selection

Consider the set of all fitting places discovered for a certain noise threshold τ. The selection of an adequate subset of these places, such that also the resulting Petri net can replay a fraction of at least τ traces, is non-trivial for a variety of reasons. First of all, the definition of an optimal solution is not straightforward. Several maximal subsets of places satisfying this requirement may exist, which differ, for example, in size, fraction of replayable traces, place complexity (number of connected activities) or subjective 'interestingness' measures for the places retained. Fig. 4 illustrates all maximal sets of places that can be built from the example places given in Fig. 3. 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 optimal solution is not clear.

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 very large 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 decsribed in detail in Subsections 4.1 and 4.2. 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 (for details compare [25,26]). While generality is desirable, additional information would be required to evaluate it, which is why we consider it outside the scope of this work.

Our fitness evaluation of the returned Petri net N uses the standard alignment-based fitness as defined in [27]. For precision, we use the approach as implemented in [28]. Simplicity is harder to evaluate, since it is a rather subjective metric that can be influenced by a variety of features. In this paper, we simplify the notion to express the fraction of net elements that are transitions, arguing that a Petri net with relatively few places is likely to be perceived as simple.

Definition 7 (Simplicity). *Given a Petri net $N = (A, P)$, we define simplicity as the fraction of nodes that are transitions, i.e.,*

$$\mathit{Simplicity}(N) = 1 - \frac{|P|}{|P| + |A|}.$$

All used quality metrics return values between 0 and 1, where a value close to 1 indicates high quality in general. However, note that for simplicity a value around 0.5 indicates a model with roughly as many places as transitions, for example a simple sequence, while a higher value would arise from a Petri net with extremely few places. Therefore, we consider a value close to 0.5 to be rather optimal in terms of simplicity.

4.1 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 Fig. 4 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 global 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 users 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. Examples for such strategies are presented in Sec. 5.

Definition 8 formalizes the use of τ, δ and *adapt* to decide for the discovered fitting places, whether they should be added, kept for later re-evaluation or discarded.

Definition 8 (Place Classification Using τ, δ and *adapt*). *Consider a set of activities A, a set of places $P \subseteq \mathbb{P}(A) \times \mathbb{P}(A)$, a place $p \in \mathbb{P}(A) \times \mathbb{P}(A)$ and an event log L. We*

use parameters $\tau \in [0,1]$ *and* $\delta \in [0,1]$*, and a function adapt*: $([0,1], \mathbb{P}(A) \times \mathbb{P}(A)) \to [0,1]$ *to categorize* p *as follows:*

$$\texttt{keep}_{L,A,\tau}(P,p) = |\mathit{fit}(L(A,P)) \cap \mathit{fit}(L,p)| \geq \tau \cdot |L|$$
$$\texttt{add}_{L,A,\tau,\delta}(P,p) = \texttt{keep}_{L,A,\tau}(P,p) \wedge |\mathit{fit}(L(A,P))| - |\mathit{fit}(L,(A,P)) \cap \mathit{fit}(L,p)| \leq \mathit{adapt}(\delta,p) \cdot |L|$$

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

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

4.2 Selection Framework

An overview of our approach, indicating inputs, outputs and use of parameters, is given in Fig. 5. Since we consider simplicity to be a desirable property, we set the eST-Miner to 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, arguing that places with many transitions are generally not desirable - such places are usually devastating to simplicity while their constraints can be sufficiently approximated by much simpler places.

After the eST-Miner framework evaluates a candidate place p to be fitting with respect to a fraction τ of traces (Def. 3), we use the *classification functions* given in Def. 8 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 by place simplicity (few transitions are better) and place fitness (number of replayable log traces). Optionally, the length of Q can be limited, 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 delta adaption functions proposed in Section 5, which give preference to places less complex than indicated by the current tree level. After reaching the lowest tree level, the approach can either terminate immediately, or iterate over the potential places queue while artificially increasing the tree depth d^+ times. This can be relevant for delta adaption functions depending on place complexity, as exemplified in Sec. 5. Here, the artificial tree depth allows for increased leniency also for the most complex places evaluated.

Finally, the resulting Petri net N may contain dead parts: in particular infrequent activities with erratic behavior are likely to occur only in those traces that are no longer replayable on N. Therefore, as a final step, we detect and remove all activities that do not occur in $\mathit{fit}(L,N)$ together with their connected arcs.

Before returning this Petri net as final output, the eST-Miner framework removes implicit places, merges places that are equal except for self-loops, i.e., $(I \cup X_1 | O \cup X_1)$ and $(I \cup X_2 | O \cup X_2)$ are merged into $(I \cup X1 \cup X_2 | O \cup X_1 \cup X_2)$, and adds start and end places.

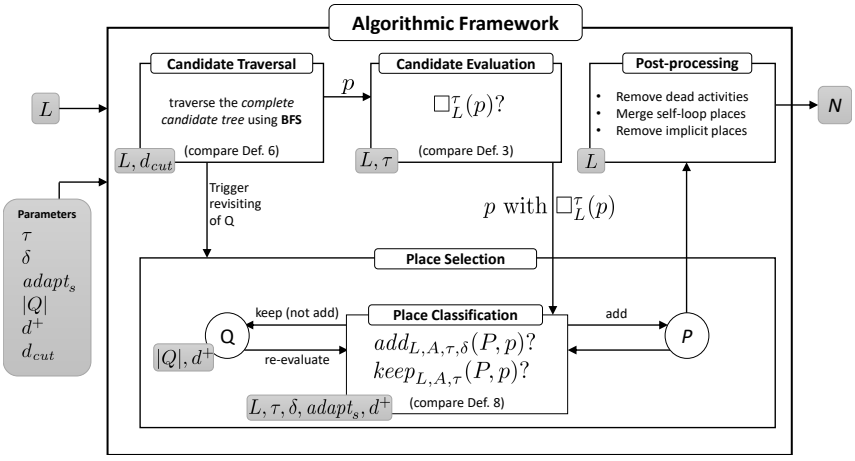


Fig. 5. Overview of the presented approach, including input, output and parameter use.

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

Theorem 1 (Guarantees). *Given a set of activities A, event log L over A, an adaption function adapt: $([0,1], \mathbb{P}(A) \times \mathbb{P}(A)) \to [0,1]$ and parameters $\tau \in [0,1], \delta \in [0,1], s \in \mathbb{N}, |Q| \in \mathbb{N}, d^+ \in \mathbb{N}, d_{cut} \in \mathbb{N}$, the presented approach 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.*

Furthermore, if the length of Q is not limited, and thus a place p is discarded only if it does not satisfy $keep_{L,A,\tau}(P,p)$, 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.

5 Selection Strategies

As illustrated by the example place combinations in Fig. 4, 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 to an investigation of the impact and suitability of our approach.

The *linear* and *sigmoid* delta adaption functions both aim 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 other strategies can be compared.

No Delta As a baseline to compare to, we introduce a function that ignores delta and simply adds every fitting place to the Petri net as soon as it is discovered. Within the

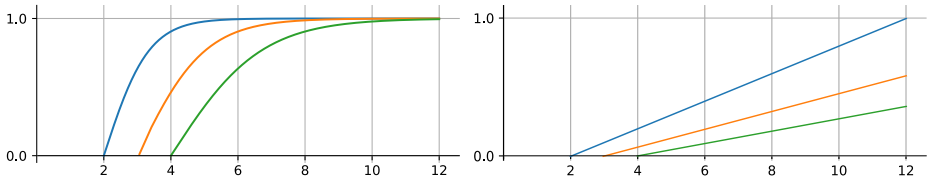


Fig. 6. Example behavior of the delta adaption modifiers $mod_{sigmoid}(\delta)$ (left) and $mod_{linear}(\delta)$ (right) for three places with $2, 3$ and 4 activities, respectively. The x-axis indicates the current tree depth d, with $d_{max} = 12$, while the y-axis indicates the modifier to be multiplied with δ.

framework, this can be formalized to

$$adapt_{noDelta}(\delta, p) = 1.$$

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:

$$adapt_{constant}(\delta, p) = \delta$$

Linear and Sigmoid Delta Adaption For a set of activities A, let $d_{max} = 2|A|$ be the maximum depth of the complete candidate tree, and let $d \in [2, 3, ..., d_{max}]$ be the current depth of the candidate tree traversal. We call $s \in \mathbb{N}\backslash\{0\}$ the *steepness modifier*. Consider a place $p = (I|O)$.

The *linear delta adaption function* computes the adapted δ as follows:

$$\begin{aligned} adapt_{linear}(\delta, (I|O)) &= \delta \cdot mod_{linear}((I|O)) \\ &= \delta \cdot \frac{s}{(|I| + |O|)} \cdot \frac{d - (|I| + |O|)}{d_{max} - 2} \end{aligned}$$

We define the *sigmoid delta adaption function* as follows:

$$\begin{aligned} adapt_{sigmoid}(\delta, (I|O)) &= \delta \cdot mod_{sigmoid}((I|O)) \\ &= \delta \cdot \left(\frac{2}{1 + \exp\left((-1) \cdot \frac{s}{(|I|+|O|)} \cdot (d - (|I| + |O|)) \right)} - 1 \right) \end{aligned}$$

Fig. 6 illustrates the behavior of the modifier each adaption function multiplies with the parameter δ for three example places of varying complexity. Both, the *linear* and *sigmoid* delta adaption are 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, both functions 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 only difference is the steepness of the functions: while the linear function increases linearly with the place complexity, the sigmoid function grows fast in the beginning but stagnates towards the end. Thus, the linear function becomes uniformly less lenient with increasing place complexity. In contrast, the sigmoid function prefers the simpler places more strongly, while the more complex places are (roughly) equally undesirable.

6 Experimental Results and Evaluation

We perform several introductory experiments where we run the proposed algorithm with a wide variation of combinations of possible parameter settings on several event logs with different properties. To investigate the impact of the proposed heuristics, we use a lexicographical ordering of the activities, thus fixing the order of candidate evaluation.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.

6.1 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 making up only 3.33 % of the event log. Activities are repeated often within a 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 algorithms ability to discover complex control flow structures, as well as the option to abstract from rare behavior. For each event log we perform 4200 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.

6.2 Results and Evaluation

For each model discovered we compute alignment-based fitness, precision and simplicity as described in Sec. 4. Based on alignment-based fitness and precision, we also present the F_1-Score, i.e., the harmonic mean of alignment-based fitness and precision.

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	[25]
Order-Handling	`Orders`	8	9	[31]

Table 2. Overview of the parameters settings used in our experimentation. The combinations result in 4200 runs for each event log. The values 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 $|Q| \geq 10000$. 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.
adapt	$adapt_{noDelta}$, $adapt_{constant}$, $adapt_{linear}$, $adapt_{sigmoid}$	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 $adapt_{linear}$ and $adapt_{sigmoid}$ only).
$\|Q\|$	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 $adapt_{linear}$ and $adapt_{sigmoid}$ only).
d_{cut}	5	Stop candidate traversal after the specified tree level.

In Fig. 7 an overview of the quality results of the 4200 models generated for each log is given. Fitness and simplicity remain rather stable, with fitness being generally high and simplicity values clustering around 0.5 (which we consider a good value, recall Def. 7). On the other hand, precision, and by extension the F_1-score, vary a lot for the discovered models. This clearly indicates that the choice of parameters has a strong impact on this quality aspect.

While we discovered only 7 unique models for `Orders`, there were 19 unique models found for `RTFM`, 27 for `Teleclaims` and 140 for `Sepsis`. The quality results and frequencies of the 10 most frequently discovered Petri nets are given in Fig. 8 (Model IDs 1 to 10). Additionally, we provide the same results for the models discovered by the Inductive Miner infrequent (IMf) with default settings as implemented in ProM [32], the models discovered by the eST-Miner with $\tau = 1.0$ (comparable to region theory results), as well as the model with the highest F_1-Score discovered over all runs.

In Figures 9, 10, 11 and 12 we present a selection of models for each log: the model discovered by IMf, the model discovered by the eST-Miner with $\tau = 1.0$, the most frequently discovered model and the model with the highest F_1-score discovered by the experiments with our proposed approach.

All models shown in Fig. 9 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 execution of *send invoice* is eventually

Fig. 7. Overview of model quality results for all 4200 runs with varying parameters but fixed candidate traversal order.

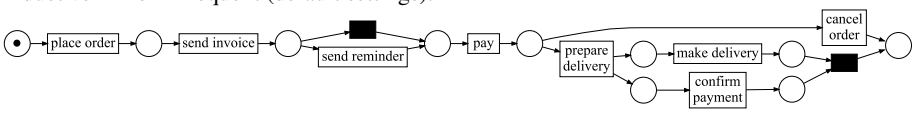
followed either by *pay* (and then delivery) or by *cancel order*, but never both. In rare cases, payment occurs before sending the invoice. After sending the invoice, reminders can be sent repeatedly, until payment is received or the order is canceled. This behavior is fully expressed only by the model 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. In contrast to the model discovered most frequently by our extended eST-Miner, the model with the highest F_1-Score does not contain the activity *cancel order* (occurring in 13.03 % of traces) at all, resulting in slightly lower fitness but notably 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. Fig. 10 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 skipable, resulting 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 most frequent model discovered by our extension, this problem becomes less severe and is significantly reduced for the discovered model with the highest F_1-Score. This model manages to capture the main behavior hidden in the traces while ignoring infrequent activity behavior, achieving comparatively high precision.

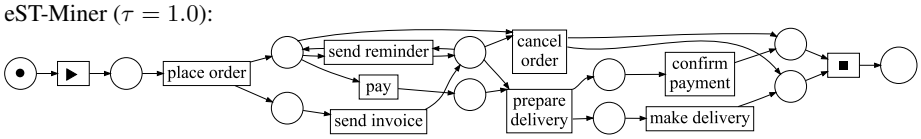
Log	Metric	IMf (default settings)	eST (τ = 0.1)	Model ID 1	2	3	4	5	6	7	8	9	10	highest F1
RTFM (discovered 19 unique models)	F1	0.6432	0.6531	0.6763	0.9638	0.6794	0.9371	0.7570	0.8816	0.9270	0.8540	0.7792	0.6757	
	Fitness	0.9820	1.0000	0.9940	0.9301	0.9642	0.9642	0.9603	0.7882	0.8640	0.8707	0.9809	0.9642	
	Precision	0.4782	0.4849	0.5125	1.0000	0.5245	0.9114	0.6248	1.0000	1.0000	0.8379	0.6463	0.5201	Model 2
	Simplicity	0.5526	0.6842	0.5714	0.5000	0.5714	0.5333	0.5200	0.4667	0.5000	0.5000	0.5333	0.5333	
	Frequency	-	-	1271	1107	383	367	312	243	150	150	34	33	
Sepsis (discovered 140 unique models)	F1	0.3215	0.3266	0.4264	0.4639	0.5205	0.5752	0.6086	0.4849	0.4672	0.6893	0.6036	0.4582	0.7836
	Fitness	0.9060	1.0000	0.9942	0.9846	0.9603	0.9681	0.9560	0.9679	0.9781	0.9230	0.9635	0.9942	0.9115
	Precision	0.1954	0.1952	0.2714	0.3034	0.3570	0.4091	0.4464	0.3235	0.3069	0.5500	0.4395	0.2977	0.6871
	Simplicity	0.5106	0.7826	0.6667	0.5926	0.5926	0.5909	0.5385	0.5806	0.6087	0.4516	0.6154	0.6522	0.52
	Frequency	-	-	408	354	198	187	186	180	180	150	150	120	52
Teleclaims (discovered 27 unique models)	F1	0.9496	0.5993	0.9316	0.4697	0.3895	0.4536	0.6272	0.9469	0.9328	0.8621	0.8202	0.8825	
	Fitness	0.9490	1.0000	0.9538	1.0000	0.9889	0.9889	0.9733	0.9244	0.8740	0.9423	0.9572	0.8461	
	Precision	0.9503	0.4279	0.9105	0.3069	0.2425	0.2943	0.4627	0.9706	1.0000	0.7945	0.7175	0.9222	Model 6
	Simplicity	0.5172	0.5200	0.5000	0.5417	0.6190	0.5909	0.5200	0.5200	0.4815	0.5200	0.5000	0.4783	
	Frequency	-	-	650	507	474	440	374	356	210	182	182	150	
Orders (discovered 7 unique models)	F1	0.9340	0.9319	0.9258	0.9664	0.9502	0.9374	0.9319	0.9096	0.9296				
	Fitness	0.9600	1.0000	0.9996	0.9562	0.9279	0.8822	1.0000	0.9279	0.9562				
	Precision	0.9094	0.8725	0.8622	0.9768	0.9735	1.0000	0.8725	0.8921	0.9044				Model 2
	Simplicity	0.5000	0.4762	0.5000	0.4737	0.5000	0.4706	0.4762	0.4737	0.4500				
	Frequency	-	-	1472	1424	1000	225	64	11	4				

Fig. 8. Overview of the qualitative evaluation results of the IMf (default settings), the eST-Miner with $\tau = 0$ and the 10 most frequently discovered models by the presented approach including their frequencies. The final column indicates the model with the best F_1-Score (only for `Sepsis` this is not contained in the most frequent models). A selection of models is presented for each event log in Figures 9 to 12.

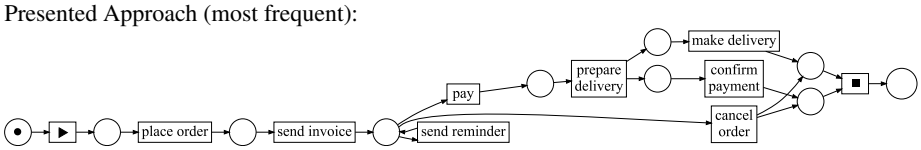
Inductive Miner infrequent (default settings):



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



Presented Approach (most frequent):



Presented Approach (highest F_1):

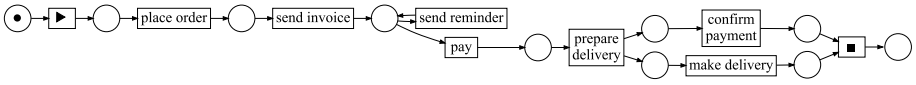
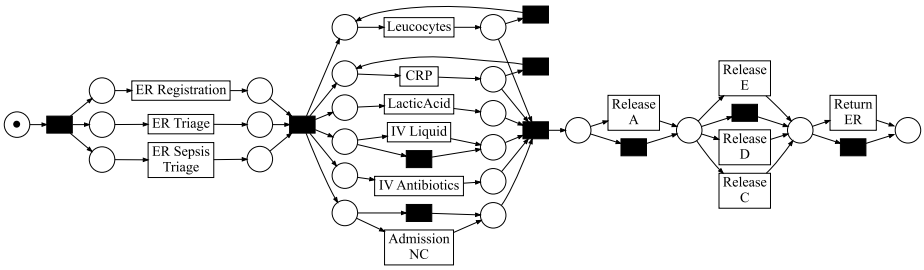


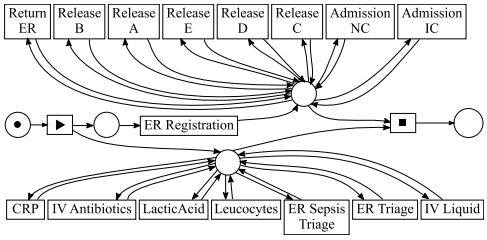
Fig. 9. The Petri nets discovered based on the `Orders` log using the Inductive Miner infrequent (default settings), the eST-Miner with $\tau = 1.0$, and Model 1 (most frequent) and Model 2 (highest F_1-Score) discovered with our runs of the presented approach.

Fig. 11 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 tenden-

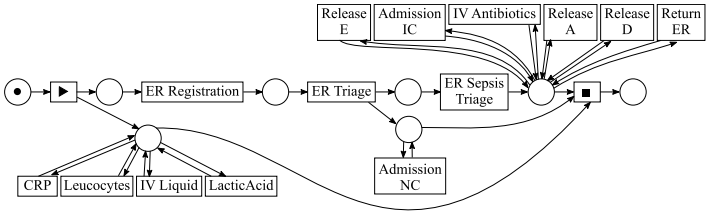
Inductive Miner infrequent (default settings):



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



Presented Approach (most frequent):



Presented Approach (highest F_1):

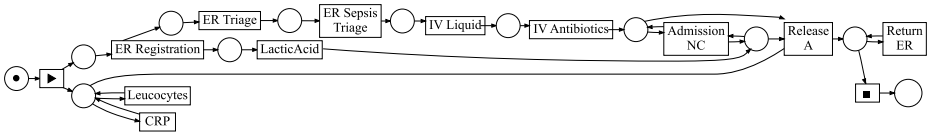
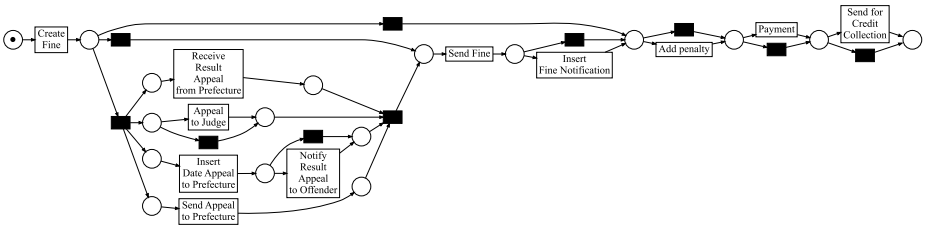


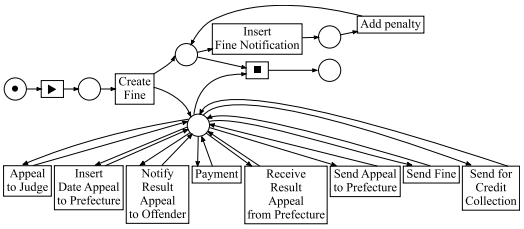
Fig. 10. The Petri nets discovered based on the `Sepsis` log using the Inductive Miner infrequent (default settings), the eST-Miner with $\tau = 1.0$, and Model 1 (most frequent) and the model with the highest F_1-Score discovered with our runs of the presented approach.

cies as for the previous logs. For the models discovered by our approach, we note that quite many 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 the appeals have been filtered to focus on the main process.

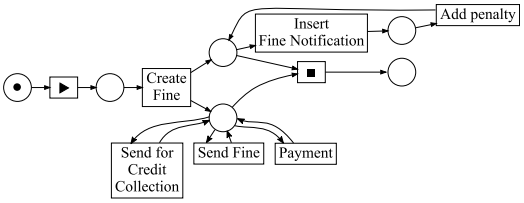
Inductive Miner infrequent (default settings):



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



Presented Approach (most frequent):



Presented Approach (highest F_1):

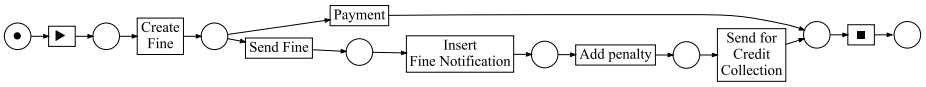
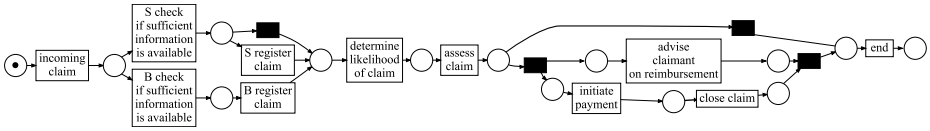


Fig. 11. The Petri nets discovered based on the `RTFM` log using the Inductive Miner infrequent (default settings), the eST-Miner with $\tau = 1.0$, and Model 1 (most frequent) and Model 2 (highest F_1-Score) discovered with our runs of the presented approach.

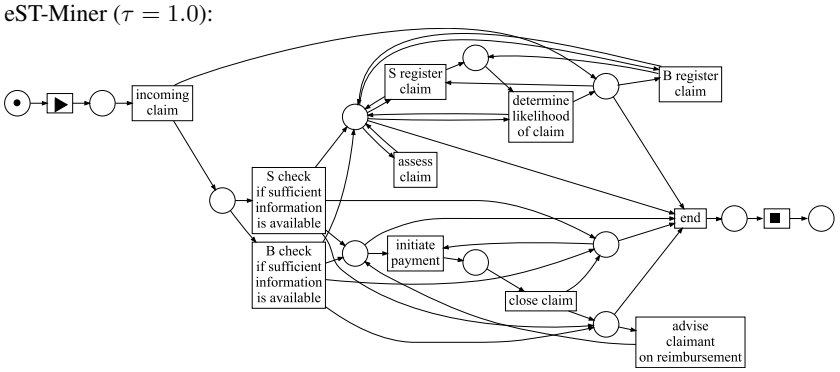
A set of process models discovered from the `Teleclaims` log is presented in Fig. 12. The models discovered by the IMf and our approach express similar behavior, with the main difference being the representation of skipable activities: with all transitions being uniquely labeled, our approach has to 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 indicate 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 9 to 12 we can observe that these models seem to abstract well from infrequent activity behavior, giving a clear representation of the

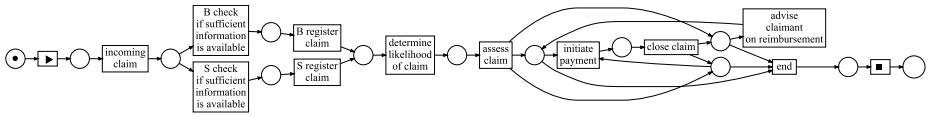
Inductive Miner infrequent (default settings):



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



Presented Approach (most frequent):



Presented Approach (highest F_1):

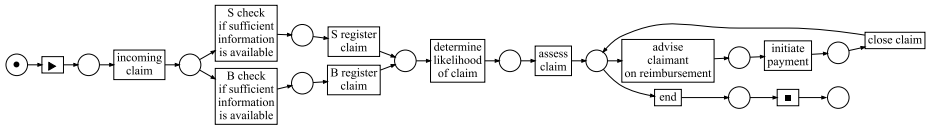


Fig. 12. The Petri nets discovered based on the `Teleclaims` log using the Inductive Miner infrequent (default settings), the eST-Miner with $\tau = 1.0$, and Model 1 (most frequent) and Model 6 (highest F_1-Score) discovered with our runs of the presented approach.

main process. However, models with higher fitness may reveal complex control-flow structures and interesting variations. The presented approach is able to return models anywhere on this scale between fitness and precision based on the choice of parameters.

While the quality metrics clearly indicate that our approach is able to discover models balancing fitness and precision while maintaining reasonable simplicity, the choice of parameters has a significant impact that requires further investigation. We used decision tree analysis to search for certain parameter settings that would result in the highest quality models as indicated by the F_1-Score. The results of this analysis are shown in Table 3, where each line represents a set of parameter combinations that leads to the discovery of the best model.

For the four event logs investigated in this paper, the most important parameters seem to be τ and δ. This is not surprising, since τ has a direct impact on which places

are available for addition to the Petri net and δ is limiting the range of the adaption strategies, which include the use of s. Notably, the artificial tree depth d^+ as well as $|Q|$ have had no major impact on the discovery of any of the examined models and the sigmoid and linear delta adaption strategies are often interchangeable.

Some interdependencies between the parameters are expected, and seem to be confirmed by the results in Table 3. For example, when using the delta adaption functions $adapt_{constant}$ and $adapt_{noDelta}$, the steepness modifier s has no impact. For the `Orders` log there is an indication of an exceptionally low s value working well with a high artificial tree depth modifier d^+, and for low τ values requiring the use of δ to discover the best model. For event logs like the `RTFM` log and the `Orders` log, which have a few very dominant trace variants, we seem to generally achieve good results for rather high values of τ. In contrast, for event logs with a high variety of traces as for example `Sepsis` log, a low τ-value seems mandatory. 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 behavior. However, more results are needed to validate such speculations.

Interestingly, the results from the `Sepsis` log, which contains a high variety of traces, seem to confirm our algorithms ability to discover the main behavior hidden in an event log even in the absence of main trace variants: for a low value of τ, e.g. $\tau = 0.3$, the fraction of log traces replayable by the return Petri net is indeed 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 using decision trees shows, that certain parameter choices result in high quality models. It gives a first indication about which parameters have a more notable impact and whether certain settings are more suitable for logs with certain properties. Further experimentation needs to be performed to investigate to which degree a generalization is possible. Note, that the impact of the candidate traversal order has not been investigated yet, and may allow for further improvements.

7 Conclusion

In this paper, we introduced an extension to the eST-Miner that returns a Petri net which satisfies user-definable minimal fitness requirements. The presented approach employs heuristics to efficiently select a suitable subset of the discovered places, while aiming towards high precision and simplicity. The algorithm is capable to discover complex control-flow structures such as non-local dependencies, to deal with noise in the event log and to provide guarantees without over- or underfitting.

Our first 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

Table 3. Overview of the parameter choices resulting in the discovery of the model with the highest F_1-Score. For each log we indicate how often this model has been discovered in our experimental runs and in which figure to find it. For each parameter that our decision tree analysis has revealed to be impactful, the possible values are indicated. Each line corresponds to a set of parameter combinations, with the frequency of the model being discovered using these combinations given to the right.

Log and Model	τ	δ	**Strategy**	s	d^+	**#Combinations**
Sepsis (Fig. 10)	0.3	0.15	$adapt_{constant}$	-	-	30
(discovered	0.3	0.25	$adapt_{sigmoid}$	$[3, 5]$	-	18
52 times)						
RTFM (Fig. 11)	$[0.3, 0.6]$	$[0.15, 0.25]$	$adapt_{constant}$	-	-	360
(discovered	$[0.4, 0.6]$	$[0.2, 0.25]$	$adapt_{sigmoid}$, $adapt_{linear}$	$[2, 5]$	-	288
1107 times)	0.6	-	$adapt_{noDelta}$	-	-	150
	$[0.4, 0.6]$	0.15	$adapt_{linear}$	$[3, 5]$	-	54
Teleclaims (Fig. 12)	$[0.3, 0.4]$	$[0.15, 0.25]$	$adapt_{constant}$	-	-	180
(discovered	$[0.3, 0.4]$	$[0.2, 0.25]$	$adapt_{sigmoid}$	$[4, 5]$	-	48
356 times)	$[0.3, 0.4]$	0.15	$adapt_{linear}$	$[3, 4]$	-	24
	$[0.3, 0.4]$	0.25	$adapt_{linear}$	$[4, 5]$	-	24
Orders (Fig. 9)	$[0.7, 0.8]$	$[0.15, 0.25]$	-	[2,5]	-	576
(discovered	$[0.3, 0.6]$	$[0.15, 0.2]$	$adapt_{sigmoid}$, $adapt_{constant}$	[2,5]	-	384
1424 times)	$[0.7, 0.8]$	-	$adapt_{noDelta}$	-	-	120
	$[0.7, 0.8]$	$[0.15, 0.25]$	-	1	10	72

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.

Next to an analysis of the running-time, future work includes further experimentation to explore the generalization the preliminary results, as well as the impact of the candidate place traversal order and its interaction with the heuristics used. Improvements or variations of the strategies are likely possible. 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.

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 my 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: Special thanks goes to Tobias Brockhoff for supporting the implementation and evaluation of the presented experiments. We thank the Ministry of Culture and Science of the German State of North Rhine-Westphalia (MKW) and the Excellence Strategy of the Federal Government and the Länder for supporting our research.

References

1. Reisig, W.: Understanding Petri Nets: Modeling Techniques, Analysis Methods, Case Studies. Springer Berlin Heidelberg (2013)
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)
4. Berthelot, G.: Transformations and decompositions of nets. In: Petri Nets: Central Models and Their Properties, pp. 359–376. Springer, Berlin, Heidelberg (1987)
5. Wen L. van der Aalst, W.M.P., Wang, J., Sun, J.: Mining process models with non-free-choice constructs. Data Mining and Knowledge Discovery **15**(2), 145–180 (2007)
6. 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)
7. Badouel, E., Bernardinello, L., Darondeau, P.: Petri Net Synthesis. Text in Theoretical Computer Science, EATCS Series. Springer (2015)
8. 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, pp. 637–647. IEEE Press, Piscataway, NJ, USA (2007)
9. Bergenthum, R., Desel, J., Lorenz, R., Mauser, S.: Process mining based on regions of languages. Proc. 5th Int. Conf. on Business Process Management pp. 375–383 (2007)
10. van der Werf, J.M., van Dongen, B., Hurkens, C., Serebrenik, A.: Process discovery using integer linear programming. In: Applications and Theory of Petri Nets. Springer, Berlin, Heidelberg (2008)
11. van Zelst, S., van Dongen, B., van der Aalst, W.: Avoiding over-fitting in ILP-based process discovery. In: Business Process Management, pp. 163–171. Springer International Publishing, Cham (2015)
12. van Zelst, S., van Dongen, B., van der Aalst, W.: ILP-based process discovery using hybrid regions. In: ATAED@Petri Nets/ACSD (2015)
13. Carmona, J., Cortadella, J., Kishinevsky, M.: A region-based algorithm for discovering Petri nets from event logs. In: Business Process Management, p. 358–373. Springer (2008)
14. Darondeau, P.: Deriving unbounded Petri nets from formal languages. In: CONCUR'98 Concurrency Theory, pp. 533–548. Springer, Berlin, Heidelberg (1998)
15. Bergenthum, R., Desel, J., Lorenz, R., Mauser, S.: Synthesis of Petri nets from finite partial languages. Fundam. Inf. **88**(4), 437–468 (2008)
16. Ehrenfeucht, A., Rozenberg, G.: Partial (set) 2-structures. Acta Informatica **27**(4), 343–368 (1990)
17. Carmona, J., Cortadella, J., Kishinevsky, M., Kondratyev, A., Lavagno, L., Yakovlev, A.: A symbolic algorithm for the synthesis of bounded Petri nets. In: K.M. van Hee, R. Valk (eds.) Applications and Theory of Petri Nets, pp. 92–111. Springer Berlin Heidelberg, Berlin, Heidelberg (2008)
18. 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, p. 280–300. Springer-Verlag, Berlin, Heidelberg (2020)
19. Mannel, L.L., van der Aalst, W.M.P.: Finding complex process-structures by exploiting the token-game. In: Application and Theory of Petri Nets and Concurrency. Springer Nature Switzerland AG (2019)
20. van der Aalst, W.M.P.: 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, pp. 1–20 (2018)

21. Garcia-Valles, F., Colom, J.: Implicit places in net systems. Proceedings 8th International Workshop on Petri Nets and Performance Models pp. 104–113 (1999)
22. Berthomieu, B., Botlan, D.L., Dal-Zilio, S.: Petri net reductions for counting markings. CoRR **abs/1807.02973** (2018)
23. Colom, J., Silva, M.: Improving the linearly based characterization of p/t nets. In: Advances in Petri Nets 1990, pp. 113–145. Springer, Berlin, Heidelberg (1991)
24. Mannel, L.L., Bergenthum, R., van der Aalst, W.M.P.: 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, vol. 2625, pp. 20–32. CEUR-WS.org
25. van der Aalst, W.: Process Mining: Data Science in Action, 2 edn. Springer, Heidelberg (2016)
26. Carmona, J., van Dongen, B., Solti, A., Weidlich, M.: Conformance Checking - Relating Processes and Models. Springer, Cham (2018)
27. Adriansyah, A.: Aligning observed and modeled behavior. Ph.D. thesis, Mathematics and Computer Science (2014)
28. Munoz-Gama, J., Carmona, J.: A fresh look at precision in process conformance. In: BPM, vol. 6336, pp. 211–226 (2010)
29. Mannhardt, F.: Sepsis cases - event log (2016)
30. De Leoni, M., Mannhardt, F.: Road traffic fine management process (2015)
31. van der Aalst, W.M.P.: Spreadsheets for BPM. Business Process Management Journal **24**, 105–127 (2010)
32. 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, pp. 444–454. Springer, Berlin, Heidelberg (2005)