Memory-Efficient Alignment of Observed and Modeled Behavior

A. Adriansyah, B.F. van Dongen, and W.M.P. van der Aalst

Department of Mathematics and Computer Science
Eindhoven University of Technology
P.O. Box 513, 5600 MB Eindhoven, The Netherlands
(a.adriansyah,b.f.v.dongen,w.m.p.v.d.aalst)@tue.nl

Abstract. Experience shows that in systems where process executions are not strictly enforced by process models, often deviations occur. Alignments between logged process executions and models reveal useful insights and can be used for both conformance checking and performance analysis. In this article, we present a memory-efficient approach using marking equations of Petri nets to calculate optimal alignments between process executions and process models. A comparative study shows that in most cases the approach significantly reduces the memory required. This makes it possible to analyze larger logs and models using alignments. The more deviations exist, the better the approach performs compared to the approaches without using marking equations. The approach has been implemented, tested against both artifical and real life logs, and is publicly available as part of the ProM 6 framework.

1 Introduction

Process models are created to document processes, provide insights, automate processes, and to evaluate alternative designs [11]. Analysis techniques ranging from verification to simulation all start from process models (e.g. [16,37,38]). However, *process mining* research shows that hand-made process models often do not describe reality well [27,17]. Real processes tend to *deviate* from the corresponding *normative* or *descriptive* models. Therefore, it is essential to *diagnose the conformance of event logs and process models*. Conformance checking techniques highlight differences between observed behavior (i.e., event logs) and modeled behavior. Even when models are discovered using process mining techniques, it is important to compare the modeled and observed behavior, e.g., to judge the quality of the process discovery technique.

To diagnose the conformance of a model with respect to an event log, activities in the log need to be related to tasks in the model. The alignment should clearly show where the log and the model disagree, for example, show observed activities in the log that are not allowed according to the model, and vice versa. Furthermore, event logs may not necessarily contain all the activities executed. Logging everything might be costly and affect the performance of process executions. Still, the unlogged activities can influence process behavior. The alignment should also identify such activities in event logs to prevent misleading diagnosis.

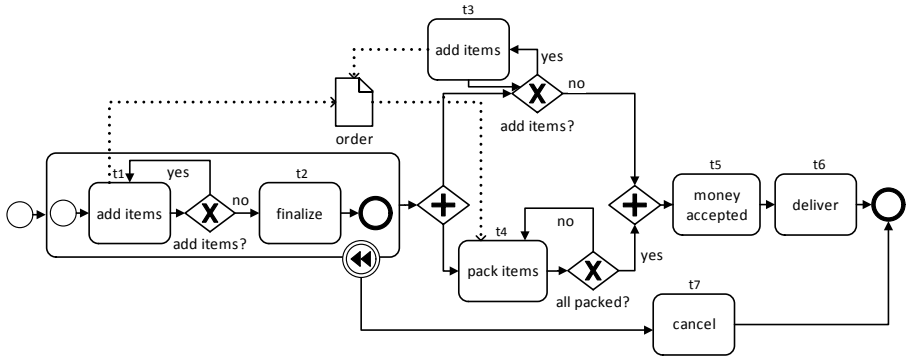

Fig. 1. An online transaction for an electronic bookstore in BPMN notation

add items	*cancel*	*finalize*	*pack items*	*money accepted*
add items	*cancel*	≫	≫	≫
t_1	t_7			

Fig. 2. An alignment between trace ⟨*add items, cancel, finalize, pack items, money accepted*⟩ and the model in Figure 1

Alignments are not only useful for identifying and measuring deviations (e.g. [4,2,25,1]), but also for other types of analysis, such as auditing [31] and compliance analysis [22]. Furthermore, knowing the relation between observed and modeled behavior and deviations enables various types of analysis driven by observed events. For example, non deviating events can be used to derive simulation parameter [24] and analyze performance [3]. Knowing frequently occured deviations enables process model repair [12] and process improvement in general [27,28].

Take for example the BPMN model shown in Figure 1 which describes an online transaction process for an electronic bookstore. The model has 7 tasks, each task is labeled with an activity it represents. Customers can add as many items to their cart before finalizing the order. After an order is finalized, items in the order are packed individually. All items of an order are sent to customers after they are packed and payment for the order is authorized and received. Customers may add items after their orders are finalized, but they can only cancel their order as long as it has not been finalized. The process ends when all goods are sent or the process is cancelled. Note that the two tasks in the model labeled *add items* represent the same activity that is performed in different contexts: before or after orders are finalized.

Consider the trace $\sigma = \langle$*add items, cancel, finalize, pack items, money accepted*$\rangle$ that is recorded in an event log. A possible alignment between σ and

add items	*cancel*	*finalize*	*pack items*	*money accepted*	$\gg$
add items	$\gg$	*finalize*	*pack items*	*money accepted*	*deliver*
t_1		t_2	t_4	t_5	t_6

Fig. 3. Alignment between the same trace and model as the one shown in Figure 2, but with less number of deviating columns

the model is shown in Figure 2. The alignment is presented as a table with two rows. The top row represents the trace as recorded in the event log, while the bottom row represents a sequence of tasks allowed by the model to terminate properly. We write task identifier below each task to uniquely distinguish two different tasks with the same activity label, e.g. tasks t_1 and t_3 are both labeled with *add items*. Deviations occur at positions where either the top or the bottom row contains a "no move" (i.e. $\gg$): a logged activity cannot be executed according to the model, or an activity should have occured according to the model but did not appear in the log. Note that we do not consider elements other than tasks such as gateways, events, connectors, etc. as they are used to describe possible behavior rather than representing some real actions.

Alignment in Figure 2 shows that after adding items and canceling order, the process should have finished according to the model, while in reality some activities were still performed (i.e. *finalize*, *pack items*, and *money accepted*). Assuming that the alignment in Figure 2 is a proper alignment of model and reality, there are more deviating than non-deviating columns.

We refer to each column of the alignment as a *move*. An element in the top row is a movement in the log and an element in the bottom row is a movement in the model. If we move both in the log and the model, we call that a *synchronous move*. If we move only in the log and not in the model (or the other way around), we call it *move on log* (*move on model*).

Consider the alignment in Figure 3 between the same trace and model as the one used to construct Figure 2. Alignment in Figure 3 has more synchronous moves and less moves on log/moves on model than the one in Figure 2. Thus, Figure 3 shows a "better" alignment because the deviations between the trace and the model are not as severe as the ones indicated by Figure 2. Therefore, the alignment in Figure 3 provides a better intuition showing which events are deviating from the process model. Given a trace and a process model, the challenge is to identify *optimal alignments* between them, i.e. the ones that shows least number of deviations.

Alignment-based analysis needs to be robust with respect to tasks in the model whose activities are not logged (i.e. invisible tasks) and tasks in the model that are labeled with the same activity (i.e. duplicate tasks) [5]. For example, suppose that the activity *deliver* in the process shown in Figure 1 is not logged, the move on model of task t_6 in Figure 3 should not be accounted as deviations. Creating an assignment is essentially an optimization problem and may be very

time consuming in case of large event logs and models with many (or even infinitely) possible execution sequences.

In this article, we consider process models in the form of Petri nets [18]. We extend the approach to construct alignments in [4] that is based on the $\mathcal{A}^\star$ algorithm [13] by providing a better estimation function. The function prunes search spaces and guides state space exploration using the *marking equation* for Petri nets. The idea is that partial alignments that are "hopeless" can be discarded as soon as possible (e.g., if the final marking cannot be reached anymore) and if there are better alignments, one does not need to further explore a partial alignment. This technique significantly reduces the memory usage of the approach without necessarily sacrificing computation time. The approach is extendable to any modelling languages for which translation to Petri nets is available. In fact, in this paper we also show that the approach is extendable to Petri nets with reset/inhibitor arcs. A wide range of process modeling languages, such as BPMN [20], Web Services Business Process Execution Language (WS-BPEL) [19], YAWL [29], EPCs [26], and UML's sequence diagram [6] can be mapped onto Petri nets with reset/inhibitor arcs. We show that our approach outperforms existing approaches and is applicable to real-life scenarios.

Section 2 present preliminaries used throughout this paper. Section 3 defines the core problem: finding an optimal alignment relating a trace and a model. Section 4 shows how the problem can be formulated as a shortest path problem, and Section 5 shows how the shortest path can be constructed efficiently in terms of memory usage. Section 6 extends the memory-efficient approach to deal with Petri nets having reset/inhibitor nets (thus enabling the analysis of languages allowing for cancellation, priorities, etc.). Section 7 shows experimental results, and Section 8 concludes the article.

2 Preliminaries

In this section, we define basic notations for matrices, vectors, sets, sequences, graphs, and Petri nets.

Definition 2.1. (Matrix, vector) A *matrix* is a rectangular array of numbers. Let $\mathbf{M}$ be a matrix of size $m \times n$. $\mathbf{M}_{i,j}$ denotes the element of matrix $\mathbf{M}$ in the i-th row and j-th column where $1 \leq i \leq m$ and $1 \leq j \leq n$. $\mathbf{M}^T$ of size $n \times m$ is the *transpose* of $\mathbf{M}$ such that for all $1 \leq i \leq m, 1 \leq j \leq n : \mathbf{M}_{i,j} = \mathbf{M}^T_{j,i}$.

Let $\mathbf{M_1}$ and $\mathbf{M_2}$ be a pair of matrices with the same size $m \times n$. The addition of two matrices $\mathbf{M_1}$ and $\mathbf{M_2}$, denoted $\mathbf{M_1} + \mathbf{M_2} = \mathbf{M_3}$ such that for all $1 \leq i \leq m, 1 \leq j \leq n, \mathbf{M_3}_{i,j} = \mathbf{M_1}_{i,j} + \mathbf{M_2}_{i,j}$. Substraction of two matrices is defined in the same way as addition by replacing summation '+' with substraction '-'. The *dot product* of $\mathbf{M_1}$ and $\mathbf{M_2}$, denoted $\mathbf{M_1} \cdot \mathbf{M_2} = \sum_{1 \leq i \leq m, 1 \leq j \leq n} \mathbf{M_1}_{i,j} \cdot \mathbf{M_2}_{i,j}$.

Let $\mathbf{M_4}$ be a matrix of size $n \times o$. The *cross product* of $\mathbf{M_1}$ and $\mathbf{M_4}$ is a matrix $\mathbf{M_5}$ of size $m \times o$, denoted $\mathbf{M_1} \times \mathbf{M_4} = \mathbf{M_5}$, such that for all $1 \leq i \leq m, 1 \leq j \leq o, \mathbf{M_5}_{i,j} = \sum_{r=1}^{n} \mathbf{M_1}_{i,r} \cdot \mathbf{M_4}_{r,j}$.

A *row vector* $\vec{v}$ of size m is a matrix of size $1 \times m$. Similarly, a *column vector* $\vec{w}$ of size n is a matrix of size $n \times 1$. $\vec{v}_i$ and $\vec{w}_i$ refer to the value of the i-th column in $\vec{v}$ and the value of the i-th row in $\vec{w}$ respectively.

Definition 2.2. (Sets, sequences, multi-sets)
$\mathbb{N}$, $\mathbb{R}$, and $\mathbb{R}^+$ denote the set of all natural numbers, positive real numbers, and positive real numbers without zero respectively. Let $W = \{w_1, \ldots, w_n\}$ be a *set* of size $n \in \mathbb{N}$. $|W| = n$ denotes the size of set W and $\mathcal{P}(W)$ is the powerset of set W, e.g. $\mathcal{P}(\{a, b\}) = \{\emptyset, \{a\}, \{b\}, \{a, b\}\}$. For all functions $f : A \not\to B$ partially mapping elements of set A to B, $\mathrm{Dom}(f)$ denotes the domain of f and $\mathrm{Rng}(f)$ denotes the range of f.

W^* denotes the set of all finite *sequences* over W. $\langle\rangle$ denotes the empty sequence. A non-empty sequence $\sigma = \langle\sigma[1], \ldots \sigma[n]\rangle$ is given by listing its elements between angled brackets, where $\sigma[i]$ refers to the i-th element of a sequence and $|\sigma| = n$ denotes the length of σ. For all $w \in W, \sigma(w)$ counts the number of occurrences of w in σ, i.e. $\sigma(w) = |\{1 \leq i \leq |\sigma| \mid \sigma[i] = w\}|$. Concatenation of two sequences σ and σ' is denoted with $\sigma \cdot \sigma'$. Similarly, concatenation of an element a and a sequence σ is denoted $a \cdot \sigma$. Prefix sequences are denoted with $<$, such that $\sigma < \sigma'$ if and only if there is a sequence $\sigma'' \neq \langle\rangle$ with $\sigma' = \sigma \cdot \sigma''$. When we iterate over $w \in \sigma$, we take multiple occurrences of the same value into account, e.g. for all $f : W \to \mathbb{N}$, $\sum_{w \in \sigma} f(w) = \sum_{1 \leq i \leq |\sigma|} f(\sigma[i])$. For all $W' \subseteq W$, $\sigma_{\downarrow W'}$ denotes the projection of a sequence $\sigma \in W^*$ on W', e.g., $\langle a, a, b, c\rangle_{\downarrow\{a,c\}} = \langle a, a, c\rangle$.

We assume all sets to be countable and totally ordered, i.e. for any set $W = \{w_1, \ldots, w_n\}$, we assume a bijection $\lambda : W \to \{1, \ldots, |W|\}$ exists, and for all $1 \leq i \leq |W|$, we write $W[i]$ as a shorthand for $\lambda^{-1}(i)$. The *parikh vector* $\vec{\sigma}$ of a sequence σ over W is a column vector, such that $\vec{\sigma} = (\sigma(W[1]), \ldots, \sigma(W[n]))^T$, i.e. $\forall_{1 \leq i \leq |W|} \vec{\sigma}_i = \sigma(W[i])$.

A *multi-set* m over W is a function $m : W \to \mathbb{N}$. We write e.g. $m = [x, y^2]$ for a multi-set m over W where $x, y \in W, m(x) = 1, m(y) = 2$, and $m(z) = 0$ for all $z \in W \setminus \{x, y\}$. The *parikh vector* $\vec{m}$ of m is a column vector, such that $\vec{m} = (m(W[1]), \ldots, m(W[|W|]))^T$, i.e. $\forall_{1 \leq i \leq |W|} \vec{m}_i = m(W[i])$. For all $W' \subseteq W, m_{\downarrow W'} : W \to \mathbb{N}$ denotes the projection of m to domain W', such that for all $w' \in W', m_{\downarrow W'}(w') = m(w')$ and $m_{\downarrow W'}(x') = 0$ for all $x' \in W \setminus W'$.

In this article, we use A to denote a set of activities that appear in a business process, i.e. either in a model and/or in a log.

Definition 2.3. (Tuples) Let W be a set and let $z = (x_1, x_2, \ldots, x_n) \in W_1 \times \ldots \times W_n$ be a tuple of n elements. $\pi_i(z)$ refers to the i-th element of z, e.g. Let $(a, b) \in W \times W$ be a tuple of 2 elements (i.e. *pair*), $\pi_1((a, b)) = a$ and $\pi_2((a, b)) = b$. We generalize this notation to sequences of tuples, e.g for all sequence of pairs $\sigma \in (W \times W)^*, \pi_i(\sigma) = \langle\pi_i(\sigma[1]), \ldots, \pi_i(\sigma[|\sigma|])\rangle$.

An optimal alignment between a given process model and a trace is constructed by exploring the state space of the model and the trace. We consider state spaces as labeled directed graphs.

Definition 2.4. (Labeled Directed Graphs, Path, Shortest Path) A *labeled directed graph* is a tuple $G = (N, E, L)$ where N is a set of nodes, $E \subseteq N \times L \times N$ is a set of labeled edges with labels L. For all nodes $n, n' \in N$, a *path* from n to n' is a sequence of edges $\sigma \in E^*$, where $\sigma = \langle\rangle \implies n = n'$ and $\sigma \neq \langle\rangle \implies \pi_1(\sigma[1]) = n, \pi_3(\sigma[|\sigma|]) = n'$, and for all $1 \leq i < |\sigma| : \pi_3(\sigma[i]) = \pi_1(\sigma[i+1])$. $\Psi_G(n, n')$ is the set of all paths from n to n' in G. The annotation G can be omitted if the context is clear.

Let $\delta : E \to \mathbb{R}$ be a distance function. The *distance* of a path $\delta(\sigma)$ is the sum of distances of all edges in path σ where we abuse the distance function notation δ, such that $\delta(\sigma) = \sum_{1 \leq j \leq |\sigma|} \delta(\sigma[j])$. A path $\sigma \in \Psi(n, n')$ is a *shortest path* from n to n' if for all $\sigma' \in \Psi(n, n'), \delta(\sigma) \leq \delta(\sigma')$.

We translate the problem of finding optimal alignments as shortest path problem in labeled directed graphs. We use a strategy based on the $\mathcal{A}^\star$ algorithm [13] to find a shortest path between two nodes in such graphs.

Definition 2.5. (The $\mathcal{A}^\star$ Algorithm, Permissible Underestimation Function) Let $G = (N, E, L)$ be a labeled directed graph and let $\delta : E \to \mathbb{R}^+$ be a distance function. The $\mathcal{A}^\star$ *algorithm* for graph G is a function $a^\star_{G,\delta} : (N \times N) \to E^*$ such that for a source node $n_s \in N$ and target node $n_t \in N$, $a^\star_{G,\delta}(n_s, n_t) = \sigma$ where $\sigma \in \Psi(n_s, n_t)$ is a shortest path from n_s to n_t.

The $\mathcal{A}^\star$ algorithm requires a *permissible underestimation function* $h_G : (N \times N) \to \mathbb{R}$ that returns an underestimation of the distance between nodes, such that for all $n, n' \in N$:

- $h_G(n, n') = +\infty$ if $\Psi_G(n, n') = \emptyset$, and
- $h_G(n, n') \leq \delta(\sigma)$ for all $\sigma \in \Psi_G(n, n')$ otherwise.

Annotations G, δ on $a^\star$ and h can be omitted if the context is clear. Notice that the $\mathcal{A}^\star$ algorithm require distance function that maps edges to positive non-zero values.

Note that in Definition 2.5, the $\mathcal{A}^\star$ algorithm require graphs where the distance of edges are positive and non-zero. Given a graph, a source node, a target node, and a permissible underestimation function, the $\mathcal{A}^\star$ algorithm works in a best-first search manner. Priority is assigned to a node in the graph based on the sum of the shortest distance from the source node to the node and the estimated remaining distance from the node to the target node. In situations where there exists an infinite path without loop with total distance of 0 from the currently visited node and the estimated remaining distance for all nodes in the path is the same, the algorithm does not terminate because it explores the infinite path. This does not happen if the distance between nodes are positive non-zero. Notice that permissible underestimation function may provide value of 0.

As mentioned in Section 1, the approach presented in this article exploits Petri net theory, but is extendable for reset/inhibitor nets and other modeling formalisms for which translation to reset/inhibitor nets are available. Petri net and related concepts are defined as usual.

Definition 2.6. (Petri Net, Incidence Matrix) A *Petri net* over a set of activities A is a tuple $N = (P, T, F, \alpha, m_i, m_f)$ where P and T are sets of places and transitions respectively. $F : (P \times T) \cup (T \times P) \to \mathbb{N}$ is a flow relation that returns the weight of arcs. $m_i, m_f : P \to \mathbb{N}$ are the initial marking and the final marking, respectively, and $\alpha : T \not\to A$ is a partial function mapping transitions to activities. A transition $t \in T$ is *invisible* if $t \notin \mathrm{Dom}(\alpha)$.

A *marking*, i.e. a state of the Petri net, is a a multiset of places. A transition $t \in T$ is *enabled* at marking $m : P \to \mathbb{N}$ if and only if $\forall_{p \in P}\ F(p,t) \leq m(p)$ hold. $m \xrightarrow{t}_N m'$ denotes the *firing* of an enabled transition t in net N from m that leads to new marking $m' : P \to \mathbb{N}$, such that $\forall_{p \in P}\ m'(p) = m(p) - F(p,t) + F(t,p)$. A sequence $\varrho \in T^*$ of transitions is a *firing sequence* from marking m to m' if $m \xrightarrow{\varrho[1]}_N m_1 \xrightarrow{\varrho[2]}_N m_2 \ldots \xrightarrow{\varrho[|\varrho|]}_N m'$, abbreviated with $m \xrightarrow{\varrho}_N m'$. Sequence $\varrho \in T^*$ is a *complete firing sequence* if $m_i \xrightarrow{\varrho}_N m_f$. The annotation N of firing sequence is omitted if the context is clear.

The *incidence matrix* $\mathbf{N}$ of N is a $|P| \times |T|$ matrix such that forall $1 \leq j \leq |P|, 1 \leq k \leq |T|, \mathbf{N}_{j,k} = F(T[k], P[j]) - F(P[j], T[k])$.

Transitions for Petri nets correspond to tasks in process models: They are both labeled with activities. In the remainder, we use term "transition" and "task" interchangeably. The full behavior of a Petri net can be characterized by a transition system, formalized as follows.

Definition 2.7. (Transition System of Petri Net)
Let $N = (P, T, F, \alpha, m_i, m_f)$ be a Petri net over a set of activities A. The *transition system* of N is a tuple $M = (S, E, T, \alpha, s_i, s_f)$ where:

- $S = \{m : P \to \mathbb{N} \mid \exists_{\varrho \in T^*}\ m_i \xrightarrow{\varrho} m\}$ is the set of all reachable markings,
- $E = \{(m, t, m') \in S \times T \times S \mid m, m' \in S \wedge t \in T \wedge m \xrightarrow{t} m'\}$ is the set of all state transitions,
- $s_i = m_i$ is the initial state, and
- $s_f = m_f$ is the final state.

Note that (S, E, T) is a labeled directed graph. $TS(N)$ denotes the transition system of N.

Using the notations introduced in this section, we can now formally define the problem of constructing an optimal alignment.

3 Problem Statement

Alignments are created by comparing recorded process executions to process models. Typically, an instance of a process does not directly influence other instances of the same process. Take for example an insurance claim handling process in an insurance company. The way a claim is handled does not influence the way other claims are handled. Therefore, the construction of an alignment is performed separately for each instance of a process (i.e. trace).

An alignment between a trace and a Petri net is constructed by pairing each activity in the trace to a transition allowed by the net. By such, we match each movement in the trace to a movement allowed by the model. If a movement in the trace cannot be mimicked by a movement allowed by the model (or the other way around), no movement is performed in either the trace or the model. We explicitly denote "no move" by $\gg$. For convenience, for all sets W, we introduce the set $W^{\gg} = W \cup \{\gg\}$ where $\gg \notin W$.

Definition 3.1. (Movement Sequence) Let $\sigma \in A^*$ be a trace over a set of activities A and let $N = (P, T, F, \alpha, m_i, m_f)$ be a Petri net over A. A *movement sequence* $\gamma \in (A^{\gg} \times T^{\gg})^*$ between σ and N is a sequence of pairs such that

- $\pi_1(\gamma)_{\downarrow A} \leq \sigma$, i.e. its sequence of movements in the trace (ignoring $\gg$) is a prefix of σ,
- There exists a complete firing sequence $\varrho \in T^*$ such that $\pi_2(\gamma)_{\downarrow T} \leq \varrho$, i.e. its sequence of movements in the model (ignoring $\gg$) is a prefix of a complete firing sequence,
- For all $(a, t) \in \gamma$, $(a, t) \neq (\gg, \gg)$ and $a = \alpha(t)$ if $a \neq \gg$ and $t \neq \gg$, i.e. each pair consists of either an activity and a corresponding transition, or an activity/transition and $\gg$.

For all tuples $(a, t) \in \gamma$ in a movement sequence, we say that (a, t) is one of the following *movements*:

- *move on log* if $a \in A$ and $t = \gg$,
- *move on model* if $a = \gg$ and $t \in T$,
- *synchronous move* if $a \in A$ and $t \in T$.

Recall that an alignment between a trace and a model is a sequence of pairs of activities in the trace and transitions in the model. If the trace is perfectly aligned to the model, each movement in the trace can be mimicked by a movement in the model. Furthermore, the trace ends exactly when the final state of the model is reached. Therefore, we define an alignment as follows:

Definition 3.2. (Alignment) Let $\sigma \in A^*$ be a trace over a set of activities A and let $N = (P, T, F, \alpha, m_i, m_f)$ be a Petri net over A. An *alignment* $\gamma \in (A^{\gg} \times T^{\gg})^*$ between σ and N is a movement sequence such that:

- $\pi_1(\gamma)_{\downarrow A} = \sigma$, i.e. its sequence of movements in the trace (ignoring $\gg$) yields the trace, and
- $m_i \xrightarrow{\pi_2(\gamma)_{\downarrow T}} m_f$, i.e. its sequence of movements in the model (ignoring $\gg$) yields a complete firing sequence of N.

$\Gamma_{\sigma,N}$ is the set of all alignments between a trace σ and a Petri net N.

The middle row in Figure 2 and Figure 3 can be derived using α. Therefore, alignments and movement sequences do not explicitly list transition labels in Definitions 3.1 and 3.2.

Note that alignments require termination of both trace and process model. Thus, no alignment can be constructed for process models whose termination

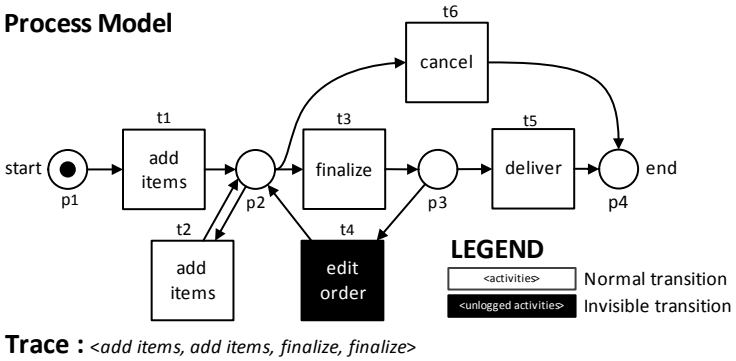

Fig. 4. Example of a Petri Net model and an unfitting trace

$$\gamma_1 = \begin{array}{|c|c|c|c|c|}\hline \textit{add items} & \textit{add items} & \textit{finalize} & \textit{finalize} & \gg \\ \hline \textit{add items} & \textit{add items} & \textit{finalize} & \gg & \textit{deliver} \\ t_1 & t_2 & t_3 & & t_5 \\ \hline \end{array}$$

$$\gamma_2 = \begin{array}{|c|c|c|c|c|c|}\hline \textit{add items} & \textit{add items} & \textit{finalize} & \gg & \textit{finalize} & \gg \\ \hline \textit{add items} & \textit{add items} & \textit{finalize} & \textit{edit order} & \textit{finalize} & \textit{deliver} \\ t_1 & t_2 & t_3 & t_4 & t_3 & t_5 \\ \hline \end{array}$$

Fig. 5. Two possible alignments between the trace and model in Figure 4

state is not reachable from the initial state. In the remainder, we only consider process models whose termination state is reachable from its initial state, i.e. *easy sound* models [30]. Easy soundness is a weaker correctness notion than classical soundness or relaxed soundness [30]. We can formulate the easy soundness [30] as follows:

Definition 3.3. (Easy soundness) Let $N = (P, T, F, \alpha, m_i, m_f)$ be a Petri net over a set of activities A. N is *easy sound* if and only if there exists a firing sequence $\varrho \in T^*$ such that $m_i \xrightarrow{\varrho} m_f$.

Take for example the trace and model in Figure 4. The model shows an online transaction process for an electronic bookstore similar to the one shown in Figure 1. The model consists of 6 transitions. Transition t_4 is invisible as activity *edit order* is not logged.

Figure 5 shows two possible alignments between the trace and model in Figure 4. For each alignment, the sequence of activities on the top row yields the original trace, while the sequence of transitions on the bottom row (i.e. $\langle t_1, t_2, t_3, t_5 \rangle$ for γ_1 and $\langle t_1, t_2, t_3, t_4, t_3, t_5 \rangle$ for γ_2) is a complete firing sequence of the model shown in Figure 4.

In practice, severity of deviations may differ between different activities. For example in Figure 5, one may consider move on model on t_4 (*edit order*) in γ_2

to be less severe than move on model on t_5 (*deliver*), because activity *edit order* is not logged while activity *deliver* is logged whenever it occurs. To measure the quality of alignments, we assign costs to each movement (i.e. synchronous move, move on model, and move on log). Given a trace a model, we are interested in alignments with the least total cost according to the assigned cost function. Such an alignment is called an *optimal alignment*.

Definition 3.4. (Optimal alignment) Let $\sigma \in A^*$ be a trace over a set of activities A and let $N = (P, T, F, \alpha, m_i, m_f)$ be an easy sound Petri net over A. Let $\kappa : A^{\gg} \times T^{\gg} \rightarrow \mathbb{R}^+$ be a cost function for movements. We say that $\gamma \in \Gamma_{\sigma,N}$ is an *optimal alignment* between σ and N if and only if for all $\gamma' \in \Gamma_{\sigma,N}, \kappa(\gamma) \leq \kappa(\gamma')$. $\Gamma^{\kappa}_{\sigma,N}$ is the set of all optimal alignments between σ and N with respect to cost function κ. Note that $\kappa(\gamma) = \sum_{(a.t)\in\gamma} \kappa(a,t)$

The cost function provides a certain level of flexibility to determine the desired optimal alignment between a given trace and model. In this paper, we are interested in alignments to identify deviations between the trace and the model. Thus, we do not penalize synchronous moves and moves on model of invisible transitions because their absence cannot be observed from traces. However, as discussed in Section 2, the $\mathcal{A}^\star$ algorithm requires graph with positive non-zero distance between nodes. Therefore, we define a *standard cost function* that assigns a negligibly small cost $\epsilon \in \mathbb{R}^+$ to all synchronous moves and moves on model of invisible transitions. Furthermore, the function assign cost $1 + \epsilon$ to all moves on log/moves on model of normal (not invisible) transitions. The choice for $\epsilon > 0$ depends on the specific log and model. As a guideline, one could use $\epsilon = \frac{highest\ cost}{(lowest\ cost)\cdot(size\ of\ longest\ trace)}$. However, ϵ should always be significantly smaller than all other costs. If chosen ϵ is too small, finding an optimal alignment may take longer. After getting an optimal alignment γ, $\epsilon \cdot |\gamma|$ can be substracted from the total cost of γ to get the "real" total cost of γ. In the remainder sections, we use the standard cost function unless indicated otherwise.

Take for example the alignments in Figure 5. Using the standard cost function, the total costs of alignment γ_1 and γ_2 are $(5\epsilon + 2)$ and $(6\epsilon + 1)$ respectively. After substracting the cost with their extra costs (i.e. due to addition of ϵ), we obtain the "real" total cost of deviation in γ_1 and γ_2 as 2 and 1 respectively. the total cost of γ_2 is less than γ_1, γ_2 is a better alignment than γ_1. Furthermore, there is no other alignment with less total cost than γ_2 for the trace and the model in Figure 4. Hence, γ_2 is an optimal alignment.

Computing optimal alignments between traces and process models is computationally expensive. Cook and Wolf [7] proposed a tree-based state space exploration method to calculate optimal alignments. This approach is improved by Adriansyah et al. [4] by modeling state spaces as connected graphs instead of trees to prune the search space and use the $\mathcal{A}^\star$ algorithm [13] to compute optimal alignments. The $\mathcal{A}^\star$ algorithm requires the least number of visited states among all shortest path algorithms to find a shortest path from a source node to a target node, as long as it uses a permissible underestimation function that provides an underestimate for the distance from each node to the target node [9]. Given a trace and a process model, the approach in [4] uses a permissible

underestimation function that always returns the costs of synchronous moves for replaying the remainder of traces. However, this function performs poorly as the $\mathcal{A}^\star$ algorithm works in a breadth first search manner. A poor estimation increases the number of states required to create an alignment and severely limits the applicability of the approach. To overcome this problem, we use the marking equation of Petri nets to further prune the search space by providing a better estimate for the lower bound on the remaining distance. Section 4 explains the translation from the problem of computing optimal alignments to shortest path problems which underlies our approach.

4 Optimal Alignments Correspond to Shortest Paths

An alignment between a given trace and an easy sound Petri net is constructed by performing a sequence of movements that changes the "state" of the trace, state of the model, or both. We define the problem of constructing an optimal alignment by explicitly modeling the change of states in both the trace and the model. We use the trace and model in Figure 4 as a running example.

We model traces as *event nets* [2] such that the state of a trace is captured explicitly by its marking. The event net of a trace is a Petri net with a linear structure, such that each transition in the net represents a unique activity occurrence in the trace. The net only has one complete firing sequence: the one that follows the order of the activities in the trace. Figure 6 shows the event net of the trace $\langle add\ items, add\ items, finalize, finalize\rangle$.

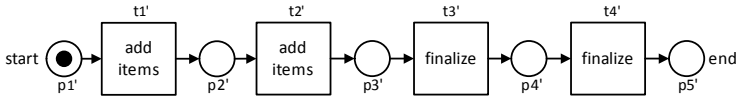

Fig. 6. The event net of trace $\sigma = \langle add\ items, add\ items, finalize, finalize\rangle$

Formally, we define event net of a trace as follows:

Definition 4.1. (Event net) Let $\sigma \in A^*$ be a trace of length n over a set of activities A. The *event net* of σ is a Petri net $N = (P, T, F, \alpha, m_i, m_f)$, where

- $P = \{p_j \mid 1 \leq j \leq n+1\}$,
- $T = \{t_j \mid 1 \leq j \leq n\}$,
- $F : (P \times T) \cup (T \times P) \to \mathbb{N}$, such that
  - $F(p_j, t_j) = 1$, for all $1 \leq j \leq n, p_j \in P, t_j \in T$,
  - $F(t_j, p_{j+1}) = 1$, for all $1 \leq j \leq n, p_j \in P, t_j \in T$, and
  - $F(x, y) = 0$ otherwise.

- $\alpha : T \rightarrow A$ is a function mapping transitions to activities such that for all $1 \leq j \leq n, \alpha(t_j) = \sigma[j]$,
- $m_i : P \rightarrow \mathbb{N}$ is the initial marking, such that for $p_1 \in P, m_i(p_1) = 1$ and for all other $p' \in \{p_2, \ldots, p_{n+1}\}, m_i(p') = 0$,
- $m_f : P \rightarrow \mathbb{N}$ is the final marking, such that for $p_{|P|} \in P, m_f(p_{|P|}) = 1$ and for all other $p' \in \{p_1, \ldots, p_n\}, m_f(p') = 0$,

Note that by definition, event nets are easy sound net. To prevent ambiguity between event nets and the easy sound Petri net that represents the process model, from this section on we refer to the latter the *process net*.

Having modeled traces as event nets, we explicitly model all possible movements by taking the *product* of two Petri nets: event nets and process nets. The product of two Petri nets is the union of both nets with extra *synchronous transitions* that are constructed by pairing transitions in one net with transitions in the other net that have the same label [39]. This way, possible synchronous moves are modeled by synchronous transitions, while moves on log and moves on model are modeled as unpaired transitions in the event net and process net respectively. Note that all event nets are easy sound nets.

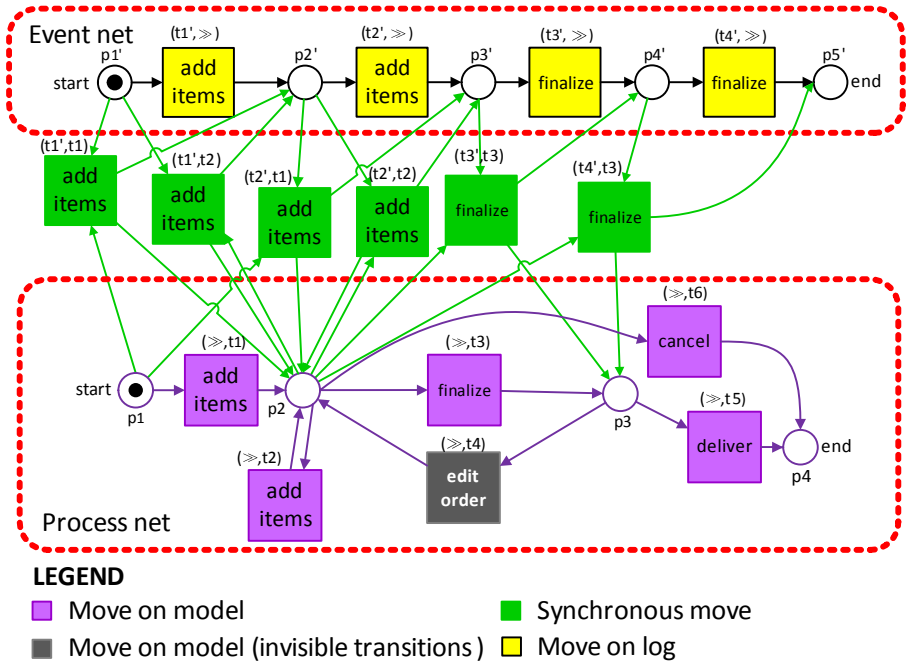

Fig. 7. Product of the event net of trace $\sigma = \langle add\ items, add\ items, finalize, finalize \rangle$ and the process net in Figure 4

For example, the product of the event net in Figure 6 and the easy sound process net in Figure 4 is shown in Figure 7. Recall that the product is derived from the trace and the model shown previously in Figure 4. The color of transitions and places distinguishes elements of the original nets and the added synchronous transitions (colored green). Yellow, purple, gray, and green-colored transitions represent moves on log, moves on model (normal transitions), moves on model (invisible transitions), and synchronous moves respectively. If a yellow transition is fired, the state of the event net is changed but not the state of the process net. Similarly, firing a purple/gray transition only changes the state of the process net. Firing a green transition changes the state of both nets. All transitions in the net represents movements. For example, the transition (t'_1, t_1) represents a synchronous move by doing the first activity *add items* in the trace and firing transition t_1 in the process net. As another example, the transition $(\gg, t_1)$ represents a move on model by firing the transition t_1 in the process net without moving in the trace.

The product of two Petri nets is formalized as follows [39].

Definition 4.2. (Product of Two Petri Nets)
Let $N_1 = (P_1, T_1, F_1, \alpha_1, m_{i,1}, m_{f,1})$ and $N_2 = (P_2, T_2, F_2, \alpha_2, m_{i,2}, m_{f,2})$ be two Petri nets over a set of activities A. The *product* of N_1 and N_2 is the Petri net $N_3 = N_1 \times N_2 = (P_3, T_3, F_3, \alpha_3, m_{i,3}, m_{f,3})$ where

- $P_3 = P_1 \cup P_2$ is the union set of places,
- $T_3 \subseteq (T_1^{\gg} \times T_2^{\gg})$, such that $T_3 = \{(t_1, \gg) \mid t_1 \in T_1\} \cup \{(\gg, t_2) \mid t_2 \in T_2\} \cup \{(t_1, t_2) \in \mathrm{Dom}(\alpha_1) \times \mathrm{Dom}(\alpha_2) \mid \alpha_1(t_1) = \alpha_2(t_2)\}$ is the set of the original transitions with additional *synchronous transitions*,
- $F_3 : (P_3 \times T_3) \cup (T_3 \times P_3) \to \mathbb{N}$ is the arc weight function, such that
  - $F_3((p_1, (t_1, \gg))) = F_1(p_1, t_1)$ if $p_1 \in P_1$ and $t_1 \in T_1$,
  - $F_3(((t_1, \gg), p_1)) = F_1(t_1, p_1)$ if $p_1 \in P_1$ and $t_1 \in T_1$,
  - $F_3((p_2, (\gg, t_2))) = F_2(p_2, t_2)$ if $p_2 \in P_2$ and $t_2 \in T_2$,
  - $F_3(((\gg, t_2), p_2)) = F_2(t_2, p_2)$ if $p_2 \in P_2$ and $t_2 \in T_2$,
  - $F_3(p_1, (t_1, t_2)) = F_1(p, t_1)$ if $p \in P_1$ and $(t_1, t_2) \in T_3 \cap T_1 \times T_2$,
  - $F_3(p_2, (t_1, t_2)) = F_2(p, t_2)$ if $p \in P_2$ and $(t_1, t_2) \in T_3 \cap T_1 \times T_2$,
  - $F_3((t_1, t_2), p_1) = F_1(t_1, p)$ if $p \in P_1$ and $(t_1, t_2) \in T_3 \cap T_1 \times T_2$,
  - $F_3((t_1, t_2), p_2) = F_2(t_2, p)$ if $p \in P_2$ and $(t_1, t_2) \in T_3 \cap T_1 \times T_2$,
  - otherwise $F_3(x, y) = 0$.
- $\alpha_3 : T_3 \not\to A$ is the partial mapping from transitions to activities, such that for all $(t_1, t_2) \in T_3$, $\alpha_3((t_1, t_2)) = \alpha(t_1)$ if $t_1 \in \mathrm{Dom}(\alpha_1)$ and $\alpha_3((t_1, t_2)) = \alpha_2(t_2)$ if $t_2 \in \mathrm{Dom}(\alpha_2)$,
- $m_{i,3} : P_3 \to \mathbb{N}$ is the initial marking, such that $\forall_{p_1 \in P_1}\ m_{i,3}(p_1) = m_{i,1}(p_1)$ and $\forall_{p_2 \in P_2}\ m_{i,3}(p_2) = m_{i,2}(p_2)$,

– $m_{f,3} : P_3 \rightarrow \mathbb{N}$ is the final marking such that $\forall_{p_1 \in P_1}\ m_{f,3}(p_1) = m_{f,1}(p_1)$ and $\forall_{p_2 \in P_2}\ m_{f,3}(p_2) = m_{f,2}(p_2)$,

Since transitions in all products of event nets and easy sound process nets represent movements, firing sequences in such products yield movement sequences. Take for example a firing sequence $\langle (t'_1, t_1), (t'_2, t_2), (t'_3, t_3), (t'_4, \gg), (\gg, t_5) \rangle$ of the net shown in Figure 7. The sequence shows three synchronous moves, followed by a move on log and a move on model. By mapping each transition in the sequence back to its movement (i.e. pair of activities and transitions), we obtain sequence of movements γ_1 in Figure 5. Note that such a sequence of movements is by definition an alignment.

We define a reverse function that maps transitions in a product of two Petri nets to movements as follows:

Definition 1 (Reverse Function). *Let $N_1 = (P_1, T_1, F_1, \alpha_1, m_{i,1}, m_{f,1})$ and $N_2 = (P_2, T_2, F_2, \alpha_2, m_{i,2}, m_{f,2})$ be two Petri nets over A, and let $N_1 \times N_2 = (P_3, T_3, F_3, \alpha_3, m_{i,3}, m_{f,3})$ be the product of N_1 and N_2.*

$rev_{(N_1 \times N_2)} : T_3 \not\rightarrow A^{\gg} \times T^{\gg}$ is the partial reverse function *of $N_1 \times N_2$ that maps some transitions in T_3 to movements, such that for all $(t_1, t_2) \in T_3$,*

– *$rev_{(N_1 \times N_2)}((t_1, t_2)) = (\alpha_1(t_1), \gg)$ if $t_2 = \gg$ and $t_1 \in Dom(\alpha_1)$, i.e. all transitions derived from only transitions of T_1 (except invisible transitions) are mapped to moves on log,*
– *$rev_{(N_1 \times N_2)}((t_1, t_2)) = (\gg, t_2)$ if $t_1 = \gg$, i.e. transitions derived from only transitions of T_2 are mapped to moves on model, and*
– *$rev_{(N_1 \times N_2)}((t_1, t_2)) = (\alpha(t_1), t_2)$ if $t_1 \in Dom(\alpha_1)$ and $t_2 \in Dom(\alpha_2)$ and $\alpha(t_1) = \alpha(t_2)$, i.e. synchronous transitions are mapped to synchronous moves.*

We use the reverse function and the theory of marking reachability in product of two Petri nets in [39] to prove that complete firing sequences of products of event nets and easy sound process nets yield alignments. We can reformulate the result of [39] as follows:

Theorem 1 (Reachability of Marking in Product of Two Petri Nets). *Let $N_1 = (P_1, T_1, F_1, \alpha_1, m_{i,1}, m_{f,1})$ and $N_2 = (P_2, T_2, F_2, \alpha_2, m_{i,2}, m_{f,2})$ be two Petri nets over a set of activities A. Let $N_1 \times N_2 = (P_3, T_3, F_3, \alpha_3, m_{i,3}, m_{f,3})$ be the product of N_1 and N_2. For all firing sequences $m_{i,3} \xrightarrow{\varrho} m, \varrho \in T_3^*$, both $m_{i,3}{\downarrow_{P_1}} \xrightarrow{\pi_1(\varrho)_{\downarrow T_1}} m_{\downarrow P_1}$ and $m_{i,3}{\downarrow_{P_2}} \xrightarrow{\pi_2(\varrho)_{\downarrow T_2}} m_{\downarrow P_2}$ hold, i.e. the projection of the marking to each original net is also reachable from its initial marking.*

Proof. Refer to [39].

Using this result we can prove the following theorem:

Theorem 2 (Firing Sequences Define Movement Sequences). *Let $\sigma \in A^*$ be a trace over a set of activities A and let $N_1 = (P_1, T_1, F_1, \alpha_1, m_{i,1}, m_{f,1})$ be its event net. Let $N_2 = (P_2, T_2, F_2, \alpha_2, m_{i,2}, m_{f,2})$ be a process net over A, and let $N_1 \times N_2 = (P_3, T_3, F_3, \alpha_3, m_{i,3}, m_{f,3})$ be the product of N_1 and N_2.*

For all firing sequences $m_{i,3} \xrightarrow{\varrho} m, \varrho \in T_3^*$, $rev_{(N_1 \times N_2)}(\varrho)$ *is a movement sequence*[1]. *Furthermore, if* $m_{i,3} \xrightarrow{\varrho} m_{f,3}$, *then* $rev_{(N_1 \times N_2)}(\varrho)$ *is an alignment.*

Proof. We prove the first part of this lemma by induction. If $\varrho = \langle\rangle$, then $rev_{(N_1 \times N_2)}(\varrho) = \langle\rangle$ is an alignment. Assume that $\varrho = \varrho' \cdot (t_1, t_2)$ and that $rev_{(N_1 \times N_2)}(\varrho)$ is a movement sequence. There exist markings m_1 and m_2 where $m_{i,3} \xrightarrow{\varrho'} m_1 \xrightarrow{(t_1,t_2)} m_2$. We show that $rev_{(N_1 \times N_2)}(\varrho)$ is a movement sequence by considering all three cases:

- Assume $rev_{(N_1 \times N_2)}((t_1, t_2)) = (a, \gg), a \in A$ (move on log). By definition, $\mathrm{Dom}(\alpha_1) = T_1$. Thus, there exists a transition $t_1 \in T_1$ where $\alpha_1(t_1) = \sigma[1 + |\pi_1(\varrho')_{\downarrow T_1}|] = a$ and $m_{i,3\downarrow P_1} \xrightarrow{\pi_1(\varrho')_{\downarrow T_1}} m_{1\downarrow P_1} \xrightarrow{t_1} m_{2\downarrow P_1}$. Furthermore, $m_{i,3\downarrow P_2} \xrightarrow{\pi_2(\varrho)_{\downarrow T_2}} m_{1\downarrow P_2}$. It is easy to see that $rev_{(N_1 \times N_2)}(\varrho)$ is a movement sequence,
- Assume $rev_{(N_1 \times N_2)}((t_1, t_2)) = (\gg, t_2), t_2 \in T_2$ (move on model). We know that $m_{i,3\downarrow P_1} \xrightarrow{\pi_1(\varrho')_{\downarrow T_1}} m_{1\downarrow P_1}$ and $m_{i,3\downarrow P_2} \xrightarrow{\pi_2(\varrho')_{\downarrow T_2}} m_{1\downarrow P_2} \xrightarrow{t_2} m_{2\downarrow P_2}$ (see Theorem 2). It is easy to see that $rev_{(N_1 \times N_2)}(\varrho)$ is a movement sequence,
- Assume $rev_{(N_1 \times N_2)}((t_1, t_2)) = (a, t_2), a \in A, t_2 \in T_2$ (synchronous move). We know that $m_{i,3\downarrow P_2} \xrightarrow{\pi_2(\varrho')_{\downarrow T_2}} m_{1\downarrow P_2} \xrightarrow{t_2} m_{2\downarrow P_2}$. By definition, $t_1 \in \mathrm{Dom}(\alpha_1)$ such that $\alpha(t_1) = \sigma[1 + |\pi_1(\varrho')_{\downarrow T_1}|] = a$ and $m_{i,3\downarrow P_1} \xrightarrow{\pi_1(\varrho')_{\downarrow T_1}} m_{1\downarrow P_1} \xrightarrow{t_1} m_{2\downarrow P_1}$. It is easy to see that $rev_{(N_1 \times N_2)}(\varrho)$ is a movement sequence.

We know that $rev_{(N_1 \times N_2)}(\varrho)$ is a movement sequence. If $m_{i,3} \xrightarrow{\varrho} m_{f,3}$ then $m_{i,3\downarrow P_1} \xrightarrow{\pi_1(\varrho)_{\downarrow T_1}} m_{f,3\downarrow P_1}$ and $m_{i,3\downarrow P_2} \xrightarrow{\pi_2(\varrho)_{\downarrow T_2}} m_{f,3\downarrow P_2}$. By definition, this implies $m_{i,1} \xrightarrow{\pi_1(\varrho)_{\downarrow T_1}} m_{f,1}$ and $m_{i,2} \xrightarrow{\pi_2(\varrho)_{\downarrow T_2}} m_{f,2}$. Since $\alpha_1(\pi_1(\varrho)_{\downarrow T_1}) = \sigma$, we show that $rev_{(N_1 \times N_2)}(\varrho)$ is an alignment. □

To find an optimal alignment between a trace and a Petri net using the $\mathcal{A}^\star$ algorithm, we use the transition system of the product between two nets: the event net of the trace and the original net. Take for example the transition system of the net in Figure 7, shown in Figure 8. Arcs between states are labeled with transitions, and the color of each arc indicates the movement of its transition label. The transition labels are abbreviated with their initials. We define the distance of an arc in the cost of movement of its label. In this running example, we use the standard cost function previously defined in the end of Section 3.

By definition, paths from the initial state of a transition system to its final state yield complete firing sequences. Theorem 2 shows that complete firing

[1] We abuse the reverse function to handle sequences of transitions. Let $\varrho \in T^*$ be a sequence of transitions, $rev_{(N_1 \times N_2)}(\varrho) = \langle rev_{(N_1 \times N_2)}(\varrho[1]), \ldots, rev_{(N_1 \times N_2)}(\varrho[\varrho])\rangle$

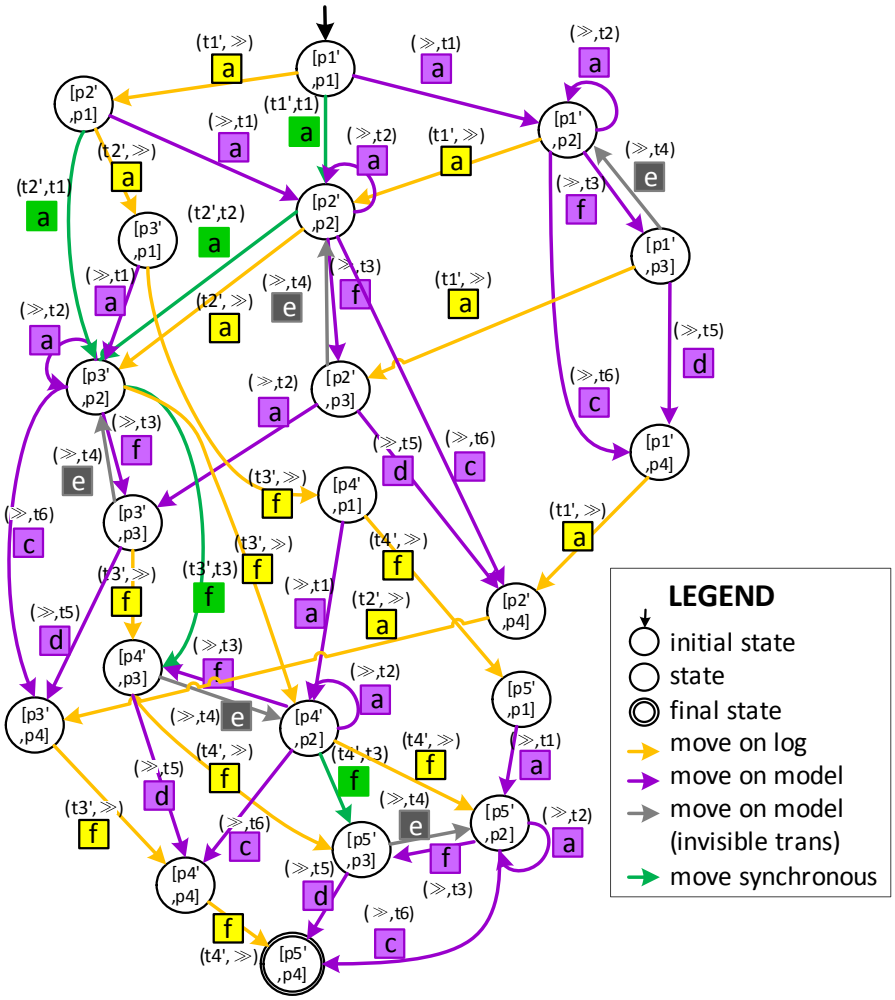

Fig. 8. Transition System of the Petri net in Figure 7

sequences yield alignments. Transitively, such paths yield alignments. Furthermore, since arc distances are derived from movement costs, distances of the paths yield cost of alignments.

Take for example the path $\sigma = \langle (t'_1, t_1), (t'_2, t_2), (t'_3, t_3), (t'_4, \gg), (\gg, t_5) \rangle$ from the initial state to the final state of the transition system in Figure 8. σ is also a complete firing sequence of the net in Figure 7. Each transition in σ can be mapped back to its movement, e.g. (t'_1, t_1) is mapped to a synchronous move between activity *add items* and transition t_1, $(t'_4, \gg)$ is mapped to a move on log of activity *finalize*. Such mapping yields an alignment whose total cost is the same as its path distance. In this example, mapping back all transitions in σ

yields alignment γ_1 in Figure 5. Notice that the total distance of σ is the same as the total cost of γ_1 (i.e. $\epsilon+\epsilon+\epsilon+1+\epsilon+1+\epsilon = 5\epsilon+2$). Therefore, a shortest path from the initial state to the final state of the transition system yields an optimal alignment. The formal proof is as follows:

Lemma 1 (Paths Yield Movement Sequences). *Let $\sigma \in A^*$ be a trace over A. Let N_1 be the event net of σ and let N_2 be a process net over A. Let $TS(N_1 \times N_2) = (S, E, T, \alpha, s_i, s_f)$ be the transition system of $N_1 \times N_2$.*

For all states $s \in S$, let $\varrho \in \Psi_{(S,E,T)}(s_i, s)$ be a path from s_i to s. $\gamma = rev_{(N_1 \times N_2)}(\pi_2(\varrho))$ is a movement sequence. Furthermore, if $s = s_f$, then γ is an alignment between σ and N_2. □

Proof. By definition, all paths from the initial state s_i are firing sequences. Theorem 2 shows that all firing sequences yield movement sequences, and all complete firing sequences yield alignments. Transitively, all paths from the initial state s_i yield movement sequences, and all paths from the initial state s_i to the final state s_f yield alignments.

We translate the problem of finding optimal alignments to a problem of finding a shortest path in a directed graph. In the following, we show that such a path always exists.

Lemma 2 (Path from the Initial State to the Final State Exists). *Let $N_1 = (P_1, T_1, F_1, \alpha_1, m_{i,1}, m_{f,1})$ and $N_2 = (P_2, T_2, F_2, \alpha_2, m_{i,2}, m_{f,2})$ be an event net and an easy sound process net over a set of activities A respectively. Let $TS(N_1 \times N_2) = (S_3, E_3, T_3, \alpha_3, s_{i,3}, s_{f,3})$ be the transition system of $N_1 \times N_2$.*

$\Psi_{(S_3,E_3,T_3)}(s_{i,3}, s_{f,3}) \neq \emptyset$, i.e. the path from the initial state $s_{i,3}$ to the final state $s_{f,3}$ of $TS(N_1 \times N_2)$ always exists.

Proof. From Definition 4.1, we know that there exists $\varrho_1 \in T_1^*$, such that $m_{i,1} \xrightarrow{\varrho_1} m_{f,1}$. From Definition 3.3, we know that there exists a sequence $\varrho_2 \in T_2^*$ such that $m_{i,2} \xrightarrow{\varrho_2} m_{f,2}$. Since markings $m_{f,1}$ and $m_{f,2}$ are reachable from $m_{i,1}$ and $m_{i,2}$ respectively, there exists $\varrho_3 \in T_3^*$ such that $s_{i,3} \xrightarrow{\varrho_3} s_{f,3}$ (see Theorem 1), thus $\Psi_{(S_3,E_3,T_3)}(s_{i,3}, s_{f,3}) \neq \emptyset$. □

Intuitively, there are many trivial alignments between a trace and a model. It is possible to first do move on model only and then do move on log only for all activities in the trace, as well as all interleavings between its moves on model and moves on log. Typically, such trivial alignments do not correspond to an optimal alignment. In Lemma 1, we showed that paths in transition systems yield alignments and such path always exists (see Lemma 2). By associating distances of arcs in the transition systems according to cost of movements, we convert the problem of finding an optimal alignment into the problem of finding a shortest path.

Theorem 3 (Shortest Path Yields Optimal Alignment). *Let N_1 be an event net over A, let N_2 be an easy sound process net over A. Let $TS(N_1 \times N_2) =$*

$(S_3, E_3, T_3, \alpha_3, s_{i,3}, s_{f,3})$ be the transition system of $N_1 \times N_2$. Let $\kappa : A^{\gg} \times T^{\gg} \rightarrow \mathbb{R}^+$ be a cost function of movements. Let $\delta : E_3 \rightarrow \mathbb{R}^+$ be a distance function, such that for all $e \in E_3, \delta(e) = \kappa(rev_{(N_1 \times N_2)}(\pi_2(e)))$.

For all shortest paths $\varrho \in \Psi_{(S_3,E_3,T_3)}(s_{i,3}, s_{f,3})$ from the initial state to the final state such that for all $\varrho' \in \Psi_{(S_3,E_3,T_3)}(s_{i,3}, s_{f,3}), \delta(\varrho) \leq \delta(\varrho')$, the sequence of movements $\gamma = rev_{(N_1 \times N_2)}(\pi_2(\varrho))$ is an optimal alignment.

Proof. Lemma 1 proves that γ is an alignment. By the definition, $\delta(\varrho) = \sum_{1 \leq j \leq |\varrho|} \delta(\varrho[j]) = \sum_{1 \leq j \leq |\varrho|} \kappa(rev_{(N_1 \times N_2)}(\pi_2(\varrho[j]))) = \sum_{1 \leq h \leq |\gamma|} \kappa(\gamma[h])$, i.e. the distance between two nodes is the same as the cost of movements. Since ϱ is a shortest path from s_i to s_f, its distance yields the minimal cost of movements of all possible alignments. □

Consider again our previous example of the transition system of Figure 8. Figure 9 shows a shortest path from the initial state to the final state of the transition system with such a cost function. The path consists of three consecutive synchronous transitions (i.e. *add items,add items,finalize*) followed by a move on model of an invisible transition (*edit order*), another synchronous transition (*finalize*), and a move on model (*deliver*). By mapping each transition back to its movement, the alignment constructed from the path is shown in Figure 10. The alignment shown in Figure 10 is an optimal alignment.

Given a directed graph in form of a transition system, we use the $\mathcal{A}^\star$ algorithm [13] to find a shortest path from the initial state to a final state efficiently as mentioned in Section 1. This approach is explained in Section 5.

5 Efficient Search Space Exploration using Marking Equation

In Section 4, we translated the problem of construcing an alignment between a trace and a model to the problem of finding a shortest path between two nodes in a transition system. However, finding such a path in a transition system is not trivial. First of all, process models may have an unbounded number of states. Second, in reality, trace executions often deviate from predefined process models. *In such cases, using a breadth-first-search approach to find shortest paths requires a huge amount of memory because many states need to be visited and queued.*

Therefore, we introduce a permissible underestimation function for the $\mathcal{A}^\star$ algorithm that provides a good underestimation of the remaining distance from each state in the system to its final state based on the marking equation of Petri nets. The function improves the efficiency of the $\mathcal{A}^\star$ based state space exploration in two ways (see Figure 11). First of all, the *state space is pruned* as for some states we can state with certainty that the final state is no longer reachable. If, according to the marking equation, the final state is no longer reachable, we do not need to explore successor states in the transition system. Second, by using the marking equation we can provide a better underestimation of the total cost required to reach the final state. This way state space exploration can be limited to those states that most likely lead to the final state.

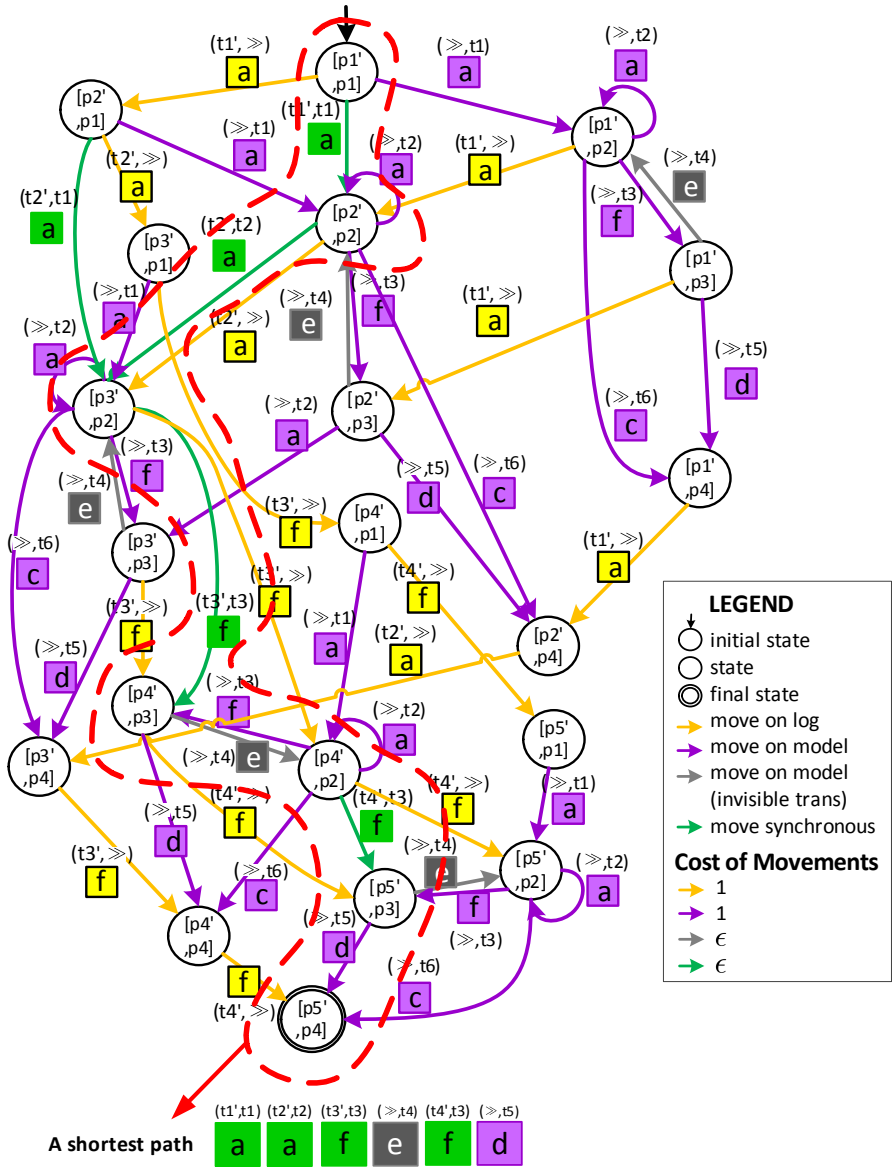

Fig. 9. A shortest path in the transition system in Figure 7 yields an optimal alignment

5.1 Pruning the Exploration Graph

To prune the exploration graph we exploit the well known theory of the *Petri net marking equation*. The following result can be found in any textbook on Petri nets, for example [18].

add items	*add items*	*finalize*	≫	*finalize*	≫
add items	*add items*	*finalize*	*edit order*	*finalize*	*deliver*
t_1	t_2	t_3	t_4	t_3	t_5

Fig. 10. An optimal alignment between trace $\langle$*add items*, *add items*, *finalize*, *finalize*$\rangle$ and the net in Figure 4

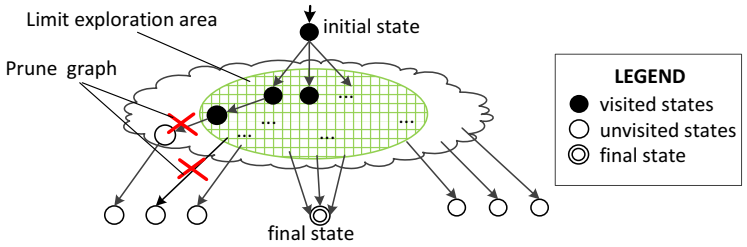

Fig. 11. Two utilizations of marking equation: pruning exploration graph and limit exploration area

Theorem 4 (Reachability Implies Solution to Marking Equation). *Let N be a Petri net over a set of activities A, let $\mathbf{N}$ be the incidence matrix of N. Let $TS(N) = (S, E, T, \alpha, s_i, s_f)$ be the transition system of N. For all $s, s' \in S, \varrho \in T^*$ such that $s \xrightarrow{\varrho} s'$, the following marking equation holds: $\vec{s} + \mathbf{N} \cdot \vec{\varrho} = \vec{s'}$*

Proof. See for example the proof in [18].

Theorem 4 states that reachability implies a solution. This implies that if no solution exists then one state is not reachable from the other. Recall that we need to find a shortest path to the final node and all paths are firing sequences. Hence, we exploit this knowledge to identify nodes in the exploration graph from which the target node cannot be reached.

Theorem 5 (State Pruning). *Let N be a Petri net over a set of activities A. Let $\mathbf{N}$ be the incidence matrix of N. Let $TS(N) = (S, E, T, \alpha, s_i, s_f)$ be the transition system of N. For all $s \in S$, if there is no solution $\vec{x}$ to $\vec{s} + \mathbf{N} \cdot \vec{x} = \vec{s_f}$, $\Psi_{(S,E,T)}(s, s_f) = \emptyset$.*

Proof. We prove this theorem by contradiction. Suppose that there exists a state $s' \in S$ such that no solution to $\vec{s'} + \mathbf{N} \cdot \vec{x} = \vec{s_f}$, but there exists a path $\sigma \in \Psi_{(S,E,T)}(s', s_f)$ from state s' to final state s_f. Let $\varrho = \pi_2(\sigma)_{\downarrow T}$ be the firing sequence of σ. By definition, we know that $s \xrightarrow{\varrho} s_f$ and according to Theorem 4, $\vec{s} + \mathbf{N} \cdot \vec{\varrho} = \vec{s_f}$ holds. Thus, $\vec{x} = \vec{\varrho}$ is a solution. □

The marking equation helps in pruning the exploration graph. However, we can also use the marking equation to provide a better underestimate of the remaining cost.

5.2 Limiting the State Space Exploration Area

We showed that the existence of a solution to the marking equation for a state in Theorem 4 strongly correlates with the existence of path from the state to the final state of corresponding transition system. Next, we show that if such path exists, a solution to the marking equation with the minimum total cost yields a lower bound for the path distance. Since all transitions in the product of event nets and process nets represent movements, we define cost of transitions in such a product based on cost of movements.

Definition 5.1. (Cost of Transitions) Let N_1 be an event net over a set of activities A and let N_2 be a process net over A. Let $N_1 \times N_2 = (P_3, T_3, F_3, \alpha_3, m_{i,3}, m_{f,3})$ be the product of N_1 and N_2. Let $\kappa : A^{\gg} \times T^{\gg} \to \mathbb{R}^+$ be a cost function of movements. $c_\kappa : T_3 \to \mathbb{R}^+$ is the *cost of firing transitions*, defined by κ such that for all $t_3 \in T_3, c_\kappa(t_3) = \kappa(rev_{(N_1 \times N_2)}(t_3))$.

For simplicity, in the remainder we use the cost of firing transitions directly instead of the cost of movements. Next, we show that marking equation provides lower bound for the total cost of movements.

Theorem 6 (Marking Equation Solution provides Lower Bound). *Let N be the product of an event net and an easy sound process net over a set of activities A. Let $\mathbf{N}$ be the incidence matrix of N. Let $TS(N) = (S, E, T, \alpha, s_i, s_f)$ be the transition system of N. Let $c_\kappa : T \to \mathbb{R}^+$ be a cost function of firing transitions, and let $\delta : E \to \mathbb{R}^+$ be a distance function, such that for all $e \in E, \delta(e) = c_\kappa(\pi_2(e))$.*

For all states $s \in S$ where $\Psi_{(S,E,T)}(s, s_f) \neq \emptyset$

- *let $v = \min(\{\sum_{1 \leq j \leq |T|} \vec{x}_j \cdot c_\kappa(t_j) \mid t_j \in T \wedge \vec{s} + \mathbf{N} \cdot \vec{x} = \vec{s_f}\})$ be the minimum value of total cost of solution to marking equation, and*
- *let $v' = \min(\{\delta(\varrho) \mid \varrho \in \Psi_{(S,E,T)}(s, s_f)\})$ be the minimum distance of all paths from s to s_f.*

The following statement holds: $v \leq v'$

Proof. We prove this theorem by first showing that a shortest path from s to s_f is also a solution to the marking equation. Suppose that $\sigma \in \Psi_{(S,E,T)}(s, s_f)$ such that for all $\sigma' \in \Psi_{(S,E,T)}(s, s_f)$, $\delta(\sigma) \leq \delta(\sigma')$, i.e. σ is a shortest path. From Lemma 1, we know that $\varrho = \pi_2(\sigma)$ is a firing sequence from s to s_f, i.e. $s \xrightarrow{\varrho} s_f$. Hence, ϱ yields a solution for marking equation $\vec{s} + \mathbf{N} \cdot \vec{\varrho} = \vec{s_f}$.

The total cost of solution to marking equation is the same as the distance of path (see the proof of Theorem 3). Therefore, $\delta(\sigma) = \sum_{1 \leq j \leq |\varrho|} c_\kappa(\varrho[j])$. Since $\varrho \in T^*$, we can reformulate the formula into $\sum_{1 \leq k \leq |T|} \varrho(t_k) \cdot c_\kappa(t_k)$ where $t_k \in T$, which is the same as the defined cost of solution to marking equation. □

Given a state in a transition system, we know from Theorem 5 that if a solution to marking equation in Theorem 4 does not exist, there is no path to the target node. Hence, there is no point to explore the successors of that node

anymore. If a solution exists, we know a lower bound for the distance from the node to the target node (see Theorem 6). We use this knowledge to define a permissible underestimation function for the $\mathcal{A}^\star$ algorithm.

Theorem 7 (Permissible Underestimation Function).
Let N be the product of an event net and an easy sound process net over a set of activities A. Let $\mathbf{N}$ be the incidence matrix of N. Let $TS(N) = (S, E, T, s_i, s_f)$ be the transition system of N. Let $c_\kappa : T \rightarrow \mathbb{R}^+$ be a cost function of firing transitions. Let $\delta : E \rightarrow \mathbb{R}^+$ be a distance function, such that for all $e \in E, \delta(e) = c_\kappa(\pi_2(e))$.

For all $s \in S$, a function $h : S \rightarrow \mathbb{R}$ where,

- *$h(s) = +\infty$ if there is no solution to $\vec{s} + \mathbf{N} \cdot \vec{x} = \vec{s_f}$, otherwise*
- *$h(s) = \min\left(\{\sum_{1 \leq j \leq \lvert T \rvert} \vec{x}_j \cdot c_\kappa(t_j) \mid t_j \in T \wedge \vec{s} + \mathbf{N} \cdot \vec{x} = \vec{s_f}\}\right)$*

is a permissible underestimation function. □

Proof. For all $s \in S$, if there is no solution to the marking equation, according to Theorem 5 there is no path from s to s_f. Therefore, $h(s) = +\infty$ satisfies the requirements for a permissible underestimation function. If there is a solution, we know that $h(s)$ is a lower bound for the distance from s to s_f (Theorem 6). Thus, $h(s)$ is a permissible underestimation function.

Finding a solution for the marking equation with the minimum cost can viewed as an ILP problem [8]. Many approaches to solve such a problem exist in literature. Given a trace and a process net, we use the permissible underestimation function in Theorem 7 to guide state space exploration on transition systems constructed from product of event nets and process nets, such as the one shown in Figure 8.

We extend this ILP-based approach to Petri nets with reset/inhibitor arcs [10,23,36]. Section 6 explains the extension in detail.

6 Extension to Reset/Inhibitor Nets

Next to Petri nets, many process modeling languages are used in practice. In this section, we extend the approach in Section 5 to deal with models in the form of reset/inhibitor nets [36]. Reset/inhibitor nets are Petri nets that are extended with reset arcs and/or inhibitor arcs. The addition of reset/inhibitor arcs allows modelers to express complex behavior that cannot be expressed in ordinary Petri nets. There are many approaches in literatures that translate various modeling languages to reset/inhibitor nets (e.g. [6,33,14,15,21]). Thus, by extending the approach to handle reset/inhibitor nets, we also extend the practical applicability of the approach and allow for many additional workflow patterns, e.g. cancellation.

Figure 12 shows the reset/inhibitor net of the model in Figure 1. Reset arcs are represented graphically by arcs with double arrows, while inhibitor arcs

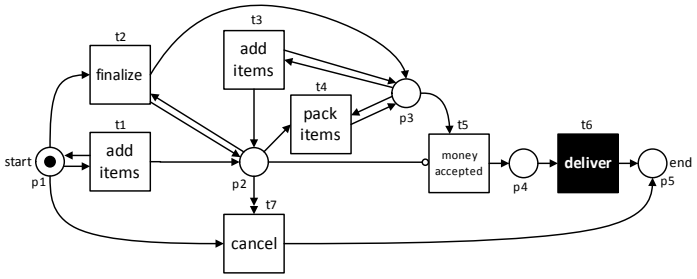

Fig. 12. Translation of the BPMN model in Figure 1 to Petri Net with a reset arc ($r(t_7) = \{p_2\}$) and an inhibitor arc ($i(t_5) = \{p_2\}$)

are decorated with a small circle. Note that cancellation, priorities, and data-constrained behavior (the number of ordered and packaged items must be the same) can be easily modeled with reset/inhibitor arcs.

Formally, we define reset/inhibitor net as follows.

Definition 6.1. (Reset/Inhibitor Net) A *Reset/Inhibitor* net N over a set of activities A is a tuple $(P, T, F, \alpha, m_i, m_f, r, i)$, where

- $(P, T, F, \alpha, m_i, m_f)$ is a Petri Net,
- $r : T \to \mathcal{P}(P)$ is a function mapping transitions to reset places, and
- $i : T \to \mathcal{P}(P)$ is a function mapping transitions to inhibitor places

A transition $t \in T$ is *enabled* at marking $m : P \to \mathbb{N}$ if and only if both $\forall_{p \in P \setminus i(t)}\ F(p,t) \leq m(p)$ and $\forall_{p \in i(t)}\ m(p) = 0$ hold. $m \xrightarrow{t} m'$ denotes the firing of an enabled transition t from m that leads to new marking $m' : P \to \mathbb{N}$, such that $\forall_{p \in P \setminus r(t)}\ m'(p) = m(p) - F(p,t) + F(t,p)$ and $\forall_{p \in r(t)}\ m'(p) = F(t,p)$.

We use the same notation as the one that is already defined in Definition 2.6 for firing sequences. Furthermore, the correctness notion such as easy soundness for Petri nets are also defined on reset/inhibitor nets similarly [30]. Note that if for all $t \in T, r(t) = \emptyset$ and $i(t) = \emptyset$, N has exactly the same behavior as the Petri net $(P, T, F, \alpha, m_i, m_f)$.

Similar to Petri net without reset/inihibitor arcs, the behavior of a reset/inhibitor net can be represented by a transition system. Since all theorems and lemmas that translate the problem of finding optimal alignments into shortest path problems in Section 4 are based on transition systems, they are still hold even for reset/inhibitor nets.

The general idea to construct an alignment between traces and reset/inhibitor nets is the same as the construction we described before for Petri nets in Section 4. Given a trace and a reset/inhibitor net, we construct the product of the event net of the trace and the reset/inhibitor net in a similar way as constructing the product of two Petri nets. The only difference is that we also keep the reset/inhibitor arcs in the event net of the trace and add new reset/inhibitor

arcs to synchronous transitions whose process net transitions are connected to reset/inhibitor arcs.

Figure 13 shows the product of the event net of a trace $\sigma = \langle add\ items, cancel, add\ items, finalize, pack\ items, pack\ items, money\ accepted \rangle$ and the reset/inhibitor net in Figure 12. Transition labels are abbreviated according to their initials. All reset and inhibitor arcs in the original reset/inhibitor net are preserved. Synchronous transitions that are constructed from transitions that are connected to reset/inhibitor arcs also have reset/inhibitor arcs (e.g. synchronous transitions c and m).

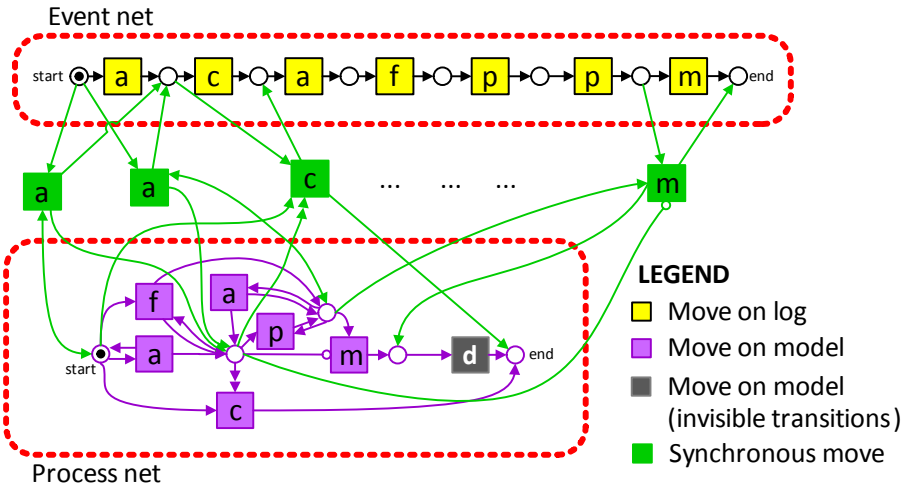

Fig. 13. Product of event net of trace $\langle add\ items, cancel, add\ items, finalize, pack\ items, pack\ items, money\ accepted \rangle$ and the reset/inhibitor net in Figure 12

Definition 6.2. (Product of Two Reset/Inhibitor Nets)
Let $N_1 = (P_1, T_1, F_1, \alpha_1, m_{i,1}, m_{f,1}, r_1, i_1)$ and $N_2 = (P_2, T_2, F_2, \alpha_2, m_{i,2}, m_{f,2}, r_2, i_2)$ be two reset/inhibitor nets over a set of activities A. The *product* of N_1 and N_2 is a reset/inhibitor net $N_3 = N_1 \times N_2 = (P_3, T_3, F_3, \alpha_3, m_{i,3}, m_{f,3}, r_3, i_3)$, such that

- $(P_3, T_3, F_3, \alpha_3, m_{i,3}, m_{f,3}) = (P_1, T_1, F_1, \alpha_1, m_{i,1}, m_{f,1}) \times (P_2, T_2, F_2, \alpha_2, m_{i,2}, m_{f,2})$, i.e. places, arcs, and transitions are created using the product of two ordinary Petri nets (see Definition 4.2),
- $r_3 : T_3 \to \mathcal{P}(P_3)$ and $i_3 : T_3 \to \mathcal{P}(P_3)$ are the reset function and the inhibitor function respectively, such that
  - for all $(t_1, \gg) \in T_3$ where $t_1 \neq \gg$, $r_3((t_1, \gg)) = r_1(t_1)$ and $i_3((t_1, \gg)) = i_1(t_1)$,

- for all $(\gg, t_2) \in T_3$ where $t_2 \neq \gg, r_3((\gg, t_2)) = r_2(t_2)$ and $i_3((\gg, t_2)) = i_2(t_2)$, else
- for all $(t_1, t_2) \in T_3$ where $t_1 \neq \gg$ and $t_2 \neq \gg, r_3((t_1, t_2)) = r_1(t_1) \cup r_2(t_2)$ and $i_3((t_1, t_2)) = i_1(t_1) \cup i_2(t_2)$

The approach to prune the state space and direct state space exploration using the marking equation of Petri nets cannot be applied directly, because the equation is based on the incidence matrix that by definition ignores reset/inhibitor arcs. However, given a reset/inhibitor net, we only need values that *underestimate* the actual distance from states of its transition system to its final state. Therefore, we propose a general idea to estimate the actual distance of states of reset/inhibitor net using a Petri net with some cost function and constraints that can be related to the original reset/inhibitor net and its cost to fire transitions. We call such a net an *estimation net*.

Definition 6.3. (Estimation net and cost)
Let $N_1 = (P, T_1, F_1, \alpha_1, m_i, m_f, r, i)$ be a reset/inhibitor net. Let $c_1 : T_1 \to \mathbb{R}^+$ define the costs of firing transitions in T_1. Let $N_2 = (P, T_2, F_2, \alpha_2, m_i, m_f)$ be a Petri net and let $c_2 : T_2 \to \mathbb{R}$ define the costs of firing transitions in T_2. N_2 with costs c_2 is an *estimation of* N_1 with costs c_1 if for all $\varrho_1 \in T_1^*$ such that $m_i \xrightarrow{\varrho_1}_{N_1} m$, there exists a sequence $\varrho_2 \in T_2^*$ such that $m_i \xrightarrow{\varrho_2}_{N_2} m$, and $\sum_{1 \leq k \leq |\varrho_2|} c_2(\varrho_2[k]) \leq \sum_{1 \leq j \leq |\varrho_1|} c_1(\varrho_1[j])$.

Given a reset/inhibitor net and the costs of firing its transitions, estimation net and cost function are used to provide a permissible underestimation to the minimum distance between states of the reset/inhibitor net.

Theorem 6.4. (Estimation net and cost provide permissible underestimation) Let $TS(N_1) = (S_1, E_1, T_1, \alpha_1, s_i, s_f)$ be the transition system of an easy sound reset/inhibitor net N_1, let $c : T_1 \to \mathbb{R}^+$ define the costs of firing transitions in T_1. Let N_2 be an estimation net of N_1 and let $TS(N_2) = (S_2, E_2, T_2, \alpha_2, s_i, s_f)$ be the transition system of N_2, let $c_2 : T_2 \to \mathbb{R}$ define the costs of firing transitions in T_2. Let $\delta_1 : E_1 \to \mathbb{R}^+$ be the distance of edges such that for all $e_1 \in E_1, \delta(e_1) = \kappa(\pi_2(e_1))$ and let $\delta_2 : E_2 \to \mathbb{R}$ be the distance of edges such that for all $e_2 \in E_2, \delta(e_2) = \kappa(\pi_2(e))$.

For all $s, s' \in S_1, \min(\{\delta_2(\sigma) \mid \sigma \in \Psi_{(S_2, E_2, T_2)}(s, s')\})$ is a permissible underestimation of the distance between s and s' in graph (S_1, E_1, T_1).

Proof. We consider both possible cases:

- Suppose that $\Psi_{(S_1, E_1, T_1)}(s, s') \neq \emptyset$, then there exists $\varrho_1 \in T_1^*$ such that $s \xrightarrow{\varrho_1}_{N_1} s'$. From Definition 2.7, we know that there exists $m_i \xrightarrow{\varrho}_{N_1} s \xrightarrow{\varrho_1}_{N_1} s'$. From Definition 6.3, we also know that there exists $\varrho', \varrho_2 \in T_2^*$ such that $m_i \xrightarrow{\varrho'}_{N_2} s \xrightarrow{\varrho_2}_{N_2} s'$. By the same definition, we know that both $\sum_{1 \leq k \leq |\varrho' \cdot \varrho_2|} c((\varrho' \cdot \varrho_2)[k]) \leq \sum_{1 \leq j \leq |\varrho \cdot \varrho_1|} c((\varrho \cdot \varrho_1[j]))$ and $\sum_{1 \leq k \leq |\varrho'|} c(\varrho[k]) \leq \sum_{1 \leq j \leq |\varrho|} c(\varrho[j]))$ hold, thus $\sum_{1 \leq k \leq |\varrho_2|} c(\varrho_2[k]) \leq$

$\sum_{1 \leq j \leq |\varrho_1|} c(\varrho_1[j])$. Therefore, $\min(\{\delta_2(\sigma_2) \mid \sigma_2 \in \Psi_{(S_2,E_2,T_2)}(s,s')\}) \leq \min(\{\delta_1(\sigma_1) \mid \sigma_1 \in \Psi_{(S_1,E_1,T_1)}(s,s')\})$

– Suppose that $\Psi_{(S_1,E_1,T_1)}(s,s') = \emptyset$, i.e. state s' is not reachable from s in N_1. $\min(\{\delta_2(\sigma) \mid \sigma \in \Psi_{(S_2,E_2,T_2)}(s,s')\}) \leq +\infty$, thus it is a permissible underestimation.

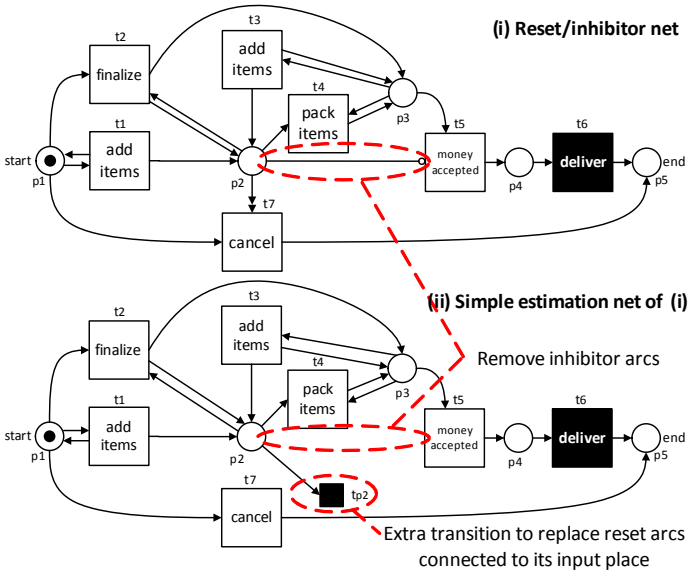

Fig. 14. An estimation net of reset/inhibitor net shown in Figure 12: The inhibitor arc is removed and the reset arc is replaced by a sink transition (t_{p_2})

Given a reset/inhibitor net and a cost function, there are possibly many estimation nets and cost functions that one can construct. We encode reset arcs explicitly as invisible transitions that take tokens from places connected to reset arcs. Figure 14 shows a reset/inhibitor net and its simple estimation net. Transitions and places of the original net are preserved and an extra transition is added for each reset place. Furthermore, inhibitor arcs are removed. Intuitively, all behavior allowed by the original net is allowed by the estimation net, but not necessarily the other way around. Furthermore, we use a cost function where the cost of firing transitions is the same as the cost of firing the original transitions, while the costs of firing the extra transitions are 0. Note that zero cost function is allowed because the net is used only to provide estimation. We show that such a net and cost is an estimation.

First, we define such an estimation net and cost function as follows:

Definition 6.5. (Simple Estimation Petri Nets and Cost)
Let $N = (P, T, F, \alpha, m_i, m_f, r, i)$ be a reset/inhibitor net over a set of activities

A, and let $c : T \to \mathbb{R}^+$ define the costs of firing transitions in N. $N' = (P, T \cup T', F', \alpha', m_i, m_f)$ is the *simple estimation net* of N where

- $T' = \{t_p \mid p \in \bigcup_{t' \in T} r(t')\}$, i.e. an extra transition is added for each reset place in N,
- $F' : (P \times (T \cup T')) \cup ((T \cup T') \times P) \to \mathbb{N}$ is a flow relation returning the weight of arcs, such that
  - for all $e \in (P \times T) \cup (T \times P), F'(e) = F(e)$,
  - for all $t_p \in T'$, the following holds:
    - $F'(t_p, p) = 0$,
    - $F'(p, t_p) = 1$, and
    - For all $p' \in P \setminus \{p\}, F(t_p, p') = F(p', t_p) = 0$
- $\alpha' : (T \cup T') \not\to A$ is a partial labeling function, such that for all $t \in \mathrm{Dom}(\alpha) : \alpha'(t) = \alpha(t)$

Furthermore, $c' : (T \cup T') \to \mathbb{R}$ is a estimation cost function for N' such that for all $t \in T : c'(t) = c(t)$ and for all $t' \in T' : c'(t') = 0$.

We show that the simple estimation net and cost satisfies the requirements in Definition 6.3.

Theorem 6.6. (Simple estimation net and cost satisfies requirements) Let $N = (P, T, F, \alpha, m_i, m_f, r, i)$ be an easy sound reset/inhibitor net over a set of activities A, and let $c : T \to \mathbb{R}^+$ define the costs of firing transitions in N. Let $N' = (P, T \cup T', F', \alpha', m_i, m_f)$ be the simple estimation net of N and let $c' : (T \cup T') \to \mathbb{R}$ be its cost function as defined in Definition 6.5. N' with costs c' provides estimation for N with costs c.

Proof. We show that for all markings m reachable from m_i and for all transition $t \in T$ such that $m \xrightarrow{t}_N m'$, there exists a sequence $\varrho \in (T \cup T')^*$ such that $m \xrightarrow{\varrho}_{N'} m'$ and $c(t) = \sum_{1 \le j \le |\varrho|} c'(t)$.

Since $m \xrightarrow{t}_N m'$, we know that for all $p \in P \setminus i(t) : m(p) \le F(p, t)$ and for all $p' \in i(t) : m(p') = 0$. By definition, for all $p \in P : F'(p, t) = F(p, t)$. Since there is no inhibitor arcs in N', t is also enabled at m in net N'.

Suppose that $m \xrightarrow{t}_{N'} m''$. For all $p \in P \setminus r(t) : F(t, p) = F'(t, p)$ thus $m''(p) = m'(p)$. For all other $p' \in r(t), m''(p') = m(p') - F(p', t) + F(t, p')$. By definition, we know that there exists $t_{p'} \in T'$ that only decrease the marking value of place p' by 1 each time it is fired. For all places $p' \in r(t)$, firing $t_{p'}$ for exactly $(m(p') - F(p', t))$ times from marking m'' leads to a marking $m''' : P \to \mathbb{N}$ where $m'''(p') = F(t, p')$ and $m'''(p'') = m''(p'')$ for all $p'' \in P \setminus \{p'\}$. It is easy to see that firing such sequence of transitions for all places in $r(t)$ from marking m'' yields marking m'. Formally, let $seq(t, n)$ be a shorthand for a sequence of transition t with size n, e.g. $seq(t, 3) = \langle t, t, t \rangle$. The sequence

$\varrho' = t \cdot seq(t_{p_1}, m(p_1) - F(p_1, t)) \cdot \ldots \cdot seq(t_{p_{|r(t)|}}, m(p_{|r(t)|}) - F(p_{|r(t)|}, t))$ where $p_1, \ldots, p_{|r(t)|} \in r(t)$ satisfies $m \xrightarrow{\varrho'}_{N'} m'$.

Furthermore, since the cost of firing all $t_p \in T'$ is 0, the total cost of ϱ' is $c'(t) + 0 \cdot ((m(p_1) - F(p_1, t)) + \ldots + (m(p_{|r(t)|}) - F(p_{|r(t)|}, t))) = c'(t) = c(t)$. □

Theorem 6.6 provides a permissible heuristic function for reset/inhibitor nets. We perform state space analysis as we did before, but instead using the marking equation of the reset/inhibitor net to provide estimation, we use the marking equation of its estimation net. Given a trace and a reset/inhibitor net, for all visited states s in the net, estimation of the remaining distance from s to its final state is the same as the estimation of remaining distance from the same state s in its estimation net to its final state. Hence, the $\mathcal{A}^\star$ algorithm and the marking equation can both be used to compute optimal alignments. Section 7 explains the implementation of this approach and obtained results.

7 Experiments

We performed various experiments to show that the proposed permissible underestimation function reduces the number of explored states during state space exploration. We implemented the approach in ProM 6 [35] and used the lp_solve tool as the ILP solver in our implementation[2].

Two sets of experiments were performed. The first set of experiments compares the proposed approach with existing approaches using two similar artificial models with real-life complexity. The second set of the experiments shows the applicability of the approach to handle real life models and logs and some insights into process executions that one can perform using a study case taken from a municipality in the Netherlands. The former is explained further in Subsection 7.1, while the latter is explained in Subsection 7.2.

7.1 Artifical Logs and Models

The goal of this experiment is to show that our proposed approach is more robust than existing ones to handle large and complex process models. We compare the approach proposed in this paper with the one proposed by Cook and Wolf [7] and Adriansyah et al. [4], using two models and a set of logs generated from both models. Both models loosely describe the process of asking building permission in a municipality in the Netherlands. One model is a Petri net (see Figure 15), and the other is a reset/inhibitor net (see Figure 16). For the sake of readability, some parts of the model are grouped into subprocesses. Both models have the same subprocess of *regular permit check*, shown in Figure 17, and contain invisible transitions, duplicate transitions, and complex control-flow patterns (e.g. OR-splits, loops, and alternatives).

We generated perfectly fitting traces from each model (traces that can be perfectly replayed) with various lengths between 20 to 69 activities per trace,

[2] see `http://sourceforge.net/projects/lpsolve/`

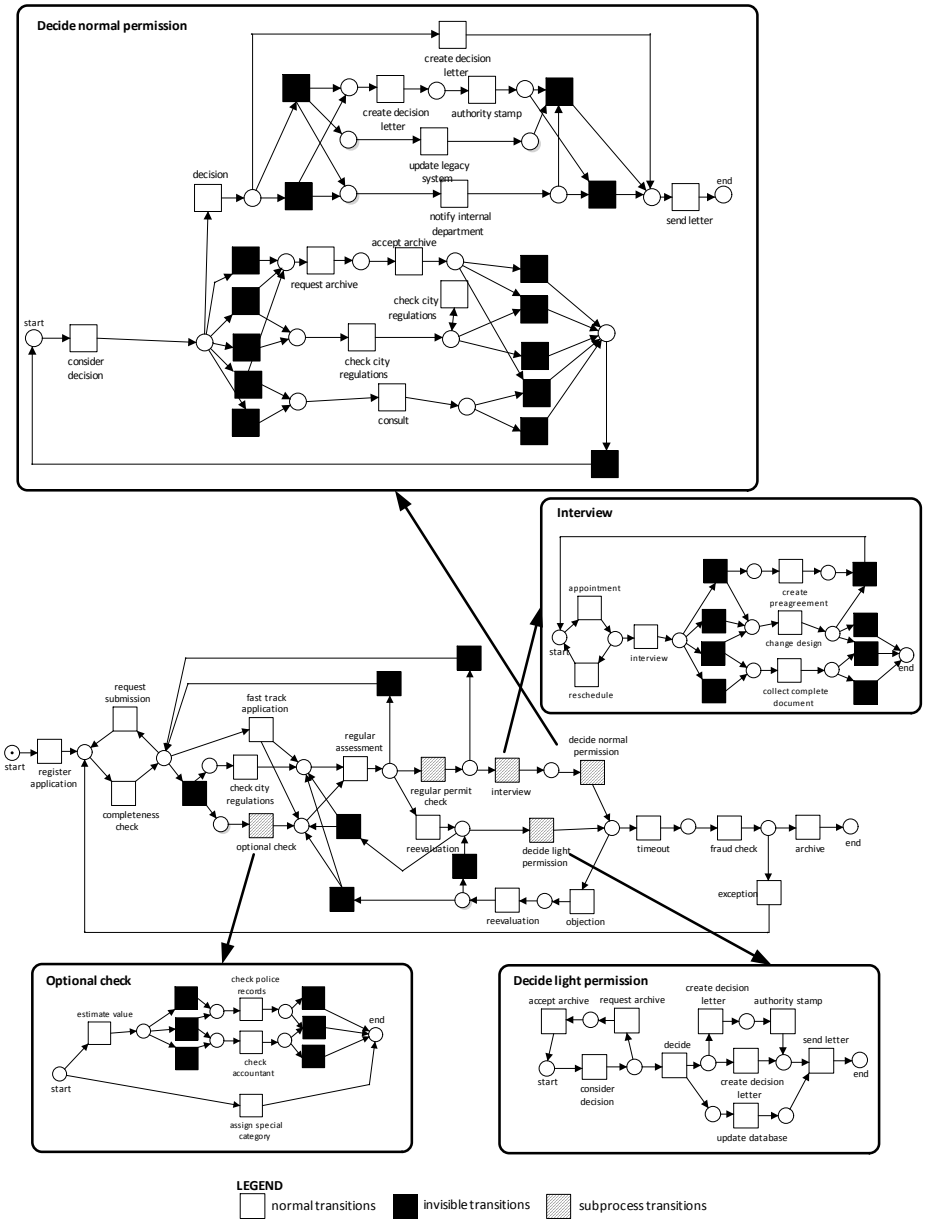

Fig. 15. The Petri net used in the experiments and details of some of its subprocesses

and then introduced noise by randomly removing and/or inserting activities. Then, we constructed an optimal alignment for each trace and its model and recorded the number of queued states (i.e. the states that are actually visited

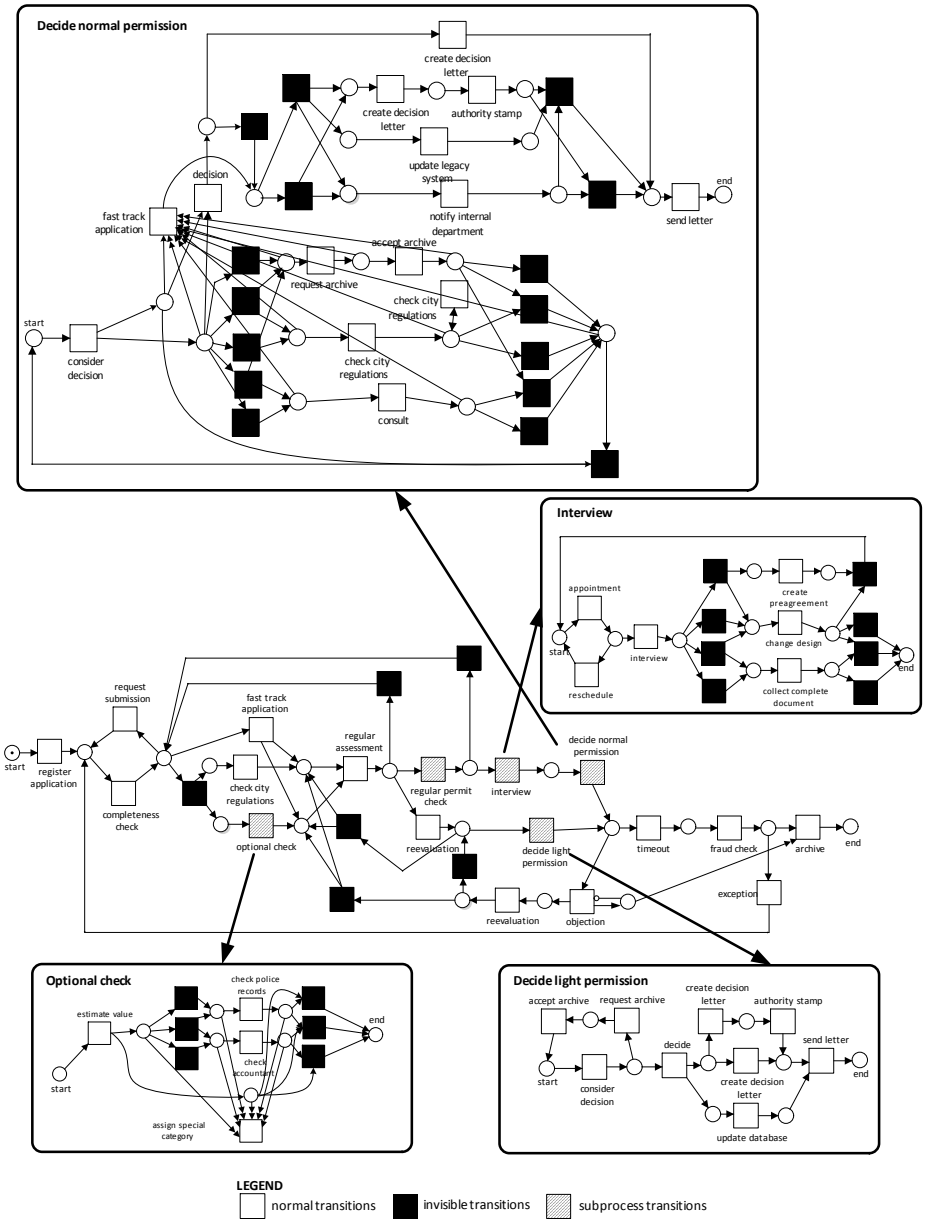

Fig. 16. The reset/inhibitor net used in the experiments and details of some of its subprocesses

and others that are considered as candidates to be visited) needed to construct it.

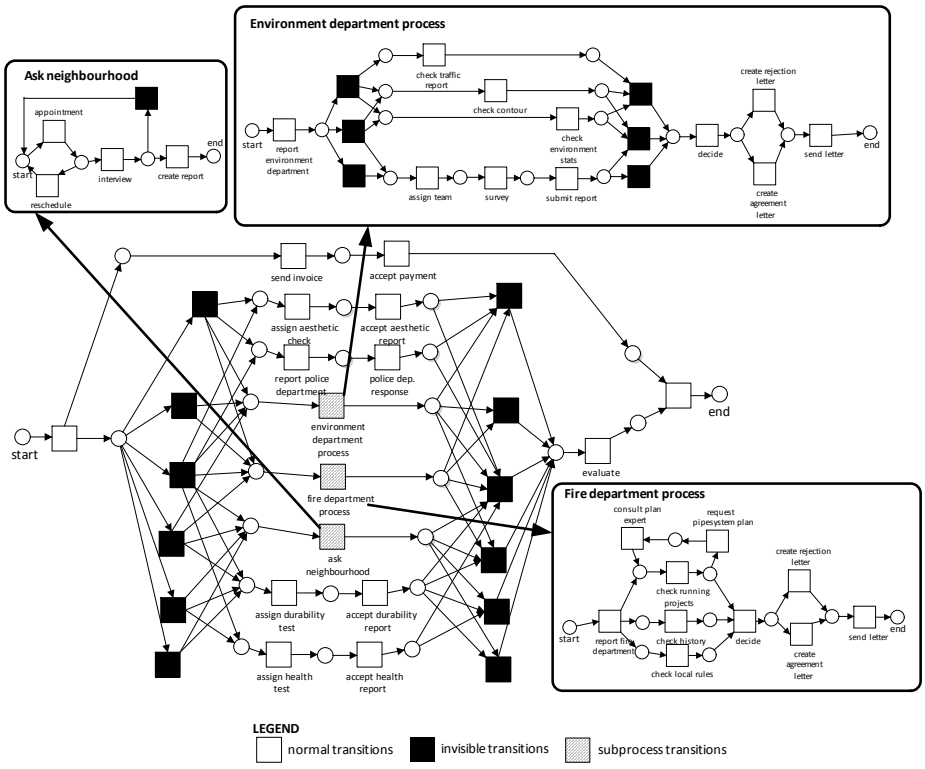

Fig. 17. The sub-process of transition *regular permit check* in both Figure 15 and Figure 16

The cost of synchronous moves/moves on model of invisible transitions has to be negligible in comparison with cost of moves on log/moves on model of non-invisible transitions. Therefore, we assigned the cost for the later movements to be 1,000 higher than the cost of the former movements. Furthermore, to avoid value inconsistency problems when using real values, costs are presented as integers instead of real numbers. Thus, we assign cost $\epsilon = 1$ for all synchronous moves and moves on model of invisible transitions, while cost 1,000 is assigned for all moves on model on non-invisible transitions and for all moves on log.

For benchmarking, we did the same set of experiments with the tree state-space-based approach proposed in [7] and the $\mathcal{A}^\star$ approach proposed in [4] where no ILP calculation is needed. We use a computer with Intel Xeon 2.66 GHz processor and use 1 GB of Java Virtual Memory. The results are shown in Figure 18 to Figure 20. Each dot in the figures is based on the average of performing the same experiments 30 times, each with a different log consists of 100 traces. The vertical bars indicate the corresponding 95% confidence interval. Note that for all figures, y-axis is shown in logarithmic scale.

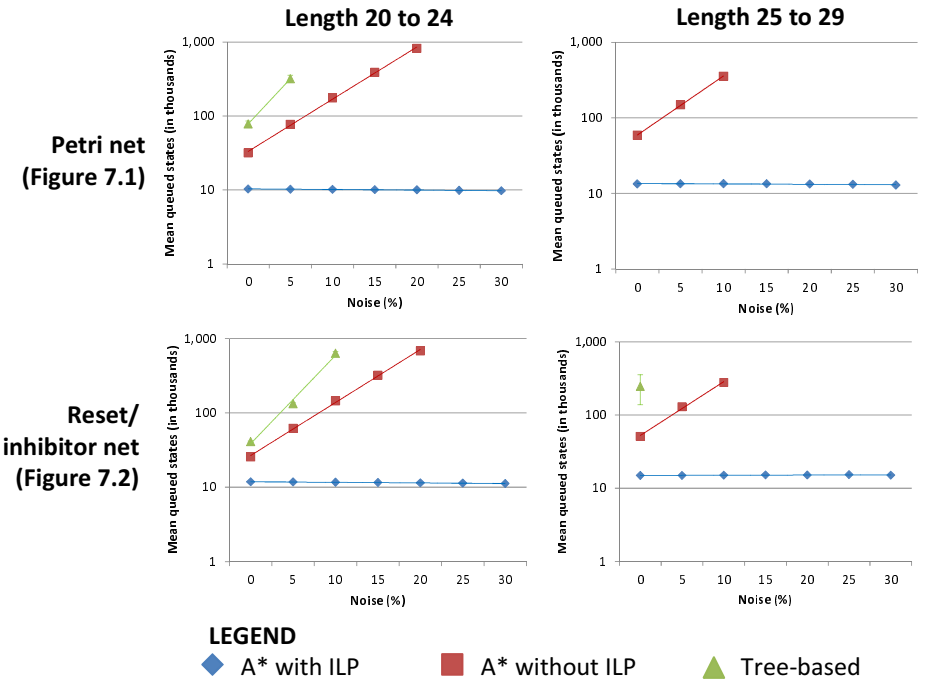

Fig. 18. The number of explored states to construct optimal alignments for Petri net in Figure 15 and reset/inhibitor net in Figure 16. Each dot in the figures is based on the average of performing the same experiment 30 times with different noisy logs where each log consists of 100 traces. Missing values are due to out-of-memory problems of tree-based and $\mathcal{A}^\star$ without ILP.

Figure 18 shows that the number of explored states to construct alignments increases as length of traces and noise level increases. Figure 18 shows that the approach with ILP computation explores much fewer states to construct alignments than other approaches in all cases. Furthermore, it is less sensitive to noise than other approaches. The permissible underestimation functions in Theorem 7 and Theorem 6.6 manage to estimate the cost such that only relevant states towards solutions are explored. Both the approach without ILP and the tree-state-space based exploration do not use such precise estimation. In cases where they need to choose which transition to fire for an OR-split/join pattern, both of them use random selection that may lead to the exploration of many irrelevant states before they finally explore the correct ones. We can also see that the estimation function significantly cuts the number of states that need to be investigated in cases where there is noise.

Figure 18 also shows that only the ILP-based approach managed to compute optimal alignments in all experiments. Other approaches have out-of-memory problems when dealing with either large or noisy logs. For example, in the ex-

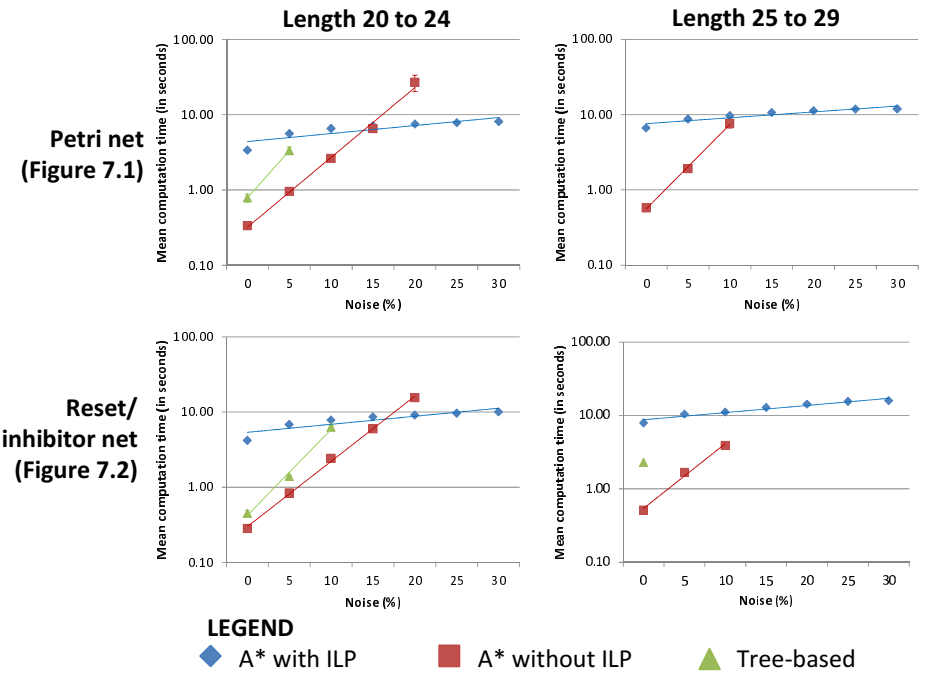

Fig. 19. The computation time required to construct optimal alignments for Petri net in Figure 15 and reset/inhibitor net in Figure 16. Each dot is based on the average of performing the same experiment 30 times with different noisy logs where each log consists of 100 traces. Missing values are due to out-of-memory problems of tree-based and $\mathcal{A}^\star$ without ILP.

periment with Petri net and logs with traces of length between 20 and 24 (see Figure 18, top-left), the tree-based state space approach [7] only managed to compute optimal alignments until noise level reaches 5%. Above 5% noise level, there are too many states that need to be explored by the approach such that out-of-memory problem occurred. The non-ILP approach performs better than the tree-based state space approach, but it can only provide results up to noise level 20% before out of memory problem occurred.

Figure 19 shows the computation time needed to construct optimal alignments using different approaches. As shown in Figure 19, the approach with ILP requires more computation time than the others if there is no noise. The overhead of computing the ILP per visited state does not pay off if there are no or just few deviations. However, in cases where noise exists and traces are long, the approach without ILP explores significantly more states than the one with ILP, such that its total computation time is higher. See for example the experiments with Petri net and log with traces of length between 20 to 24 activities in top-left of Figure 19. When noise level reaches 20%, the ILP approach has lower computation time than the one without ILP. Similarly, the experiment with the

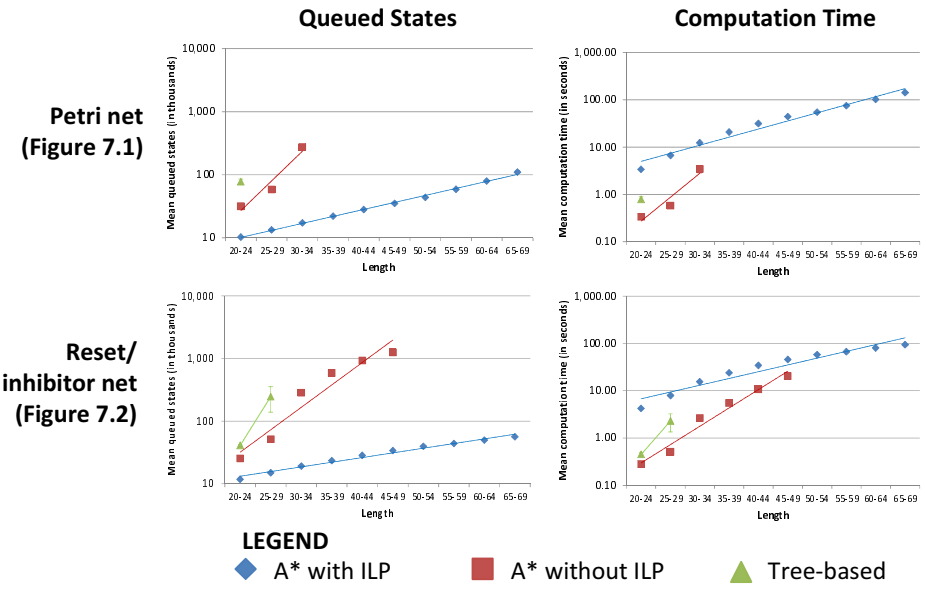

Fig. 20. The computation time and the number of explored states to construct optimal alignments for Petri net in Figure 15 and reset/inhibitor net in Figure 16. Each dot is based on the average of performing the same experiment 30 times with different perfectly fitting logs where each log consists of 100 traces. Missing values are due to out-of-memory problems of tree-based and $\mathcal{A}^\star$ without ILP.

reset/inhibitor net and traces of same length shows the same result when noise level reaches 20% (see Figure 19, bottom-left). Moreover, for larger noise levels the tree-based and $\mathcal{A}^\star$ without ILP are unable to compute alignment due to out-of-memory problems.

Figure 20 shows experiment results using the same models and logs with perfectly fitting traces of various length. As shown in the figure, only the ILP approach managed to provide optimal alignments for all experiment logs while the others fail at logs with long traces due to out of memory problems. This underlines the importance of having a memory-efficient alignment approach.

7.2 Real Life Cases

Alignments are the starting point for various analysis based on both observed and modeled behavior. To show that the approach shows various insights and robust to logs and models with real-life complexity, we take a real-life log of a Dutch financial institution as a case study [34]. The log consists of 13,087 cases with 262,200 events. Since there is no process model associated with the log, we use the algorithm that utilizes passages [32] to discover a process model for the log and modify it manually to make it weakly sound. A screenshot of the model is shown in Figure 21. Then, we computed optimal alignments between

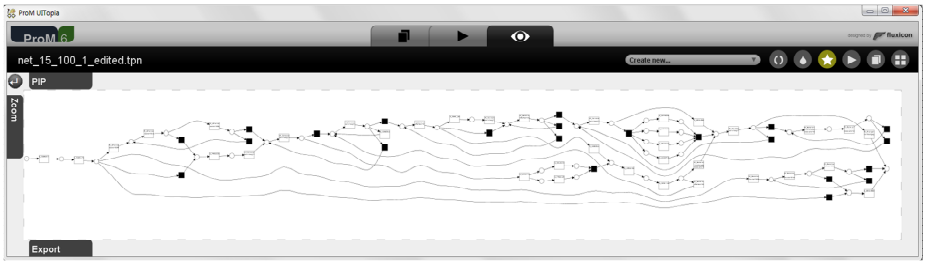

Fig. 21. Process model of a Dutch financial institute

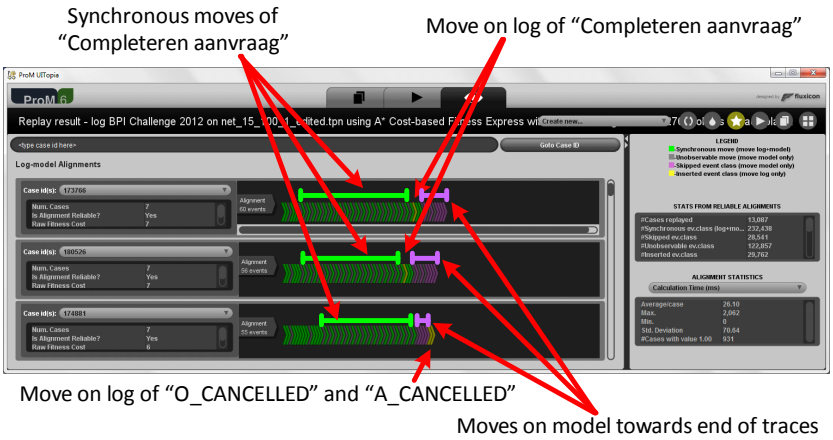

Fig. 22. Screenshot of optimal alignments obtained from the traces in BPI Challenge 2012 log and its discovered process model

the traces in the log and the model to diagnose deviations using the same cost function previously used in experiments of Section 7.1.

Figure 22 shows some of the computed optimal alignments between traces in the log and the model. As shown in Figure 22, there are many loops of activity *Completeren aanvraag* in the log that are also possible according to the model. Furthermore, the figure also shows that many moves on model occurred towards the end of traces. This may indicate that the process execution may not be finished yet, thus some activities that should have occurred according to the model have not been executed in reality. The alignments in Figure 22 clearly show where modeled and observed behavior deviate.

We projected all obtained optimal alignments to the original model to obtain an overall view of occurred deviations and frequently executed activities. Figure 23 shows the projection of all computed optimal alignments onto the original process model. The color of transitions and arcs corresponds to their frequency of occurrence, i.e. the darker their color, the more they occur in the log. For all

transitions, the frequency of move on model is indicated with the ratio of pink-colored stripes below them compares to green-colored stripes. The higher the ratio, the more often move on model occurred compared to synchronous moves. Furthermore, yellow-colored places are places in some markings where moves on log occurred.

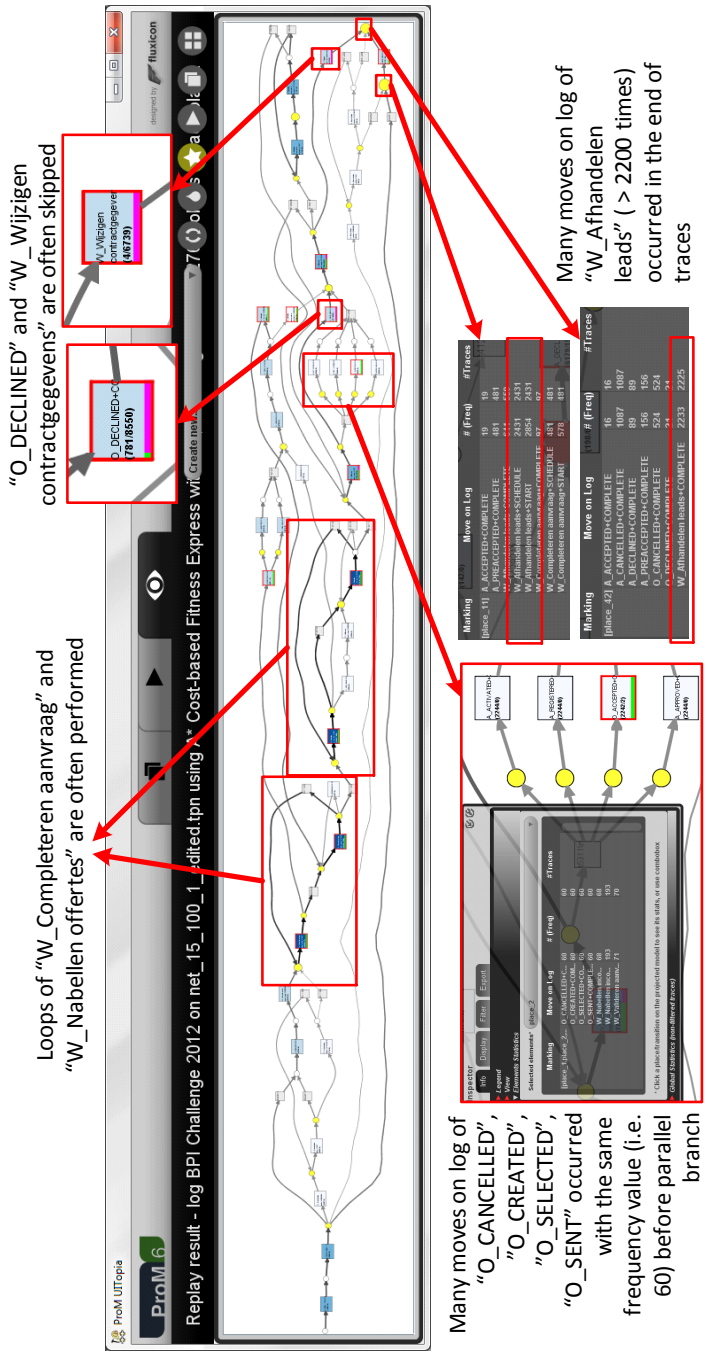

Fig. 23. Screenshots of projected optimal alignments onto process model for deviation analysis.

Thick-colored transitions in Figure 23 show some activities that occured more often than others. In this example, loops of activities *W_Completeren aanvraag* and *W_Nabellen offertes* occurred frequently. The length of pink-colored stripes compared to green stripes in both transitions *O_DECLINED* and *W_Wijzigen contractgegeven* in Figure 23 shows that both activities *O_DECLINED* and *W_Wijzigen contractgegeven* are often skipped. The frequency of moves on model for the two transitions are 8,550 and 6,739 times respectively. The figure also shows that most moves on log in the end of the trace are due to occurrences of activity *W_Afhandelen leads* they are not supposed to occur according to the model. Furthermore, many moves on log of some activities before the parallel branch of *A_ACTIVATED*, *A_REGISTERED*, *O_ACCEPTED*, and *A_APPROVED* occurred with the same frequency value. This may be an indication that the model does not capture behavior where the activities are optionally enabled all at once just before the parallel branch.

The alignments also serve as a basis for quantify the quality of process models with respect to observed behavior [28]. For example, given a process model and an event log, the *fitness* metric measures to which extent the observed behavior in the log can be reproduced by the model. By constructing optimal alignments between all traces in the log and the model and accumulate their cost of deviations (i.e. moves on model of non invisible tasks and moves on log), fitness can be quantified. Using the fitness metric proposed in [28] and the standard cost function, the fitness value between the BPI Challenge 2012 log and the model is 0.80 (from scale 0 to 1, where 1 indicates perfect fitness).

Furthermore, given a trace and a process model, projection of an optimal alignment between the trace and the model to its movements on model provides a complete firing sequence of the model that is most similar to the trace. Such a complete firing sequence can be used to measure *precision*, i.e. the degree for which the model only allow the observed behavior in reality [1]. Moreover, the precision value (a_p^1 [1]) between them is 0.833 (from scale 0 to 1, where 1 indicates prefect precision). Since both fitness and precision value between the log and the model is relatively high, we can say that the model is good enough to represent observed behavior in the log.

Alignments can be exploited to identify bottlenecks in process executions. Given an event log and a proces model, synchronous moves of all optimal alignments between the traces in the log and the model provide a way to map occurrences of events in the log to the model. This mapping can be used to project performance information onto the model to show bottlenecks as proposed in [3]. Figure 24 shows a projection of performance information in the BPI Challenge 2012 log onto its process model using the alignment-based approach proposed in [3]. The color of a place indicate the average time spent between the moment a token enters the place until the moment the same token is consumed by another transition, i.e. *waiting time*. The color of a transition indicate the time spent between the moment the transition is enabled until the moment it is fired, i.e. *sojourn time*. The closer the color of a place(transition) to red, the higher their average waiting(sojourn) time value compares to other places(transitions).

As shown in Figure 24, activity *W_Wijzigen contractgegeven* is the bottleneck of the process, as in average it is performed 1.14 months after it is enabled. However, the activity occurred rarely (only 4 times). We can also see that the average waiting time for the input place of transition *W_Nabellen offertes+START* is high (2.83 days) compared to waiting time of other places and occurred frequently (24,299 times). Thus, improving the waiting time of the place may improve overall performance more significantly than improving the sojourn time of *W_Wijzigen contractgegeven*. Interestingly, we also see that among the four parallel activities: *O_ACCEPTED*, *A_REGISTERED*, *A_ACTIVATED*, and *A_APPROVED*, the activity *O_ACCEPTED* has lower average sojourn times compared to the other activities (27.07 minutes compared to 29.56 minutes for other three activities). Furthermore, all of them have similar frequency of occurrences (i.e. 2,242 for *O_ACCEPTED* and 2,244 for the other three acivities). This may indicate that the four activities are always performed in batch and *O_ACCEPTED* triggers the execution of the other three activities.

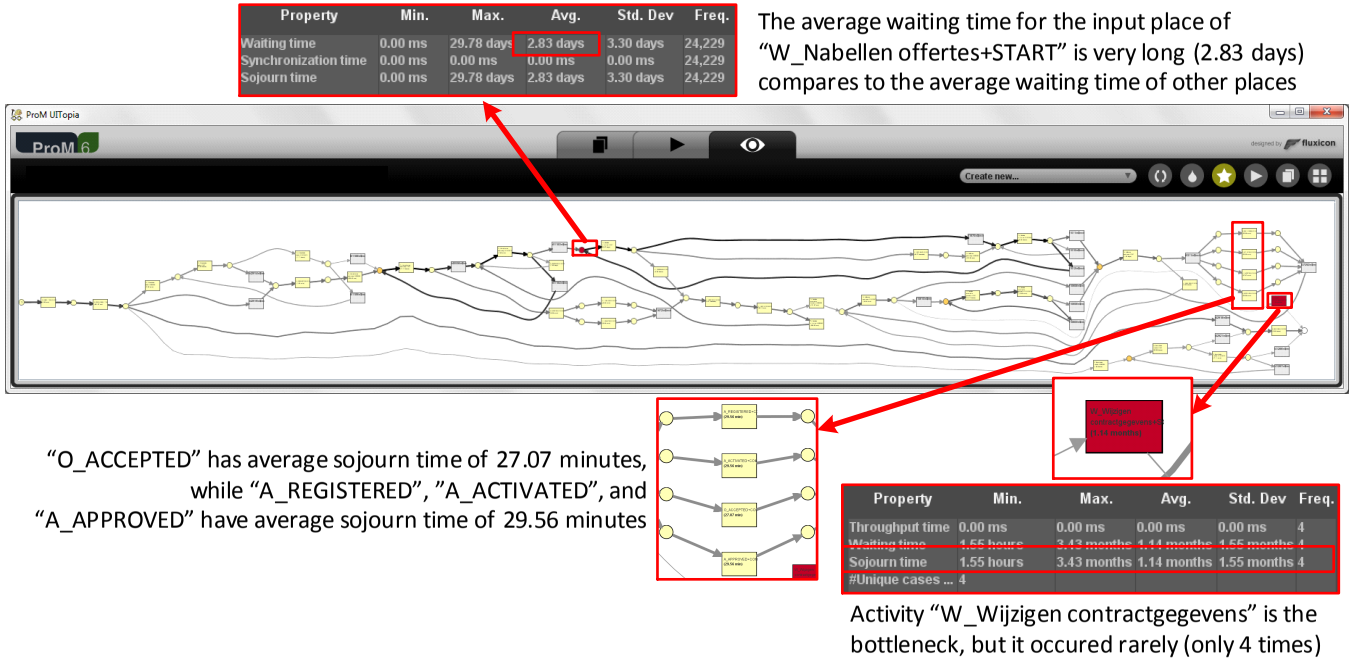

Fig. 24. Screenshots of projected optimal alignments onto process model for performance analysis. Red places/transitions indicate bottlenecks.

8 Conclusion

In situations where process executions are not enforced by systems, deviations between the operational processes of an organization and the models used to describe these processes occur frequently. The analysis of business processes based on observed behavior has proven to be a complex problem in such situations, because analysis techniques typically have difficulties relating the observed behavior to the modeled behavior. In this paper, we consider an approach to align observed behavior in the form of event logs to process models described using a notation such as BPMN. To do so, we align traces recorded in the event logs with Petri net translations of the original models. These alignments are the starting point for conformance checking and performance analysis based on models and event logs.

The alignment of traces to Petri nets is a complex problem which we addressed by translating this problem into a shortest path problem on a (possibly infinite) graph. We showed that a shortest path can always be found using an $\mathcal{A}^\star$ based algorithm with a permissible underestimation function. The techniques we presented in this paper make maximal use of the relation between the structure of the Petri net and it's potential behavior. We exploited the marking equation in our estimation function. Our experiments on real-life size logs and models showed that memory use is reduced significantly in all cases where the marking equation is exploited. In situations where no deviations occur, the computation time overhead of exploiting the marking equation causes a limited increase in overall computation time. However, in cases where deviations are more frequent, which is the case is almost all real-life applications, the use of the marking equation leads to a reduction in computation time on top of the memory reduction. In fact, without using the marking equation, out-of-memory problems occur frequently. Furthermore, we showed using a study case that the approach is robust to logs and models with real-life complexity, and alignments in general provide valuable insights into process executions.

In order to make our approach applicable to real-life languages such as BPMN, EPCs, etc, we extended our techniques to Petri nets with reset and inhibitor arcs. This way we can deal with advanced workflow patterns, such as cancellation, priorities, OR-joins, and timeouts more easily. All real-life languages can be translated to Petri nets with reset and inhibitor arcs while retaining precise semantics and our experiments show that the introduction of these arcs does not lead to a significant increase in memory usage or computation time.

The techniques presented in this paper are fully implemented using the open source framework ProM and the models and logs used in the experiments are publically available from `http://adriansyah.info/publications/ilp`.

References

1. A. Adriansyah, J. Munoz-Gama, J. Carmona, B.F. van Dongen, and W.M.P. van der Aalst. Alignment based precision checking. In *Proceedings of the 8th In-*

ternational Workshop on Business Process Intelligence 2012 (BPI 2012), Tallinn, Estonia, September 2012.
2. A. Adriansyah, N. Sidorova, and B.F. van Dongen. Cost-Based Fitness in Conformance Checking. *International Conference on Application of Concurrency to System Design*, pages 57–66, 2011.
3. A. Adriansyah, B.F. van Dongen, D. Piessens, M.T. Wynn, and M. Adams. Robust Performance Analysis on YAWL Process Models with Advanced Constructs. *Journal of Information Technology Theory and Application (JITTA)*, 12(3):5–26, 2011.
4. A. Adriansyah, B.F. van Dongen, and W.M.P. van der Aalst. Conformance Checking Using Cost-Based Fitness Analysis. pages 55–64, Los Alamitos, CA, USA, 2011. IEEE Computer Society.
5. A. Adriansyah, B.F. van Dongen, and W.M.P. van der Aalst. Towards Robust Conformance Checking. In Michael Muehlen and Jianwen Su, editors, *Business Process Management Workshops*, volume 66 of *Lecture Notes in Business Information Processing*, pages 122–133. Springer Berlin Heidelberg, 2011.
6. L. Baresi and M. Pezzè. Concurrent object-oriented programming and petri nets. chapter On Formalizing UML with High-level Petri Nets, pages 276–304. Springer-Verlag New York, Inc., Secaucus, NJ, USA, 2001.
7. J.E. Cook and A.L. Wolf. Software Process Validation: Quantitatively Measuring the Correspondence of a Process to a Model. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 8:147–176, April 1999.
8. G.B. Dantzig and M.N. Thapa. *Linear Programming 1: Introduction*. Springer, 1997.
9. R. Dechter and J. Pearl. Generalized Best-first Search Strategies and the Optimality of A*. *Journal of the ACM (JACM)*, 32(3):505–536, 1985.
10. C. Dufourd, A. Finkel, and P. Schnoebelen. Reset Nets Between Decidability and Undecidability. In *Proceedings of the 25th International Colloquium on Automata, Languages and Programming*, ICALP '98, pages 103–115, London, UK, UK, 1998. Springer-Verlag.
11. M. Dumas, W.M.P. van der Aalst, and A.H. ter Hofstede, editors. *Process-Aware Information Systems : Bridging People and Software through Process Technology*. Wiley-Interscience, Hoboken, NJ, 2005.
12. D. Fahland and W.M.P. van der Aalst. Repairing process models to reflect reality. In Alistair Barros, Avigdor Gal, and Ekkart Kindler, editors, *Business Process Management*, volume 7481 of *Lecture Notes in Computer Science*, pages 229–245. Springer Berlin / Heidelberg, 2012.
13. P. E. Hart, N. J. Nilsson, and B. Raphael. A Formal Basis for the Heuristic Determination of Minimum Cost Paths in Graphs. *IEEE Trans. Syst. Sci. and Cybernetics*, SSC-4(2):100–107, 1968.
14. N. Lohmann. A Feature-Complete Petri Net Semantics for WS-BPEL2.0. In M. Dumas and R. Heckel, editors, *Web Services and Formal Methods*, volume 4937 of *Lecture Notes in Computer Science*, pages 77–91. Springer Berlin / Heidelberg, 2008.
15. N. Lohmann, E. Verbeek, and R. Dijkman. Transactions on Petri Nets and Other Models of Concurrency II. chapter Petri Net Transformations for Business Processes — A Survey, pages 46–63. Springer-Verlag, Berlin, Heidelberg, 2009.
16. R. Lorenz, G. Juhas, R. Bergenthum, J. Desel, and S. Mauser. Executability of Scenarios in Petri nets. *Theoretical Computer Science*, 410(1213):1190 – 1216, 2009.

17. J. Munoz-Gama and J. Carmona. Enhancing precision in process conformance: Stability, confidence and severity. In *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2011)*, pages 184–191. IEEE, April 2011.
18. T. Murata. Petri nets: Properties, analysis and applications. *Proceedings of the IEEE*, 77(4):541–580, August 2002.
19. OASIS. Business process model and notation (bpmn) version 2.0, 2007.
20. OMG. Business Process Model and Notation (BPMN) Version 2.0, 2011.
21. C. Ouyang, E. Verbeek, W.M.P. van der Aalst, S. Breutel, M. Dumas, and A.H.M. ter Hofstede. Formal Semantics and Analysis of Control Flow in WS-BPEL. *Science of Computer Programming*, 67(2-3):162–198, July 2007.
22. E. Ramezani, D. Fahland, and W.M.P. van der Aalst. Where Did I Misbehave? Diagnostic Information in Compliance Checking. In Alistair Barros, Avigdor Gal, and Ekkart Kindler, editors, *Business Process Management*, volume 7481 of *Lecture Notes in Computer Science*, pages 262–278. Springer Berlin / Heidelberg, 2012.
23. K. Reinhardt. Reachability in Petri Nets with Inhibitor Arcs. *Electronic Notes in Theoretical Computer Science (ENTCS)*, 223:239–264, December 2008.
24. A. Rozinat, R. S. Mans, M. Song, and W.M.P. van der Aalst. Discovering simulation models. *Information Systems*, 34(3):305–327, May 2009.
25. A. Rozinat and W.M.P. van der Aalst. Conformance Checking of Processes Based on Monitoring Real Behavior. *Information Systems*, 33:64–95, March 2008.
26. W.M.P. van der Aalst. Formalization and Verification of Event-driven Process Chains. *Information and Software Technology*, 41(10):639–650, 1999.
27. W.M.P. van der Aalst. *Process Mining: Discovery, Conformance and Enhancement of Business Processes*. Springer Verlag, 2011. ISBN:978-3-642-19344-6.
28. W.M.P. van der Aalst, A. Adriansyah, and B.F. van Dongen. Replaying History on Process Models for Conformance Checking and Performance Analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(2):182–192, 2012.
29. W.M.P. van der Aalst and A.H.M. ter Hofstede. YAWL: Yet Another Workflow Language. *Information Systems*, 30(4):245–275, 2005.
30. W.M.P. van der Aalst, K. M. van Hee, A.H.M. ter Hofstede, N. Sidorova, H. M. W. Verbeek, M. Voorhoeve, and M. T. Wynn. Soundness of workflow nets: Classification, decidability, and analysis. *Formal Aspects of Computing*, 23:333–363, May 2011.
31. W.M.P. van der Aalst, K.M. van Hee, J.M. van der Werf, and M. Verdonk. Auditing 2.0: Using Process Mining to Support Tomorrow's Auditor. *Computer*, 43:90–93, March 2010.
32. W.M.P. van der Aalst and H.M.W. Verbeek. Process Discovery and Conformance Checking Using Passages. BPM Center Report BPM-12-21, BPMcenter.org, 2012.
33. B. F. van Dongen, M. H. Jansen-Vullers, H.M.W. Verbeek, and W.M.P. van der Aalst. Verification of the SAP Reference Models using EPC Reduction, State-space Analysis, and Invariants. *Computers in Industry*, 58(6):578–601, August 2007.
34. B.F. van Dongen. Event log for the bpi challenge 2012, 2012.
35. H.M.W. Verbeek, J.C.A.M. Buijs, B.F. van Dongen, and W. M.P. van der Aalst. ProM 6: The process mining toolkit. In M. La Rosa, editor, *Proceedings of BPM Demonstration Track 2010*, volume 615 of *CEUR Workshop Proceedings*, pages 34–39, Hoboken, USA, September 2010. CEUR-WS.org.
36. H.M.W. Verbeek, M.T. Wynn, W.M.P. van der Aalst, and A.H.M. ter Hofstede. Reduction Rules for Reset/Inhibitor Nets. *Journal of Computer and System Sciences*, 76(2):125–143, March 2010.

37. J. Wang, Y. Deng, and G. Xu. Reachability Analysis of Real-time Systems using Time Petri nets. *IEEE Transactions on Systems, Man, and Cybernetics, Part B: Cybernetics*, 30(5):725 –736, oct 2000.
38. H. Wimmel and K. Wolf. Finding a Witness Path for Non-Liveness in Free-choice Nets. In *Proceedings of the 32nd International Conference on Applications and Theory of Petri Nets*, PETRI NETS'11, pages 189–207, Berlin, Heidelberg, 2011. Springer-Verlag.
39. Glynn Winskel. Petri nets, Algebras, Morphisms, and Compositionality. *Information and Computation*, 72(3):197 – 238, 1987.