Impact-Driven Process Model Repair

ARTEM POLYVYANYY, Queensland University of Technology, Brisbane, Australia
WIL M.P. VAN DER AALST, Eindhoven University of Technology, Eindhoven, The Netherlands, and Queensland University of Technology, Brisbane, Australia
ARTHUR H.M. TER HOFSTEDE, Queensland University of Technology, Brisbane, Australia, and Eindhoven University of Technology, Eindhoven, The Netherlands
MOE T. WYNN, Queensland University of Technology, Brisbane, Australia

The abundance of event data in today's information systems makes it possible to "confront" process models with the actual observed behavior. Process mining techniques use event logs to *discover process models* that describe the observed behavior, and to *check conformance of process models* by diagnosing deviations between models and reality. In many situations it is desirable to mediate between a preexisting model and observed behavior. Hence, we would like to repair the model while improving the correspondence between model and log as much as possible. The approach presented in this paper assigns predefined costs to repair actions (allowing inserting or skipping of activities). Given a maximum degree of change, we search for models that are *optimal in terms of fitness*, i.e., the fraction of behavior in the log not possible according to the model is minimized. To compute fitness we need to align the model and log and this can be time consuming. Hence, finding an optimal repair may be intractable. We propose different alternative approaches to speed-up repair. The number of alignment computations can be reduced dramatically while still returning near-optimal repairs. The different approaches have been implemented using the process mining framework ProM and evaluated using real-life logs.

CCS Concepts: •**Theory of computation** → **Design and analysis of algorithms;** •**Information systems** → *Information systems applications;* •**Mathematics of computing** → Combinatorial optimization;

Additional Key Words and Phrases: Process mining, process model repair, repair recommendation, process model, event log

ACM Reference Format:
Artem Polyvyanyy, Wil M.P. van der Aalst, Arthur H.M. ter Hofstede, Moe T. Wynn. 2016. Impact-Driven Process Model Repair. *ACM Trans. Softw. Eng. Methodol.* (July 2016), 52 pages.

1. INTRODUCTION

Process mining techniques [van der Aalst 2011] aim to bridge the gap between *process-centric* approaches that use models to analyze or implement operational processes and *data-centric* approaches focusing on data analysis (data mining, machine learning, statistics, etc.). Process mining includes (automated) process discovery (learning process models from raw event data), conformance checking (monitoring deviations by comparing model and log), organizational mining, e.g., discovering roles in processes, decision-point analysis, bottleneck analysis, automated construction of simulation

Authors' address: A. Polyvyanyy (artem.polyvyanyy@qut.edu.au), A.H.M. ter Hofstede (a.terhofstede@qut.edu.au), and Moe T. Wynn (m.wynn@qut.edu.au), Business Process Management Discipline, Information Systems School, Science & Engineering Faculty, Queensland University of Technology, Brisbane, Australia. W.M.P van der Aalst (w.m.p.v.d.aalst@tue.nl), Department of Mathematics and Computer Science, Eindhoven University of Technology, Eindhoven, The Netherlands.

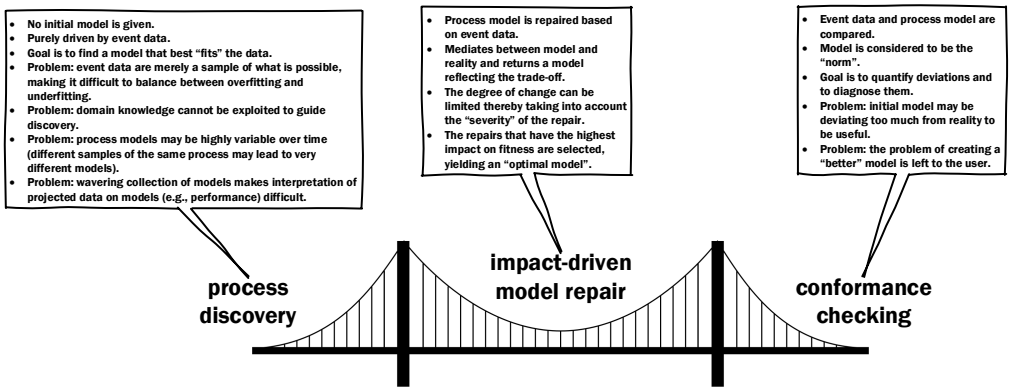


Fig. 1: Impact-driven process model repair as the bridge between two extremes: process discovery (discovering the model using just the data) and conformance checking (taking the model as the "norm", without producing a "better" model).

models (covering different perspectives), and process-centric predictive analytics, e.g., estimating the remaining flow time of a running case or recommending activities and resources.

Input for any process mining exercise is an *event log*. An event log is composed of *traces* and each trace represents a *process instance* (case). A trace corresponds to a sequence of *events* denoting the "life" of a process instance and is typically represented by a sequence of activity labels. Process mining connects observed behavior, i.e., an event log, to a process model, e.g., a Petri net, a BPMN model, a UML activity diagram, or an EPC, to answer questions related to conformance or performance.

This paper focuses on *data-driven process model repair*. Given an initial process model and an event log, a repaired process model is produced. The goal is to create a model that is as close to the original model as possible and at the same time allows for as much of the observed behavior as possible. Process model repair sits in-between *process discovery* and *conformance checking*, cf. Fig. 1. Techniques for process discovery do *not* consider an initial model and construct a model that best "fits the data". Conformance checking techniques consider the initial model to be the "norm" and only diagnose and quantify deviations *without* constructing a new process model. Unlike process discovery and conformance checking, data-driven process model repair approaches aim for a trade-off between observed and modeled behavior. Although this paper focuses on control-flow only, it is important to understand that performance analysis, decision-point analysis, predictive analytics, etc. rely on a proper control-flow model and events related to model elements. In fact, process discovery and model repair serve as a starting point for more advanced types of analysis. The latter are outside the scope of this paper, but of great practical value.

Dozens of process discovery algorithms have been proposed in literature. For example, inductive mining techniques [Leemans et al. 2013; Leemans et al. 2015] provide formal guarantees (soundness and, if desired, also satisfying a preset minimal fitness level), can deal with infrequent behavior, and are highly scalable, i.e., deal with extremely large event logs and support distribution. This triggers the following question: *Why repair an existing model and not simply discover a "fresh" model from the event log?* This question indicates that repair only makes sense if the initial hand-made model provides additional insights. In many process mining scenarios one can find such models. In fact, there may be various reasons for using an initial model:

- The event log may be incomplete and domain knowledge captured in the initial model can help to guide discovery. For more complex processes typically only a fraction of all possible traces can be observed. An initial model may help to avoid overfitting or underfitting the event data.
- Users may be familiar with a process model, e.g., a reference model, and should not be confronted with unnecessarily different models. The process may change over time and one would *only* like to see the significant changes, i.e., repairing a process model rather than discovering a model from scratch. Process discovery techniques may yield very different models even when changes in the

process are small, so, to avoid wavering, changes should only be made if there are significant changes in the event data.
- Process models may be descriptive or normative. If the desired model has normative elements but should also reflect reality as well as possible, then trade-offs are unavoidable. Some deviations may be unacceptable and should not end up in the repaired model. Other deviations may trigger model changes to better reflect reality.

Note that event logs only contain sample data collected over a particular period. Even when the underlying process does not change, the event log in one period may contain traces not observed in other periods and vice versa. If a trace did not happen, one cannot conclude that it could not have happened, i.e., there are no negative traces. Given these uncertainties, it makes sense to exploit a reference or handcrafted model. Process models are used to project information on, e.g., about current bottlenecks or risks. End-users need to be able to interpret this information and, therefore, should have sufficient control over the models used to project this information on.

The repair approach described in this paper builds on the notion of *alignments* [van der Aalst et al. 2012; Adriansyah 2014]. An alignment relates each trace in the event log to a path in the process model, i.e., a model execution. An alignment can be viewed as a sequence of *moves*. Model and reality move synchronously if they agree. If reality and model disagree, the model needs to make a move that cannot be mimicked by the event log or the event log needs to make a move that cannot be mimicked by the model. Alignments serve two purposes: diagnosing where model and reality disagree and mapping traces in the log onto model executions (even when the fit is not perfect). This information can be used to repair the model and to quantify the (possible) effects of repair actions.

Conformance checking techniques can be used to diagnose imperfections of the model, but leave the actual repair of the model to the user. Fahland et al. [Fahland and van der Aalst 2015] proposed a technique to repair models by decomposing the log into several sublogs of non-fitting subtraces. For each sublog, either a loop is discovered that can replay the sublog or a subprocess is derived that is then added to the original model at the appropriate location. Just like in [Fahland and van der Aalst 2015], we aim to exploit the alignments between the original process model and event log. However, we want to control the degree of change and make sure the impact of the changes we allow is maximal. We refer to this as *impact-driven process model repair* (see Fig. 1). Instead of fixing every problem as in [Fahland and van der Aalst 2015], we look for those process model repairs that have most impact (thereby considering various trade-offs).

The approach proposed in this paper can be summarized as follows. We assign costs to repair actions of *inserting* and *skipping* activities in the model. For example, one can decide that inserting a payment activity in the model is more costly than skipping a check activity. To limit model change, the total repair cost (the cost of all repair actions) should not exceed a preset level of available *repair resources*. Thus, if certain repair actions are undesirable, their costs can be set high. We are interested in the set of repair actions that maximize *fitness*. Model and log have perfect fitness if all moves in their alignment are synchronous, i.e., there are no deviations. At the same time, cost of repair must be lower than the given repair resources. We use another cost model to quantify deviations, i.e., mismatches between the model and log, as some deviations may indeed be more severe than others. The sum of all deviations is a measure of fitness, which we strive to minimize.

By checking all possible combinations of repair actions, i.e., inserting and skipping activities, we can always find an "optimal" repaired model within the range set by the available repair resources. This is a model whose total repair cost does not exceed the preset level and has the best fitness with the log. There may be multiple "optimally repaired" models that have the same fitness with the log. Note that it may be time consuming to compute alignments when log and model are large, as we need to solve an optimization problem per trace and there may be thousands or millions of traces in the log. Hence, a brute-force approach becomes intractable if there are many candidate repairs and large event logs. Therefore, we propose various alternative approaches limiting the number of alignment computations. It turns out that we can dramatically decrease computation time, while still returning models of good quality.

The different repair techniques have been implemented using the process mining framework ProM.[1] The impact-driven process model repair approach has been evaluated using artificial event logs (to illustrate particular phenomena) and real-life event logs (to show feasibility in practice). Experiments show that to obtain a small improvement in repair quality, often much more computation time is needed. If the degree of change allowed is small, all the proposed techniques tend to discover same repair recommendations. The results justify our exploration of different repair strategies and demonstrate scalability and feasibility of impact-driven process model repair in real-life settings.

The remainder of the paper is organized as follows. Section 2 introduces basic notions such as event logs and Petri nets. The results are not specific for Petri nets and are equally applicable to other notations, including BPMN, BPEL, EPCs, and UML Activity Diagrams. However, we need a theoretical foundation to reason about repairs and alignments. Section 3 introduces the topic of conformance checking thereby focusing on the seminal notion of alignments. Alignments are later used for repairing models. In Section 4, the topic of event-based model repair is introduced. Section 5 introduces the notion of repair resources to limit the search space of possible repairs, i.e., repair resources are bounded. Given this, the notion of "optimal repair" is defined. Section 6 provides algorithms for different repair approaches including an exhaustive search method for solving the optimal model repair recommendation problem and alternative approaches capable of discovering viable minimal optimal repair recommendations using efficient approximation schemes. In Section 7 the different approaches are evaluated and compared. Related work is discussed in Section 8. Section 9 concludes the paper and outlines future work. Finally, Appendix A and Appendix B list all the proofs of mathematical statements claimed in the paper and the specific mathematical notations used in the paper, respectively.

2. PRELIMINARIES

This section introduces basic concepts that are related to *event logs* and *Petri nets*. These concepts will be used to support later discussions.

2.1. Multisets, Sequences, and Functions

In mathematics, a *multiset*, or a *bag*, is a generalization of the concept of a set that allows a multiset to contain multiple instances of the same element. It is a common practice to employ multisets to encode states of Petri nets. Moreover, an event log is usually formalized as a multiset of traces to capture the fact that the same trace may appear multiple times in the event log.

Let A be a set. By $\mathcal{P}(A)$ and $\mathcal{B}(A)$, where $\mathcal{B}: A \to \mathbb{N}_0$, we denote the power set of A and the set of all finite multisets over A, respectively.[2] For some multiset $B \in \mathcal{B}(A)$, $B(a)$ denotes the multiplicity of element a in B, i.e., the number of times element $a \in A$ appears in B. For example, $B_1 := [\,]$, $B_2 := [a]$, $B_3 := [a,b,b]$, $B_4 := [a,b,b,c,b,a]$, and $B_5 := [a^2,c,b,b^0,b^2]$ are multisets over $A := \{a,b,c\}$. Multiset B_1 is *empty*, i.e., contains no elements. Multiset B_2 contains a single element $a \in A$. Multiset B_3 contains three elements: one occurrence of element $a \in A$ and two occurrences of element $b \in A$. Multisets B_4 and B_5 both contain six elements and are equal, i.e., $B_4 = [a^2,b^3,c] = B_5$. The above examples demonstrate the notation (with its compact version) for describing multisets and hint at the fact that the ordering of elements in multisets is irrelevant.

The standard set operations have been extended to deal with multisets as follows. If element a is a member of multiset B, this is denoted by $a \in B$, while if element b is not a member of B, we write $b \notin B$. For example, for the multisets defined above, it holds that $a \in B_2$, $b \notin B_2$, and $a,b \in B_3$. The union of two multisets C and D, denoted by $C \uplus D$, is the multiset that contains all elements of C and D such that the multiplicity of an element in the resulting multiset is equal to the sum of multiplicities of this element in C and D. For example, $[a,a,b,b] = B_2 \uplus B_3 = [a^2,b^2]$. The difference of two multisets C and D, denoted by $C \setminus D$, is the multiset that for every element x in C

[1] Use our command line tool or download ProM from http://www.promtools.org and use the `SelectivePRepair` package. The command line tool and the package are available via https://svn.win.tue.nl/repos/prom/Packages/SelectivePRepair.

[2] $\mathbb{N}_0$ denotes the set of all natural numbers including zero.

contains $max(0, C(x) - D(x))$ occurrences of element x. For example, it holds that $B_4 \setminus B_3 = [a, b, c]$, $B_2 \setminus B_4 = [\,]$, and $B_4 \setminus B_2 = [a, b^3, c]$. Finally, the cardinality of a multiset B, denoted by $|B|$, is the sum of the occurrences of its members. For example, it holds that $|B_1| = 0$, $|B_2| = 1$, $|B_3| = 3$, and $|B_4| = 6 = |B_5|$.

Given a multiset $B \in \mathcal{B}(A)$ over some set A, by $Set(B)$ we refer to the set that contains all and only elements in B, i.e., $Set(B) := \{b \in A \mid b \in B\}$.

In mathematics, a *sequence* is an ordered list of elements. Similar to multisets, in a sequence, instances of the same element can appear multiple times. However, unlike in multisets, positions of element occurrences in the sequence matter. We use sequences to capture traces in event logs and orderings of transition occurrences in Petri nets. By $\sigma := \langle a_1, a_2, \ldots, a_n \rangle \in A^*$, we denote a sequence of length $n \in \mathbb{N}_0$ over some set A, $a_i \in A$, $i \in [1..n]$.[3] The *empty* sequence, i.e., the sequence without elements, is denoted by $\langle\rangle$. By $|\sigma|$, we indicate the number of all occurrences of elements in σ. Given a sequence σ and a set K, by $\sigma|_K$ we denote the restriction of σ to K, i.e., a sequence obtained from σ by deleting all elements of σ that are not members of K without changing the order of the remaining elements. A sequence σ is said to be a *prefix* of a sequence ω iff there exists a sequence σ' such that concatenation of σ and σ', i.e., the sequence obtained by appending σ' to the end of σ, is equal to ω. Given a sequence $\sigma := \langle a_1, a_2, \ldots, a_n \rangle \in A^*$, $n \in \mathbb{N}_0$, we define the i-th prefix of σ, $i \in [0..n]$, as $\sigma^{[i]} := \langle a_1, a_2, \ldots, a_i \rangle$. For example, if $\sigma := \langle a, b, a, b, a, h, a, l, a, m, a, h, a \rangle$, then it holds that $|\sigma| = 13$, $\sigma^{[2]} = \langle a, b \rangle$, $\sigma^{[5]} = \langle a, b, a, b, a \rangle$, $\sigma^{[7]} = \langle a, b, a, b, a, h, a \rangle$, and $\sigma^{[0]}$ is the empty sequence.

Given a sequence $\sigma := \langle a_1, a_2, \ldots, a_n \rangle \in A^*$, $n \in \mathbb{N}_0$, by $Set(\sigma)$ we refer to the set that contains every element of σ, i.e., $Set(\sigma) := \{b \in A \mid \exists\, i \in [1..n] : a_i = b\}$.

Let $k := (k_1, k_2, \ldots, k_n) \in K_1 \times K_2 \times \ldots \times K_n$ be a point in n-dimensional space, where $K_1, K_2, \ldots, K_n$ are some sets. The *projection function* $\pi_i(k)$, $i \in [1..n]$, is defined as $\pi_i(k) := k_i$, where k_i is the i-th coordinate of k. Let $\kappa := \langle \kappa_1, \kappa_2, \ldots, \kappa_m \rangle$, $m \in \mathbb{N}_0$, where $\kappa_j \in K_1 \times K_2 \times \ldots \times K_n$, $j \in [1..m]$, be a sequence of points in n-dimensional space. Then, $\pi_i(\kappa) := \langle \pi_i(\kappa_1), \pi_i(\kappa_2), \ldots, \pi_i(\kappa_m) \rangle$, $i \in [1..n]$, is the sequence of i-th coordinates of elements of κ.

2.2. Petri Nets and Net Systems

Petri nets is a well-established mathematical modeling language for the description of distributed systems [Reisig 1998]. In this paper, we use Petri nets to discuss our ideas and present results. However, all the discussions can be easily adapted for various other modeling languages, e.g., BPMN, BPEL, EPC, UML Activity Diagram, etc.

DEFINITION 2.1 (PETRI NET).
A *Petri net* is a triple $N := (P, T, F)$, where P and T are finite disjoint sets of *places* and *transitions*, respectively, and $F \subseteq (P \times T) \cup (T \times P)$ is the *flow relation*.

An element $x \in P \cup T$ is a *node* of N. A node $x \in P \cup T$ is an *input* node of a node $y \in P \cup T$ iff $(x, y) \in F$. Similarly, a node $x \in P \cup T$ is an *output* node of a node $y \in P \cup T$ iff $(y, x) \in F$. By $\bullet x$, $x \in P \cup T$, we denote the *preset* of x, i.e., the set of all input nodes of x, while by $x\bullet$, we denote the *postset* of x, i.e., the set of all output nodes of x. For a set of nodes $X \subseteq P \cup T$, $\bullet X := \bigcup_{x \in X} \bullet x$ and $X\bullet := \bigcup_{x \in X} x\bullet$.

Transitions of Petri nets are used to encode actions. It is often convenient to distinguish between *observable* and *silent* transitions to describe actions that have a well-defined meaning and those that have no domain interpretation, respectively. Moreover, one may wish to assign a certain semantics in the application domain to several different transitions. To this end, one can use labeled Petri nets.

Let $\mathcal{A}$ denote the universe of *labels* and let τ signify a special label such that $\tau \notin \mathcal{A}$.

DEFINITION 2.2 (LABELED NET).
A *labeled Petri net*, or a *labeled net*, or a *net*, is a 4-tuple $N := (P, T, F, \lambda)$, where (P, T, F) is a Petri net and $\lambda : T \to \mathcal{A} \cup \{\tau\}$ is a function that assigns labels to transitions.

[3] A^* denotes the Kleene star operation on set A.

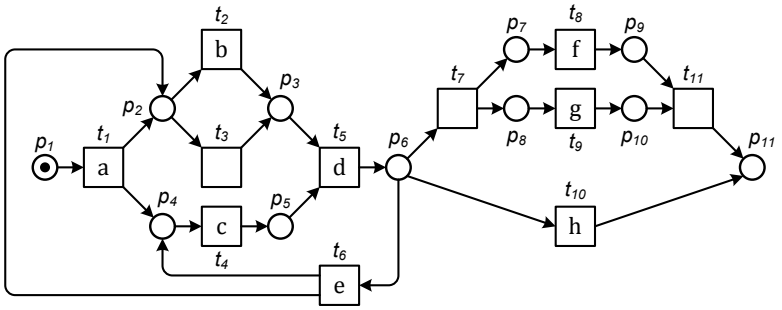


Fig. 2: A net system that is an adapted version of Fig. 2 in [van der Aalst 2013], where a = "update request registration", b = "examine file", c = "check ticket", d = "decide", e = "reinitiate request", f = "send acceptance letter", g = "pay compensation", and h = "send rejection letter".

If $\lambda(t) \neq \tau$, $t \in T$, then t is *observable*; otherwise, t is *silent*.

The execution semantics of labeled nets is defined in terms of states and state transitions. A state of a labeled net is captured by the concept of a *marking*.

DEFINITION 2.3 (MARKING OF A NET).
A *marking* of a labeled net $N := (P,T,F,\lambda)$ is a multiset $M \in \mathcal{B}(P)$. ⌟

A marking M of a labeled net $N := (P,T,F,\lambda)$ is often interpreted as an assignment of *tokens* to places, i.e., marking M 'puts' $M(p)$ tokens at place p, $p \in P$.

Let $N := (P,T,F,\lambda)$ be a labeled net. A transition $t \in T$ is *enabled* in a marking M of N, denoted by $(N,M)[t\rangle$, iff every input place of t contains at least one token, i.e., $\forall p \in {\bullet}t : M(p) > 0$. If a transition $t \in T$ is enabled in a marking M of N, then t can *occur*, which *leads* to a fresh marking $M' := (M \setminus {\bullet}t) \uplus t{\bullet}$ of N, i.e., transition t 'consumes' one token from every input place of t and 'produces' one token for every output place of t. By $(N,M)[t\rangle(N,M')$, we denote the fact that an occurrence of transition t leads from marking M to marking M' in net N.

A finite sequence of transitions $\sigma := \langle t_1,t_2,\ldots,t_n\rangle \in T^*$, $n \in \mathbb{N}_0$, is an *occurrence sequence* of a labeled net $N := (P,T,F,\lambda)$ in a marking M iff σ is empty or there exists a sequence of markings $\langle M_0,M_1,\ldots,M_n\rangle$, such that $M_0 = M$ and for every position $i \in [1..n]$ in σ it holds that $(N,M_{i-1})[t_i\rangle(N,M_i)$; we say that σ *leads* from M_0 to M_n. By $(N,M)[\sigma\rangle(N,M')$, we say that σ leads from marking M to marking M'. A marking M' is *reachable* from a marking M in N iff there exists an occurrence sequence of N that leads from M to M'.

A net system is a labeled net with two markings, one initial and the other final.

DEFINITION 2.4 (NET SYSTEM).
A *net system*, or a *system*, is a triple $S := (N,M_{ini},M_{fin})$, where $N := (P,T,F,\lambda)$ is a labeled net, $M_{ini} \in \mathcal{B}(P)$ is the *initial marking* of N, and $M_{fin} \in \mathcal{B}(P)$ is the *final marking* of N. ⌟

By $\mathbb{S}$, we denote the universe of net systems. Net systems have a well-established graphical notation. In this notation, places are visualized as circles, silent transitions are drawn as empty rectangles while each observable transition depicts the assigned label within its boundaries, every pair of nodes (x,y) in the flow relation is depicted as a directed arc that leads from x to y, and tokens induced by the initial marking are depicted as black dots inside the assigned places. Note that there is no explicit visual notation for capturing the final marking (the reader is cordially invited to invent one).

Fig. 2 visualizes a net system. At this stage the reader should easily recognize that this net system can be formalized as a tuple (N,M_{ini},M_{fin}), $N := (P,T,F,\lambda)$, where $P := \{p_1,\ldots,p_{11}\}$, $T := \{t_1,\ldots,t_{11}\}$, the flow relation F contains 27 ordered pairs that encode all the directed arcs in the figure, e.g., $(p_1,t_1),(t_1,p_2),(t_1,p_4) \in F$ are all the arcs that are incident with transition $t_1 \in T$, $\lambda :=$

$\{(t_1,\mathtt{a}),(t_2,\mathtt{b}),(t_3,\tau),(t_4,\mathtt{c}),(t_5,\mathtt{d}),(t_6,\mathtt{e}),(t_7,\tau),(t_8,\mathtt{f}),(t_9,\mathtt{g}),(t_{10},\mathtt{h}),(t_{11},\tau)\}$, and $M_{ini} := [p_1]$. Transitions t_3, t_7, and t_{11} are silent, while all the other transitions in the figure are observable. An occurrence of an observable transition signifies completion of the corresponding activity (captured by the assigned label). In the figure we use short labels. Their full counterparts are proposed in the caption to Fig. 2. For example, an occurrence of transition $t_1 \in T$ represents completion of activity a, i.e., the activity of updating request registration.

When working with a net system, it is often convenient to refer to its executions.

DEFINITION 2.5 (EXECUTION, LABEL EXECUTION, OBSERVABLE EXECUTION).
An *execution* of a net system $S := (N, M_{ini}, M_{fin})$, $N := (P, T, F, \lambda)$, is an occurrence sequence of N that leads from M_{ini} to M_{fin}. Let $\sigma := \langle t_1, t_2, \ldots, t_n \rangle \in T^*$, $n \in \mathbb{N}_0$, be an execution of S. Then, $\alpha := \langle \lambda(t_1), \lambda(t_2), \ldots, \lambda(t_n) \rangle$ is the *label execution* of S induced by σ, while $\beta := \alpha|_{\mathcal{A}}$ is the *observable execution* of S induced by σ. ⌟

An execution of a net system is an occurrence sequence that leads from its initial to its final marking. Given a net system S, by $\mathbb{E}_S$ we refer to the set of all executions of S.

Note that the labeled net N of the net system in Fig. 2 is a labeled *workflow net* [van der Aalst 1997], i.e., it has a dedicated *source* place p_1 and a dedicated *sink* place p_{11} such that every node of N is on a directed path from the source to the sink. Workflow nets are often used as abstract models for the explicit representation, validation, and verification of business procedures. Every workflow net describes a collection of business cases as executions that lead from its initial marking that puts one token at the source place and no tokens elsewhere to its final marking that puts one token at the sink place and no tokens elsewhere. Thus, it is common practice to define the final marking of a net system (N, M_{ini}, M_{fin}), where $N := (P, T, F, \lambda)$ is a labeled workflow net with a sink place $p \in P$, as the multiset $[p]$. Accordingly, we define the final marking of the net system in Fig. 2 as the marking that puts one token at place p_{11} and no tokens elsewhere.

The sequences of transitions $\langle t_1, t_2, t_4, t_5, t_{10} \rangle$ and $\langle t_1, t_4, t_3, t_5, t_6, t_2, t_4, t_5, t_7, t_9, t_8, t_{11} \rangle$ are two example executions of the net system S in Fig. 2. Note that the set $\mathbb{E}_S$, i.e., the set of all executions of the system in Fig. 2, is infinite due to the presence of a loop.

In the sequel, we require that every net system has at least one execution, i.e., for every $S \in \mathbb{S}$ it holds that $\mathbb{E}_S \neq \emptyset$, and is *bounded*, i.e., the set of all reachable markings of the system is finite. These requirements are introduced to allow alignment-based conformance checking between event logs and net systems, refer to Section 3. No other requirements are imposed on net systems. For example, given a net system, it is possible that the net enables a transition in its final marking.

2.3. Event Logs

An event log is a mathematical model for capturing the historic information about executions of dynamic systems. Formally, an event log is a multiset of traces, where every trace describes an execution of some (distributed) system, i.e., a *process instance*, as a sequence of observed events.

DEFINITION 2.6 (TRACE, EVENT LOG).
A *trace* is a finite sequence of labels. An *event log*, or a *log*, is a multiset over a set of traces. ⌟

We assume that every label in a trace $\upsilon \in \mathcal{A}^*$ signifies an *event*, i.e., an occurrence of an activity. Furthermore, we assume that the order in which labels appear in a trace encodes temporal relations between the events, i.e., a label at position i signifies an event that was observed before an event denoted by a label at position j, $j > i$, of the trace. The ability of an event log to contain multiple instances of a trace allows for capturing the fact that some process execution was observed and recorded several times. Note that in situations when labels in an event log do not directly correspond to occurrences of activities in the corresponding model, such correspondences can often be constructed automatically, e.g., using the techniques proposed in [Baier et al. 2015a; Baier et al. 2015b].

Consider these three example event logs:

- $L_1 := [\langle \mathtt{a},\mathtt{c},\mathtt{d},\mathtt{h} \rangle^{10}, \langle \mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{f},\mathtt{g} \rangle^{7}, \langle \mathtt{a},\mathtt{c},\mathtt{d},\mathtt{e},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{f},\mathtt{g} \rangle^{5}]$,

- $L_2 := [\langle \mathtt{a},\mathtt{c},\mathtt{b},\mathtt{d},\mathtt{f},\mathtt{g}\rangle^{25}, \langle \mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{e},\mathtt{x},\mathtt{c},\mathtt{h},\mathtt{a}\rangle^{15}]$, and
- $L_3 := [\langle \mathtt{a},\mathtt{b},\mathtt{c},\mathtt{f},\mathtt{d},\mathtt{e},\mathtt{f}\rangle^{10}, \langle \mathtt{a},\mathtt{c},\mathtt{d},\mathtt{c},\mathtt{e},\mathtt{d},\mathtt{g},\mathtt{f}\rangle^{9}, \langle \mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{e},\mathtt{x},\mathtt{c},\mathtt{h},\mathtt{a}\rangle^{9}, \langle \mathtt{c},\mathtt{d},\mathtt{d},\mathtt{f},\mathtt{e},\mathtt{g}\rangle^{7}, \langle \mathtt{a},\mathtt{b}\rangle^{6}, \langle \mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{e},\mathtt{d},\mathtt{f}\rangle^{2}, \langle \mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{e},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{g}\rangle^{2}]$.

The reader may recognize that event log L_1 contains information about $10+7+5=22$ executions of the net system S in Fig. 2. Indeed, trace $\upsilon_1 := \langle \mathtt{a},\mathtt{c},\mathtt{d},\mathtt{h}\rangle$ can be interpreted as capturing execution $\sigma := \langle t_1,t_3,t_4,t_5,t_{10}\rangle$ of S; note that υ_1 is the observable execution of S induced by σ. Similarly, traces $\upsilon_2 := \langle \mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{f},\mathtt{g}\rangle$ and $\upsilon_3 := \langle \mathtt{a},\mathtt{c},\mathtt{d},\mathtt{e},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{f},\mathtt{g}\rangle$ can be interpreted as capturing executions $\langle t_1,t_2,t_4,t_5,t_7,t_8,t_9,t_{11}\rangle$ and $\langle t_1,t_4,t_3,t_5,t_6,t_2,t_4,t_5,t_7,t_8,t_9,t_{11}\rangle$ of S, respectively. According to event log L_1, 10 executions followed trace υ_1, 7 executions followed trace υ_2, and 5 executions followed trace υ_3. In total, event log L_1 records information about $4\times 10+6\times 7+9\times 5=127$ observed events.

Event log L_2 stores information about 40 process instances that comprise 285 observed events. While trace $\langle \mathtt{a},\mathtt{c},\mathtt{b},\mathtt{d},\mathtt{f},\mathtt{g}\rangle$ is an observable execution of the net system in Fig. 2 induced by execution $\langle t_1,t_4,t_2,t_5,t_7,t_8,t_9,t_{11}\rangle$, there exists no execution of the net system in Fig. 2 that induces observable execution $\langle \mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{e},\mathtt{x},\mathtt{c},\mathtt{h},\mathtt{a}\rangle$. Finally, event log L_3 describes 45 process instances of some distributed system.

In what follows, we will use the three event logs listed above to exemplify and discuss proposed concepts and principles.

3. CONFORMANCE CHECKING

In an ideal situation, every trace in an event log L of a distributed system S describes some observable execution of S. In this perfect world, we can say that *L fits S perfectly*.

DEFINITION 3.1 (PERFECTLY FITTING EVENT LOG).
An event log $L \in \mathcal{B}(\mathcal{A}^*)$ *fits perfectly* a net system $S \in \mathbb{S}$ if and only if for every trace $\upsilon \in L$ there exists an observable execution β of S for which it holds that $\upsilon = \beta$. ⌟

For example, event log L_1 from Section 2.3 fits perfectly the net system in Fig. 2. In the real world, however, observed process instances may deviate from those captured in the model. The reasons for deviations are manifold. For example, a deviation can be caused by a decision to perform an ad hoc activity while executing an outdated model that no longer caters for all the required process instances, or may stem from a bypass of a recognized problem in the model. In this light, event log L_2 proposed in Section 2.3 can be interpreted as a data structure that captures the historic information on executions of the net system in Fig. 2.

In process mining, *conformance checking* refers to the following problem: Given an event log L and a model of a distributed system S, e.g., a net system, check whether traces in L are in accordance with (observable) executions of S.

Conformance checking is interesting for various reasons. It can be used to audit processes to check whether observed reality conforms to some descriptive model [van der Aalst et al. 2010], as deviations between modeled and observed processes may point to fraudulent cases and poorly designed process practices. Moreover, conformance checking can serve as the basis for evaluating performance of process discovery techniques. Specifically, techniques for conformance checking can be used to assess how well do process instances recorded in an event log L fit a model that is discovered from L, cf. [van der Aalst 2011].

A system with a 'good' *fitness* allows for most of the process instances seen in the event log. There are various ways to measure fitness between event logs and net systems [Rozinat and van der Aalst 2008; van der Aalst et al. 2012; Munoz-Gama and Carmona 2011]. A starting point when studying the degree to which a trace in an event log deviates from some execution of a system is the *alignment* between these two artifacts. Consider the five alignments proposed below.

$\gamma_1 :=$

a	c	b	d	≫	f	g	≫
a	c	b	d	τ	f	g	τ
t_1	t_4	t_2	t_5	t_7	t_8	t_9	t_{11}

$\gamma_2 :=$

a	c	b	d	≫	≫	≫	≫	≫	f	g	≫
a	c	b	d	e	c	τ	d	τ	f	g	τ
t_1	t_4	t_2	t_5	t_6	t_4	t_3	t_5	t_7	t_8	t_9	t_{11}

$$\gamma_3 := \begin{array}{|c|c|c|c|c|c|c|c|c|}\hline \texttt{a}&\texttt{b}&\texttt{c}&\texttt{d}&\texttt{e}&\texttt{x}&\texttt{c}&\texttt{h}&\texttt{a}\\ \hline \texttt{a}&\texttt{b}&\texttt{c}&\texttt{d}&\gg&\gg&\gg&\texttt{h}&\gg\\ t_1&t_2&t_4&t_5&&&&t_{10}&\\ \hline\end{array} \qquad \gamma_4 := \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|}\hline \texttt{a}&\texttt{b}&\texttt{c}&\texttt{d}&\texttt{e}&\texttt{x}&\texttt{c}&\gg&\gg&\texttt{h}&\texttt{a}\\ \hline \texttt{a}&\texttt{b}&\texttt{c}&\texttt{d}&\texttt{e}&\gg&\texttt{c}&\tau&\texttt{d}&\texttt{h}&\gg\\ t_1&t_2&t_4&t_5&t_6&&t_4&t_3&t_5&t_{10}&\\ \hline\end{array}$$

$$\gamma_5 := \begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline \texttt{a}&\texttt{b}&\texttt{c}&\texttt{d}&\texttt{e}&\gg&\texttt{x}&\texttt{c}&\gg&\texttt{h}&\texttt{a}\\ \hline \texttt{a}&\texttt{b}&\texttt{c}&\texttt{d}&\texttt{e}&\tau&\gg&\texttt{c}&\texttt{d}&\texttt{h}&\gg\\ t_1&t_2&t_4&t_5&t_6&t_3&&t_4&t_5&t_{10}&\\ \hline\end{array}$$

These are alignments between traces $\langle \texttt{a},\texttt{c},\texttt{b},\texttt{d},\texttt{f},\texttt{g}\rangle$ and $\langle \texttt{a},\texttt{b},\texttt{c},\texttt{d},\texttt{e},\texttt{x},\texttt{c},\texttt{h},\texttt{a}\rangle$ in event log L_2 listed in Section 2.3 and executions of the net system in Fig. 2. The top row of an alignment refers to 'steps on trace', while the bottom two rows refer to 'steps on system'. Note that every step on system is given by the transition and its label, as there could be multiple transitions with the same label; for example, transitions t_3, t_7, and t_{11}, all 'carry' label τ.

An alignment between a trace in an event log and an execution of a net system can be formalized as a sequence of moves, where a *move* is a pair in which the first component refers to an element in the trace and the second component refers to an element in the execution. Let '$\gg$' be a special 'no move' symbol which is neither a label, i.e., $\gg \notin \mathcal{A} \cup \{\tau\}$, nor a transition, i.e., for every net system $(N, M_{ini}, M_{fin}) \in \mathbb{S}$, where $N := (P, T, F, \lambda)$, it holds that $\gg \notin T$. A move is a pair as detailed below.

DEFINITION 3.2 (MOVE, LEGAL MOVE).
A *move* over a net system $S := (N, M_{ini}, M_{fin}) \in \mathbb{S}$, where $N := (P, T, F, \lambda)$, is a pair $(x, y) \in (\mathcal{A} \cup \{\gg\}) \times (T \cup \{\gg\}) \setminus \{(\gg, \gg)\}$.

- Move (x, y) is a *move on trace* iff $x \in \mathcal{A}$ and $y = \gg$.
- Move (x, y) is a *move on system* iff $x = \gg$ and $y \in T$.
- Move (x, y) is a *synchronous move* iff $x \in \mathcal{A}$, $y \in T$, and $\lambda(y) = x$.

Move (x, y) is a *legal move* over S if it is either a move on trace, a move on system, or a synchronous move; otherwise move (x, y) is an *illegal move* over S. ⌟

By $\mathbb{M}_S$, we denote the set of all legal moves over a net system S. Moves on trace are introduced to capture situations when events in traces cannot be mimicked by occurrences of transitions. For examples, $(\texttt{a}, \gg)$ and $(\texttt{x}, \gg)$ are the two moves on trace of alignment γ_4 proposed above. Moves on system represent cases when occurrences of transitions cannot be mimicked by events in traces. For instance, $(\gg, t_3)$ and $(\gg, t_5)$ are the two moves on system of alignment γ_4. Finally, synchronous moves signify occurrences of transitions that are mimicked by events in traces. For example, $(\texttt{a}, t_1)$, $(\texttt{b}, t_2)$, $(\texttt{c}, t_4)$, $(\texttt{d}, t_5)$, $(\texttt{e}, t_6)$, and $(\texttt{h}, t_{10})$ are all the synchronous moves of γ_4.

An *alignment* between a trace υ and an execution of a net system σ is a sequence of moves that 'respects' the orders of elements in υ and σ.

DEFINITION 3.3 (ALIGNMENT).
An *alignment* between a trace $\upsilon \in \mathcal{A}^*$ and an execution σ of a net system $S := (N, M_{ini}, M_{fin})$, where $N := (P, T, F, \lambda)$, is a finite sequence $\gamma \in \mathbb{M}_S^*$ of legal moves over S for which it holds that $\pi_1(\gamma)|_{\mathcal{A}} = \upsilon$ and $\pi_2(\gamma)|_T = \sigma$. ⌟

As illustrated above, in dealing with alignments it is convenient to represent them by means of tables. In such a representation, moves are encoded as columns, where two successive columns in a table refer to two successive moves in the alignment. Each column in an alignment table has three rows. Given a move over a net system, the corresponding column of an alignment table is populated as follows. The top row of the column contains the first component of the move, which is either a label that encodes an observed event or the special no move symbol '$\gg$'. The bottom row of the column is kept empty if the second component of the move is the no move symbol. Otherwise, it contains the second component of the move, which encodes an occurrence of a transition. The middle row contains '$\gg$' if the second component of the move is the special no move symbol. Otherwise, it contains a label assigned to a transition recorded as the second component of the move. Since an

alignment table specifies the alignment unambiguously, it is customary and convenient to refer to the table of an alignment γ as γ itself.

For instance, two tables γ_1 and γ_2 proposed in the beginning of this section specify two alignments defined by two sequences of moves $\langle(\mathtt{a},t_1),(\mathtt{c},t_4),(\mathtt{b},t_2),(\mathtt{d},t_5),(\gg,t_7),(\mathtt{f},t_8),(\mathtt{g},t_9),(\gg,t_{11})\rangle$ and $\langle(\mathtt{a},t_1),(\mathtt{c},t_4),(\mathtt{b},t_2),(\mathtt{d},t_5),(\gg,t_6),(\gg,t_4),(\gg,t_3),(\gg,t_5),(\gg,t_7),(\mathtt{f},t_8),(\mathtt{g},t_9),(\gg,t_{11})\rangle$, respectively. Note that alignments γ_1 and γ_2 do indeed 'respect' the orders of elements in traces and executions as it holds that $\pi_1(\gamma_1)|_{\mathcal{A}} = \langle\mathtt{a},\mathtt{c},\mathtt{b},\mathtt{d},\mathtt{f},\mathtt{g}\rangle$, $\pi_2(\gamma_1)|_T = \langle t_1,t_4,t_2,t_5,t_7,t_8,t_9,t_{11}\rangle$, $\pi_1(\gamma_2)|_{\mathcal{A}} = \langle\mathtt{a},\mathtt{c},\mathtt{b},\mathtt{d},\mathtt{f},\mathtt{g}\rangle$, $\pi_2(\gamma_2)|_T = \langle t_1,t_4,t_2,t_5,t_6,t_4,t_3,t_5,t_7,t_8,t_9,t_{11}\rangle$; T is the set of all transitions in Fig. 2.

Two alignments between the same trace but two distinct executions show different deviations between the observed and modeled process instances. A common approach to comparing alignments is by quantifying them based on the moves that they contain.

DEFINITION 3.4 (COST OF LEGAL MOVES).
A *cost function on legal moves* over a net system $S \in \mathbb{S}$ is a function $c : \mathbb{M}_S \to \mathbb{N}_0$ that assigns costs to legal moves over S. ⌟

The cost of an alignment is the sum of costs of all moves of this alignment.

DEFINITION 3.5 (ALIGNMENT COST).
The *cost of an alignment* γ between a trace $\upsilon \in \mathcal{A}^*$ and an execution σ of a net system $S \in \mathbb{S}$ as per a cost function c on legal moves over S is denoted by $c(\gamma)$ and is the sum of costs of all moves of γ, i.e., $c(\gamma) := \sum_{i=1}^{|\gamma|} c(\gamma_i)$. ⌟

It is often convenient to work with the standard cost function. The *standard cost function* on legal moves over a net system $S := (N, M_{ini}, M_{fin})$, $N := (P, T, F, \lambda)$, is the function $stdc_S : \mathbb{M}_S \to \{0,1\}$ such that $stdc_S(x,y) = 0$ if it holds that (x,y) is either a synchronous move over S, or $x = \gg$, $y \in T$ and $\lambda(y) = \tau$, and $stdc_S(x,y) = 1$ otherwise. We will omit the subscript S where the context is clear. The cost of an alignment as per the standard cost function is equal to the number of moves on trace and moves on system that are defined for observable transitions in the alignment, i.e., it is equal to the number of deviations between the observed process instance and the modeled one.

Given a trace and two alignments between this trace and two distinct executions of a net system, the one with the lower cost exhibits less severe deviations (as per the employed cost function) from the corresponding execution, i.e., it justifies a better fit of the trace and the net system. For example, $stdc_S(\gamma_1) = 0$, $stdc_S(\gamma_2) = 3$, $stdc_S(\gamma_3) = 4$, $stdc_S(\gamma_4) = 3$, and $stdc_S(\gamma_5) = 3$, where S is the net system in Fig. 2 and γ_1–γ_5 are the five alignments proposed in the beginning of this section. Thus, as per the standard cost function, alignment γ_1 justifies a 'better' fit between trace $\langle\mathtt{a},\mathtt{c},\mathtt{b},\mathtt{d},\mathtt{f},\mathtt{g}\rangle$ and S than alignment γ_2. Similarly, γ_4 and γ_5 justify a 'better' fit between $\langle\mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{e},\mathtt{x},\mathtt{c},\mathtt{h},\mathtt{a}\rangle$ and S than γ_3.

Given a trace $\upsilon \in \mathcal{A}^*$ and a net system $S \in \mathbb{S}$, by $\mathbb{A}_S^\upsilon$ we denote the set that for every execution $\sigma \in \mathbb{E}_S$ contains all alignments between υ and σ, and contains nothing else, i.e., $\mathbb{A}_S^\upsilon := \{\gamma \in \mathbb{M}_S^* \mid \exists\, \sigma \in \mathbb{E}_S \textit{ such that } \gamma \textit{ is an alignment between } \upsilon \textit{ and } \sigma\}$. An *optimal* alignment between a trace and a system is an alignment between this trace and an execution of the system that yields the lowest cost.

DEFINITION 3.6 (OPTIMAL ALIGNMENT).
An *optimal alignment* between a trace $\upsilon \in \mathcal{A}^*$ and a system $S \in \mathbb{S}$ as per a cost function c on legal moves over S is an alignment $\gamma \in \mathbb{A}_S^\upsilon$ such that $\forall \gamma' \in \mathbb{A}_S^\upsilon : c(\gamma) \leq c(\gamma')$. ⌟

The problem of finding an optimal alignment between a trace and a net system is an optimization problem. The reader can refer to [Adriansyah et al. 2011] for a solution to this problem. An optimal alignment between a trace and a net system suggests an execution of the net system with the 'cheapest' required deviations from the trace (as per the employed cost function), i.e., it gives one among possibly several best fits between the trace and the system. It clearly holds that alignment γ_1 is an optimal alignment between trace $\langle\mathtt{a},\mathtt{c},\mathtt{b},\mathtt{d},\mathtt{f},\mathtt{g}\rangle$ and the system S in Fig. 2 as $stdc_S(\gamma_1) = 0$. Interestingly, alignment γ_4 is an optimal alignment between trace $\langle\mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{e},\mathtt{x},\mathtt{c},\mathtt{h},\mathtt{a}\rangle$ and S (as per the standard cost function); there exists no alignment γ between this trace and S such that $stdc_S(\gamma) < 3$.

Given a trace $\upsilon \in \mathcal{A}^*$, a system $S \in \mathbb{S}$, and a cost function c on legal moves over S, by $\mathbb{O}^{\upsilon}_{S,c}$ we denote the set of all optimal alignments between υ and S as per c. It is immediate that for every two optimal alignments $\gamma, \gamma' \in \mathbb{O}^{\upsilon}_{S,c}$ it holds that $c(\gamma) = c(\gamma')$. Hence, we define the cost of optimal alignment between υ and S as per cost function c, denoted by $cost[\upsilon, S, c]$, as the cost of some optimal alignment between υ and S as per c, i.e., $cost[\upsilon, S, c] := c(\gamma)$, where $\gamma \in \mathbb{O}^{\upsilon}_{S,c}$. For example, it holds that $\mathbb{O}^{\upsilon}_{S,stdc} = \{\gamma_4, \gamma_5\}$ and $cost[\upsilon, S, stdc] = 3$, where $\upsilon := \langle \mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{e},\mathtt{x},\mathtt{c},\mathtt{h},\mathtt{a} \rangle$, S is the system in Fig. 2, and γ_4 and γ_5 are two out of the five alignments listed in the beginning of this section. The reader can refer to [Adriansyah 2014] for a solution to the problem of finding all optimal alignments between a trace and a system.

Finally, the cost of optimal alignment between an event log $L \in \mathcal{B}(\mathcal{A}^*)$ and a net system $S \in \mathbb{S}$ is the sum of costs of optimal alignments between each trace in L and S.

DEFINITION 3.7 (COST OF OPTIMAL ALIGNMENT).
The *cost of optimal alignment* between an event log $L \in \mathcal{B}(\mathcal{A}^*)$ and a net system $S \in \mathbb{S}$ as per a cost function c on legal moves over S is denoted by $cost[L, S, c]$ and equals $\sum_{\upsilon \in L} (L(\upsilon) \times cost[\upsilon, S, c])$. ⌟

The cost of optimal alignment between an event log and a net system can be used as a basis for measuring fitness between the event log and the net system. Intuitively, the lower the cost of optimal alignment, the better the event log fits the net system. For example, one can conclude that event log L_1 fits the net system S in Fig. 2 better than event log L_2 based on the observation that $cost[L_1, S, stdc] < cost[L_2, S, stdc]$. Indeed, it holds that $cost[L_1, S, stdc] = 10 \times 0 + 7 \times 0 + 5 \times 0 = 0$ and $cost[L_2, S, stdc] = 25 \times 0 + 15 \times 3 = 45$ (note for the multiplicities of traces in the event log).

There are various ways to convert the cost of an optimal alignment [van der Aalst et al. 2012; Adriansyah 2014] into a fitness value in-between 0 (poor fitness) and 1 (perfect fitness). For example, one can take the costs of undesirable moves, i.e., non-synchronous moves, and divide that by the potential costs of all moves in the alignment. It is also possible to take the cost of an optimal alignment and divide that by the cost of a worst case scenario where we have no synchronous moves, i.e., all events in the log correspond to moves on trace and the system proceeds to the end by just using moves on system. In the paper, we abstract from the fitness calculation and focus on the cost of an optimal alignment. However, intuitively it should be clear that lower optimal alignment costs correspond to higher fitness.

The optimal alignment cost of zero as per the standard cost function is an indicator of a perfectly fitting event log.

LEMMA 3.8 (PERFECTLY FITTING EVENT LOG, LEMMA 1 IN [VAN DER AALST 2013]).
An event log $L \in \mathcal{B}(\mathcal{A}^*)$ fits perfectly a net system $S \in \mathbb{S}$ iff $cost[L, S, stdc_S] = 0$. ⌟

The proof of Lemma 3.8 revolves around the fact that the standard cost function assigns zero costs only to synchronous moves and moves on system that are defined for silent transitions, which are the only permissible moves of an optimal alignment between a trace in a perfectly fitting event log and the net system, cf. [van der Aalst 2013].

4. MODEL REPAIR

In process mining, *model repair* refers to the following problem: Given an event log L that does not fit a model of a distributed system S, e.g., a net system, transform S into S' such that L fits S' 'better' than it fits S. Thus, model repair allows keeping models in sync with observed process instances by suggesting mechanisms for reflecting traces encountered in event logs in executions of models.

This section contributes to the body of knowledge on *model repair* by introducing the notions of a repair recommendation, an (optimal) alignment-based repair of a model based on a given repair recommendation, and a naïve repair of a model. While a repair recommendation can be seen as an abstract specification of what should be repaired in a model, an alignment-based repair of a model based on a given event log and a repair recommendation is any transformation of the model into its repaired version that guarantees that the recommended repairs are fulfilled and the event log fits the repaired model at least as 'good' as it fits the original model; an optimal repair recommendation

in addition requires that the fact that the repair recommendation is fulfilled is evidenced in optimal alignments between traces in the event log and the repaired version of the original model. Finally, a naïve repair of a model is a model transformation that can be used to perform alignment-based repairs. This repair is based on simple model transformation rules, hence the name, and attempts to keep changes to the original model minimal.

In some abstract sense, a model of a system can be seen as a collection of rules that guide occurrences of (observable) activities. In this light, a *repair* of a model of a system S w.r.t. an event log L is a transformation of rules of S that results in a new model S' that modifies the rules of S to allow executing certain activities in those contexts seen in L but not captured in S and skipping (omitting) execution of certain activities in those contexts captured in S but not seen in L. At this level of abstraction, every repair of model S can be characterized by two sets of labels R_{ins} and R_{skp} that represent those activities to insert and those to skip in the process instances of S, respectively, in order to obtain all the process instances of S'. To formally capture this characteristic of a model repair, we introduce the notion of a repair recommendation.

DEFINITION 4.1 (REPAIR RECOMMENDATION).
A *repair recommendation* is a pair $R := (R_{ins}, R_{skp})$, where $R_{ins} \subseteq \mathcal{A}$ and $R_{skp} \subseteq \mathcal{A}$ are finite sets of labels recommended for *insertion* and *skipping*, respectively. ⌟

Note that a repair recommendation does not specify the exact points in the net where repairs must be applied, but captures which issues should be addressed in a model repair exercise.

When solving a model repair problem, one can rely on different measures for assessing fitness between event logs and models, e.g., the cost of optimal alignment, cf. Definition 3.7. In [Fahland and van der Aalst 2015], the authors propose a technique that given an event log L and a net system S uses optimal alignments between traces in L and executions of S to automatically repair S. This technique relies on net system transformations that insert subprocesses, introduce loops, and remove unused or rarely used parts in net systems.

From the point of view of an external observer, a model that is claimed to be a repaired version of a given model w.r.t. some event log should, intuitively, result in alignments between traces in the event log and executions of the repaired model that contain less costly moves on trace and moves on system (defined for observable transitions) than alignments between traces in the event log and executions of the original model; as, indeed, these moves signify deviations between observed and modeled process instances. One can implement the above intuition for the comparison of alignments by comparing alignment costs, if these costs are computed as per some cost function that assigns zero costs to all moves that do not stand for a deviation. The lower the cost of an alignment the better is the match between the observed and modeled process instance. Another useful observation is that deviations between a trace and an execution of a net system stem from the inability of one of the artifacts to mimic an event or activity, i.e., an observed label, of the other artifact and, thus, they are independent of identities of the underlying transition occurrences. Considering the above and for the sake of simplifying the subsequent discussions, it is convenient to lift the concept of the cost function on legal moves from costs of individual transition occurrences to costs of observed labels.

DEFINITION 4.2 (COST OF LABEL MOVES).
A *cost function on label moves* for events $X \subseteq \mathcal{A}$ and activities $Y \subseteq \mathcal{A}$ is a function $f : (X \times \{\gg\}) \cup (\{\gg\} \times Y) \to \mathbb{N}$. The *cost function on legal moves* over a net system $S := (N, M_{ini}, M_{fin}) \in \mathbb{S}$, $N := (P, T, F, \lambda)$, *induced by* a cost function on label moves $f : (X \times \{\gg\}) \cup (\{\gg\} \times Y) \to \mathbb{N}$ is the function $c : \mathbb{M}_S \to \mathbb{N}_0$ such that for every $(x, y) \in \mathbb{M}_S$ it holds that $c(x, y) = f(x, \gg)$ if $x \in X$ and $y = \gg$; $c(x, y) = f(\gg, \lambda(y))$ if $x = \gg$, $y \in T$, and $\lambda(y) \in Y$; otherwise $c(x, y) = 0$. ⌟

The cost function on legal moves induced by a cost function on label moves implements the above intuition by ensuring that costs of all the synchronous moves are equal to zero, and any two moves on system that are defined for transitions that 'carry' the same label have the same cost. The *standard cost function* on label moves for events $X \subseteq \mathcal{A}$ and activities $Y \subseteq \mathcal{A}$ is the function $stdf : (X \times \{\gg\}) \cup (\{\gg\} \times Y) \to \{1\}$. Note that one can employ the standard cost function on label

moves for events and activities $Z := \{\mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{e},\mathtt{f},\mathtt{g},\mathtt{h},\mathtt{x}\}$ to induce the standard cost function on legal moves over the net system in Fig. 2 (under the assumption that $\mathcal{A} = Z$).

Every repair of a model is a transformation, but not every transformation is a repair! We define the notion of *repair* that relies on the (optimal) alignment when assessing fitness as follows.

DEFINITION 4.3 ((OPTIMAL) ALIGNMENT-BASED REPAIR).
A net system $S' := (N, M_{ini}, M_{fin}) \in \mathbb{S}$, $N := (P, T, F, \lambda)$, is the result of an *alignment-based repair* of a net system S w.r.t. an event log $L \in \mathcal{B}(\mathcal{A}^*)$ as per a cost function on label moves f using a repair recommendation $R := (R_{ins}, R_{skp})$ if and only if there exists a function ψ that maps every trace $\upsilon \in L$ to an alignment in $\mathbb{A}^{\upsilon}_{S'}$ such that:

(i) *L fits S' at least as 'good' as it fits S*, i.e., for every trace $\upsilon \in L$ it holds that $(c' \circ \psi)(\upsilon) \leq cost[\upsilon, S, c]$, where c and c' are the cost functions on legal moves over S and S' induced by f, respectively, and

(ii) *S' fulfills R*, i.e., for every trace $\upsilon \in L$ it holds that:
- for every label $x \in R_{ins}$ alignment $\psi(\upsilon)$ contains no move on trace $(x, \gg)$, and
- for every label $y \in R_{skp}$ alignment $\psi(\upsilon)$ contains no move on system from the set $\{(\gg, t) \in \{\gg\} \times T \mid \lambda(t) = y\}$,

Note that S' is the result of an *optimal alignment-based repair* of S if and only if ψ maps every trace $\upsilon \in L$ to an optimal alignment in $\mathbb{O}^{\upsilon}_{S',c'}$. ⌟

We say that function ψ *justifies* the fact that S' is the result of an (optimal) alignment-based repair of S. Clearly, every optimal alignment-based repair is an alignment-based repair, while the converse does not necessarily always hold. An alignment-based repair of a system S transforms S into a repaired system S' in such a way that for every trace $\upsilon \in L$ one can find an alignment between υ and S' which has an alignment cost equal to or lower than the cost of optimal alignment between υ and S, and which contains no moves that are recommended for repair by R. One consequence of the above observations is that every alignment-based repair of a system guarantees that the optimal alignment between the event log and the repaired system is at least as 'good' as the optimal alignment between the event log and the original system.

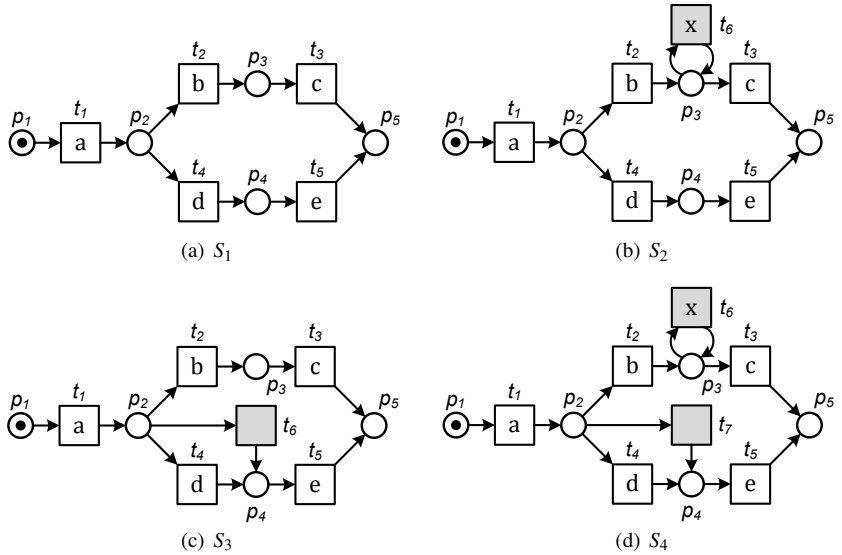
COROLLARY 4.4 (ALIGNMENT-BASED REPAIR).
If a net system $S' \in \mathbb{S}$ is the result of an alignment-based repair of a net system $S \in \mathbb{S}$ w.r.t. an event log $L \in \mathcal{B}(\mathcal{A}^*)$ as per a cost function on label moves f using a repair recommendation, then it holds that $cost[L, S', c'] \leq cost[L, S, c]$, where c and c' are the cost functions on legal moves over S and S' induced by f, respectively. ⌟

The proof of Corollary 4.4 can be found in Appendix A.

Two elementary repairs on net systems proposed in [Fahland and van der Aalst 2015] are the self-loop injection and the skip injection.

DEFINITION 4.5 (SELF-LOOP AND SKIP INJECTIONS).
Let $S := (N, M_{ini}, M_{fin}) \in \mathbb{S}$, $N := (P, T, F, \lambda)$, be a net system.
- A net system $S' := (N', M_{ini}, M_{fin}) \in \mathbb{S}$, $N' := (P, T', F', \lambda')$, is the result of *self-loop injections* in S for a label $a \in \mathcal{A}$ at places in $X \subseteq P$ if and only if there exists a set Y, for which it holds that $(P \cup T) \cap Y = \emptyset$, and a bijective function $f : X \to Y$ such that $T' = T \cup Y$, $F' = F \cup f \cup f^{-1}$, and $\lambda' = \lambda \cup \{(y, a) \mid y \in Y\}$.
- A net system $S' := (N', M_{ini}, M_{fin}) \in \mathbb{S}$, $N' := (P, T', F', \lambda')$, is the result of *skip injections* in S for transitions in $X \subseteq T$ if and only if there exists a set Y, for which it holds that $(P \cup T) \cap Y = \emptyset$, and a bijective function $f : X \to Y$ such that $T' = T \cup Y$, $F' = F \cup \{(p, y) \in P \times Y \mid \exists x \in X : p \in {\bullet x} \wedge f(x) = y\} \cup \{(y, p) \in Y \times P \mid \exists x \in X : f(x) = y \wedge p \in x{\bullet}\}$, and $\lambda' = \lambda \cup \{(y, \tau) \mid y \in Y\}$. ⌟

The system in Fig. 3(b) is the result of self-loop injections in the system in Fig. 3(a) for label $\mathtt{x}$ at places in $\{p_3\}$, while the system in Fig. 3(d) is the result of skip injections in the system in Fig. 3(b) for transitions in $\{t_4\}$. It is easy to see that different orders of the same repertoire of injections lead to the same resulting net system. For example, one indeed obtains the net system in Fig. 3(d) as the



(a) S_1 (b) S_2

(c) S_3 (d) S_4

Fig. 3: Four net systems: (a) S_1, (b) S_2 is the result of self-loop injections in S_1 for label x at places in $\{p_3\}$, (c) S_3 is the result of skip injections in S_1 for transitions in $\{t_4\}$, and (d) S_4 is the result of skip injections in S_2 for transitions in $\{t_4\}$.

result of self-loop injections in the net system in Fig. 3(c) for label x at places in $\{p_3\}$, where the net system in Fig. 3(c) is the result of skip injections in the net system in Fig. 3(a) for transitions in $\{t_4\}$. Note that all the 'injected' transitions are highlighted with grey background in Figs. 3(a)–3(d).

One can repair a net system S w.r.t. a given event log L by injecting (i) self-loops for those labels for which there exist moves on trace and (ii) skips for those observable transitions of S for which there exist moves on system in optimal alignments between traces in L and S, cf. [Fahland and van der Aalst 2015]. However, it is a challenging task to decide which injections (out of all possible ones) to introduce as many of them may be redundant. The approach we take in this work to decide which self-loop and/or skip injections to apply in a given net system requires the reader to become familiar with the notion of a minimal hitting set. Given a collection of sets C, a set which intersects all sets in C in at least one element is called a *hitting set* of C. Given the set of sets $D := \{\{1,2,5\},\{2,4,5\},\{2,3\}\}$, the sets $\{2\}$, $\{3,5\}$, and $\{1,3,4\}$ are examples of hitting sets of D. A hitting set of smallest size is known as a *minimal hitting set*. By *mhs*, we denote a function that maps every collection of sets to one of its minimal hitting sets. For example, it holds that $mhs(D) = \{2\}$. The problem of finding a minimal hitting set is equivalent to the *set covering* problem, which is shown to be NP-complete in [Karp 1972]. This problem can be solved using the technique proposed in [Wotawa 2001].

We are now in a position to propose the notion of a naïve repair of a net system.

DEFINITION 4.6 (NAÏVE REPAIR).
A net system $S' \in \mathbb{S}$ is the result of a *naïve repair* of a net system $S := (N, M_{ini}, M_{fin}) \in \mathbb{S}$, $N := (P,T,F,\lambda)$, based on the alignments in the image of a function ϕ, which maps every trace υ in an event log $L \in \mathcal{B}(\mathcal{A}^*)$ to some alignment in $\mathbb{A}_S^{\upsilon}$, using a repair recommendation $R := (R_{ins}, R_{skp})$ if and only if there exist two sequences of labels $\alpha := \langle \alpha_1, \alpha_2, \ldots, \alpha_n \rangle$ and $\beta := \langle \beta_1, \beta_2, \ldots, \beta_m \rangle$, for which it holds that $Set(\alpha) = R_{ins}$, $|\alpha| = |R_{ins}|$, $Set(\beta) = R_{skp}$, and $|\beta| = |R_{skp}|$, and a sequence of systems $\langle S_0, S_1, \ldots, S_n, S_{n+1}, \ldots, S_{n+m} \rangle$ such that $S_0 = S$, $S_{n+m} = S'$, for every S_i, $i \in [1..n]$, it holds that S_i is the result of self-loop injections in S_{i-1} for label α_i at places in $mhs(\{X \subseteq P \mid \exists M \in \mathcal{B}(P)\ \exists \upsilon \in L\ \exists k \in [1..|\phi(\upsilon)|] : X = Set(M) \wedge (N, M_{ini})[((\pi_2 \circ \phi)(\upsilon))^{[k-1]}|_T\rangle(N,M) \wedge (\phi(\upsilon))_k = (\alpha_i, \gg)\})$, and for every S_{n+j}, $j \in [1..m]$ it holds that it is the result of skip injections in S_{n+j-1} for transitions in $\{t \in T \mid \exists \upsilon \in L\ \exists k \in [1..|\phi(\upsilon)|] : (\phi(\upsilon))_k = (\gg, t) \wedge \lambda(t) = \beta_j\}$.

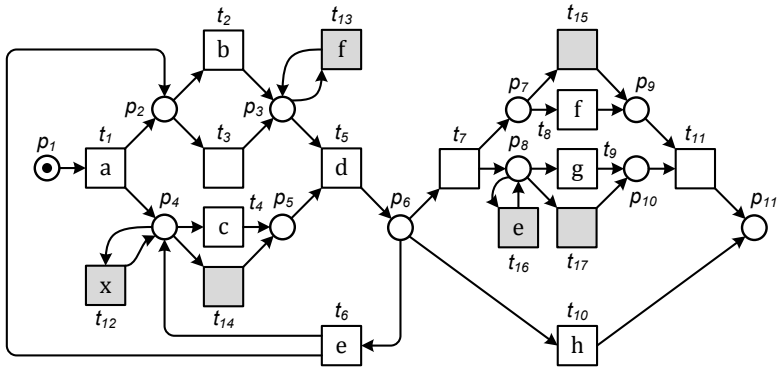


Fig. 4: A net system obtained as the result of a naïve repair of the net system in Fig. 2 using repair recommendation $R := (\{\mathtt{e},\mathtt{f},\mathtt{x}\},\{\mathtt{c},\mathtt{f},\mathtt{g}\})$.

Note that one can replace the order of injections employed in the above definition with any other order, as it does not influence the end result.

Fig. 4 shows a net system that is obtained as the result of a naïve repair of the net system $S := (N, M_{ini}, M_{fin})$, $N := (P, T, F, \lambda)$, in Fig. 2 using repair recommendation $R := (\{\mathtt{e},\mathtt{f},\mathtt{x}\},\{\mathtt{c},\mathtt{f},\mathtt{g}\})$. The repair is based on the alignments in the image of function ϕ that maps traces in event log L_3 from Section 2.3 to *optimal* alignments (as per the standard cost function on legal moves over S) between these traces and S as follows. Let $\upsilon_1 := \langle \mathtt{a},\mathtt{b},\mathtt{c},\mathtt{f},\mathtt{d},\mathtt{e},\mathtt{f}\rangle$, $\upsilon_2 := \langle \mathtt{a},\mathtt{c},\mathtt{d},\mathtt{c},\mathtt{e},\mathtt{d},\mathtt{g},\mathtt{f}\rangle$, $\upsilon_3 := \langle \mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{e},\mathtt{x},\mathtt{c},\mathtt{h},\mathtt{a}\rangle$, $\upsilon_4 := \langle \mathtt{c},\mathtt{d},\mathtt{d},\mathtt{f},\mathtt{e},\mathtt{g}\rangle$, $\upsilon_5 := \langle \mathtt{a},\mathtt{b}\rangle$, $\upsilon_6 := \langle \mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{e},\mathtt{d},\mathtt{f}\rangle$, and $\upsilon_7 := \langle \mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{e},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{g}\rangle$. Then, $\phi := \{(\upsilon_1,\gamma_0^1),(\upsilon_2,\gamma_0^2),(\upsilon_3,\gamma_0^3),(\upsilon_4,\gamma_0^4),(\upsilon_5,\gamma_0^5),(\upsilon_6,\gamma_0^6),(\upsilon_7,\gamma_0^7)\}$, where the optimal alignments γ_0^1–γ_0^7 are listed below.

$\gamma_0^1 :=$

a	b	c	f	d	≫	e	f	≫	≫
a	b	c	≫	d	τ	≫	f	g	τ
t_1	t_2	t_4		t_5	t_7		t_8	t_9	t_{11}

$\gamma_0^2 :=$

a	c	≫	d	c	e	≫	≫	d	≫	g	f	≫
a	c	τ	d	≫	e	τ	c	d	τ	g	f	τ
t_1	t_4	t_3	t_5		t_6	t_3	t_4	t_5	t_7	t_9	t_8	t_{11}

$\gamma_0^3 :=$

a	b	c	d	e	x	c	≫	≫	h	a
a	b	c	d	e	≫	c	τ	d	h	≫
t_1	t_2	t_4	t_5	t_6		t_4	t_3	t_5	t_{10}	

$\gamma_0^4 :=$

≫	c	≫	d	≫	d	f	e	g	≫
a	c	τ	d	τ	≫	f	≫	g	τ
t_1	t_4	t_3	t_5	t_7		t_8		t_9	t_{11}

$\gamma_0^5 :=$

a	b	≫	≫	≫
a	b	c	d	h
t_1	t_2	t_4	t_5	t_{10}

$\gamma_0^6 :=$

a	b	c	d	e	≫	≫	d	≫	f	≫	≫
a	b	c	d	e	τ	c	d	τ	f	g	τ
t_1	t_2	t_4	t_5	t_6	t_3	t_4	t_5	t_7	t_8	t_9	t_{11}

$\gamma_0^7 :=$

a	b	c	d	e	b	c	d	≫	g	≫	≫
a	b	c	d	e	b	c	d	τ	g	f	τ
t_1	t_2	t_4	t_5	t_6	t_2	t_4	t_5	t_7	t_9	t_8	t_{11}

The fact that the net system in Fig. 4 is the result of a naïve repair of S is justified by the existence of two sequences of labels $\alpha := \langle \mathtt{e},\mathtt{f},\mathtt{x}\rangle$ and $\beta := \langle \mathtt{c},\mathtt{f},\mathtt{g}\rangle$ and one sequence of net systems $\langle S_0, S_1, S_2, S_3, S_4, S_5, S_6\rangle$, where $S_0 = S$, S_6 is the net system in Fig. 4, S_1 is the result of self-loop injections in S_0 for label $\mathtt{e}$ at places in $\{p_8\}$, S_2 is the result of self-loop injections in S_1 for label $\mathtt{f}$ at places in $\{p_3\}$, S_3 is the result of self-loop injections in S_2 for label $\mathtt{x}$ at places in $\{p_4\}$, S_4 is the result of skip injections in S_3 for transitions in $\{t_4\}$, S_5 is the result of skip injections in S_4 for transitions in $\{t_8\}$, and finally S_6 is the result of skip injections in S_5 for transitions in $\{t_9\}$.

The places and transitions at which self-loop and for which skip injections, respectively, are performed in S, were identified based on the alignments γ_0^1–γ_0^7. For example, S_1 is obtained from S_0

as the result of self-loop injections at places in $\{p_8\}$ because it holds that $mhs(\{\{p_7,p_8\},\{p_8,p_9\}\}) = \{p_8\}$, where $\{p_7,p_8\}$ and $\{p_8,p_9\}$ are places that are 'marked', i.e., contain at least one token, in the markings $[p_7,p_8]$ and $[p_8,p_9]$, which are the markings that are reachable from M_{ini} via occurrence sequences $\sigma_1 := \langle t_1,t_2,t_4,t_5,t_7\rangle$ and $\sigma_2 := \langle t_1,t_4,t_3,t_5,t_7,t_8\rangle$ in N, respectively. Note that it holds that $\sigma_1 = ((\pi_2 \circ \phi)(\upsilon_1))^{[6]}|_T$ and $\sigma_2 = ((\pi_2 \circ \phi)(\upsilon_4))^{[7]}|_T$; the 6-th and 7-th prefix are used because γ_0^1 and γ_0^4 contain move on trace $(\mathtt{e},\gg)$ at positions 7 and 8, respectively. Note also that S_4 is the result of skip injections in S_3 for transitions in $\{t_4\}$ because of the existence of trace υ_2 which contains move on system $(\gg,t_4)$ such that $\lambda(t_4) = \mathtt{c}$, and there exists no other transition $t \in T$, $t \neq t_4$, for which some of the seven alignments proposed above contains move on system $(\gg,t)$ and $\lambda(t) = \mathtt{c}$.

The above examples reveal the rationale behind the design of the notion of a naïve repair, which is to perform the least number of self-loop and/or skip injections possible for every move on trace and/or every move on system observed in the given collection of alignments and defined on labels that are recommended for repair. The injections guarantee that moves on trace and/or moves on system that are requested to be repaired can be mimicked by the repaired net system and/or trace, respectively, in the alignments that follow *similar* paths through the repaired net system as the alignments that were used to perform the naïve repair.

Let c be the cost function on legal moves over the net system S shown in Fig. 2 induced by the standard cost function *stdf* on label moves for events $Z := \{\mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{e},\mathtt{f},\mathtt{g},\mathtt{h},\mathtt{x}\}$ and activities $Z \setminus \{\mathtt{x}\}$. Then, c is the standard cost function on legal moves over system S for which it holds that $c = (Z \times \{\gg\}) \times \{1\} \cup (\{\gg\} \times \{t_3,t_7,t_{11}\}) \times \{0\} \cup (\{\gg\} \times (T \setminus \{t_3,t_7,t_{11}\})) \times \{1\} \cup \{((\mathtt{a},t_1),0),((\mathtt{b},t_2),0),((\mathtt{c},t_4),0),((\mathtt{d},t_5),0),((\mathtt{e},t_6),0),((\mathtt{f},t_8),0),((\mathtt{g},t_9),0),((\mathtt{h},t_{10}),0)\}$. Using the optimal alignments γ_0^1–γ_0^7 proposed above, the reader can now verify that $cost[\upsilon_1,S,c] = 3$, $cost[\upsilon_2,S,c] = 2$, $cost[\upsilon_3,S,c] = 3$, $cost[\upsilon_4,S,c] = 3$, $cost[\upsilon_5,S,c] = 3$, $cost[\upsilon_6,S,c] = 2$, and $cost[\upsilon_7,S,c] = 1$. Thus, $cost[L_3,S,c] = 10 \times 3 + 9 \times 2 + 9 \times 3 + 7 \times 3 + 6 \times 3 + 2 \times 2 + 2 \times 1 = 120$.

Let c' be the cost function on legal moves over the net system S' in Fig. 4 induced by *stdf*. Then, c' is the standard cost function on legal moves over S' for which it holds that $c' = c \cup (\{\gg\} \times \{t_{14},t_{15},t_{17}\}) \times \{0\} \cup (\{\gg\} \times \{t_{12},t_{13},t_{16}\}) \times \{1\} \cup \{((\mathtt{x},t_{12}),0),((\mathtt{f},t_{13}),0),((\mathtt{e},t_{16}),0)\}$. The reader can find an *optimal* alignment between every trace in L_3 and S' as per cost function c' listed below.

$$\gamma_1^1 := \begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{f}&\mathtt{d}&\gg&\mathtt{e}&\mathtt{f}&\gg&\gg\\ \hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{f}&\mathtt{d}&\tau&\mathtt{e}&\mathtt{f}&\tau&\tau\\ t_1&t_2&t_4&t_{13}&t_5&t_7&t_{16}&t_8&t_{17}&t_{11}\\ \hline\end{array} \qquad \gamma_1^2 := \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{c}&\gg&\mathtt{d}&\mathtt{c}&\mathtt{e}&\gg&\gg&\mathtt{d}&\gg&\mathtt{g}&\mathtt{f}&\gg\\ \hline \mathtt{a}&\mathtt{c}&\tau&\mathtt{d}&\gg&\mathtt{e}&\tau&\tau&\mathtt{d}&\tau&\mathtt{g}&\mathtt{f}&\tau\\ t_1&t_4&t_3&t_5&&t_6&t_3&t_{14}&t_5&t_7&t_9&t_8&t_{11}\\ \hline\end{array}$$

$$\gamma_1^3 := \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\mathtt{x}&\mathtt{c}&\gg&\gg&\mathtt{h}&\mathtt{a}\\ \hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\mathtt{x}&\mathtt{c}&\tau&\mathtt{d}&\mathtt{h}&\gg\\ t_1&t_2&t_4&t_5&t_6&t_{12}&t_4&t_3&t_5&t_{10}&\\ \hline\end{array} \qquad \gamma_1^4 := \begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline \gg&\mathtt{c}&\gg&\mathtt{d}&\gg&\mathtt{d}&\mathtt{f}&\mathtt{e}&\mathtt{g}&\gg\\ \hline \mathtt{a}&\mathtt{c}&\tau&\mathtt{d}&\tau&\gg&\mathtt{f}&\mathtt{e}&\mathtt{g}&\tau\\ t_1&t_4&t_3&t_5&t_7&&t_8&t_{16}&t_9&t_{11}\\ \hline\end{array}$$

$$\gamma_1^5 := \begin{array}{|c|c|c|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{b}&\gg&\gg&\gg&\gg&\gg&\gg\\ \hline \mathtt{a}&\mathtt{b}&\tau&\mathtt{d}&\tau&\tau&\tau&\tau\\ t_1&t_2&t_{14}&t_5&t_7&t_{15}&t_{17}&t_{11}\\ \hline\end{array} \qquad \gamma_1^6 := \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\gg&\gg&\mathtt{d}&\gg&\mathtt{f}&\gg&\gg\\ \hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\tau&\tau&\mathtt{d}&\tau&\mathtt{f}&\tau&\tau\\ t_1&t_2&t_4&t_5&t_6&t_3&t_{14}&t_5&t_7&t_8&t_{17}&t_{11}\\ \hline\end{array}$$

$$\gamma_1^7 := \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\gg&\mathtt{g}&\gg&\gg\\ \hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\tau&\mathtt{g}&\tau&\tau\\ t_1&t_2&t_4&t_5&t_6&t_2&t_4&t_5&t_7&t_9&t_{15}&t_{11}\\ \hline\end{array}$$

Using the optimal alignments γ_1^1–γ_1^7 shown above, the reader can easily verify that indeed $cost[\upsilon_1,S',c'] = 0$, $cost[\upsilon_2,S',c'] = 1$, $cost[\upsilon_3,S',c'] = 2$, $cost[\upsilon_4,S',c'] = 2$, $cost[\upsilon_5,S',c'] = 1$, $cost[\upsilon_6,S',c'] = 0$, and $cost[\upsilon_7,S',c'] = 0$. Thus, it holds that $cost[L_3,S',c'] = 10 \times 0 + 9 \times 1 + 9 \times 2 + 7 \times 2 + 6 \times 1 + 2 \times 0 + 2 \times 0 = 47$.

Six out of seven alignments γ_1^1–γ_1^7 proposed above follow paths in the repaired model that are *similar* to those followed by alignments γ_0^1–γ_0^7 in the original model. For example, alignment

γ_1^1 follows a path that is similar to that followed by γ_0^1 as one can construct alignment γ_1^1 from alignment γ_0^1 by replacing moves $(\mathtt{f},\gg)$, $(\mathtt{e},\gg)$, and $(\gg,t_9)$, with moves $(\mathtt{f},t_{13})$, $(\mathtt{e},t_{16})$, and $(\gg,t_{17})$, respectively, i.e., with moves that are defined for the transitions injected in the repaired model. Note that alignment γ_1^5 does not follow the path of alignment γ_0^5 as, due to the repair, there exists a more optimal alignment that follows the path via injected transitions t_{15} and t_{17}.

Let $S' \in \mathbb{S}$ be the result of a naïve repair of a net system $S \in \mathbb{S}$ based on a set of alignments Γ. Then, for every alignment $\gamma \in \Gamma$ there exists a unique alignment that follows a path in S' that is similar to the path followed by γ in S. We denote this alignment by $similar[\gamma, S, S']$. For example, it holds that $similar[\gamma_0^1, S, S'] = \gamma_1^1$, where S and S' are net systems in Fig. 2 and Fig. 4, respectively.

The net system $S' := (N', M_{ini}, M_{fin})$, $N' := (P, T', F', \lambda')$, in Fig. 4 is the result of an *optimal* alignment-based repair of the net system S in Fig. 2, cf. Definition 4.3. Let $\psi := \{(\upsilon_1, \gamma_1^1), (\upsilon_2, \gamma_1^2), (\upsilon_3, \gamma_1^3), (\upsilon_4, \gamma_1^4), (\upsilon_5, \gamma_1^5), (\upsilon_6, \gamma_1^6), (\upsilon_7, \gamma_1^7)\}$ be a function that maps traces to alignments. Then, one can employ function ψ to justify that: (i) L_3 fits S' at least as 'good' as it fits S (and in fact strictly 'better' than it fits S) as obviously $(c' \circ \psi)(\upsilon_1) < cost[\upsilon_1, S, c]$ (because $0 < 3$), $(c' \circ \psi)(\upsilon_2) < cost[\upsilon_2, S, c]$ $(1 < 2)$, $(c' \circ \psi)(\upsilon_3) < cost[\upsilon_3, S, c]$ $(2 < 3)$, $(c' \circ \psi)(\upsilon_4) < cost[\upsilon_4, S, c]$ $(2 < 3)$, $(c' \circ \psi)(\upsilon_5) < cost[\upsilon_5, S, c]$ $(1 < 3)$, $(c' \circ \psi)(\upsilon_6) < cost[\upsilon_6, S, c]$ $(0 < 2)$, and finally $(c' \circ \psi)(\upsilon_7) < cost[\upsilon_7, S, c]$ $(0 < 1)$; note that as one of the consequences of the above observations it must hold that $cost[L_3, S', c'] \leq cost[L_3, S, c]$, cf. Corollary 4.4. In fact, it holds that $cost[L_3, S', c'] < cost[L_3, S, c]$ as, indeed, 47 is less than 120. (ii) S' fulfills repair recommendation $(\{\mathtt{e},\mathtt{f},\mathtt{x}\}, \{\mathtt{c},\mathtt{f},\mathtt{g}\})$, because for every trace $\upsilon \in L_3$ it holds that $\psi(\upsilon)$ contains no move $(x, \gg)$, $x \in \{\mathtt{e},\mathtt{f},\mathtt{x}\}$, and no move $(\gg, y)$, $y \in T'$, such that $\lambda'(y) \in \{\mathtt{c},\mathtt{f},\mathtt{g}\}$.

Not every naïve repair results in an optimal alignment-based repair of the system, even if performed based on optimal alignments. We demonstrate this using the system S_1 in Fig. 3(a), event log $L := [\langle \mathtt{a},\mathtt{b},\mathtt{x},\mathtt{e}\rangle, \langle \mathtt{a},\mathtt{e}\rangle]$, cost function on label moves $f := \{((\mathtt{a},\gg),1), ((\mathtt{b},\gg),1), ((\mathtt{e},\gg),1), ((\mathtt{x},\gg),1), ((\gg,\mathtt{a}),1), ((\gg,\mathtt{b}),1), ((\gg,\mathtt{c}),2), ((\gg,\mathtt{d}),3), ((\gg,\mathtt{e}),1), ((\gg,\mathtt{x}),1)\}$, and repair recommendation $R := (\{\mathtt{x}\}, \{\mathtt{d}\})$. The optimal alignments between traces in L and S_1 as per the cost function c on legal moves over S_1 induced by f are given below; it holds that $c = \{((\mathtt{a},\gg),1), ((\mathtt{b},\gg),1), ((\mathtt{e},\gg),1), ((\mathtt{x},\gg),1), ((\gg,t_1),1), ((\gg,t_2),1), ((\gg,t_3),2), ((\gg,t_4),3), ((\gg,t_5),1), ((\mathtt{a},t_1),0), ((\mathtt{b},t_2),0), ((\mathtt{c},t_3),0), ((\mathtt{d},t_4),0), ((\mathtt{e},t_5),0)\}$.

$$\gamma_2^1 := \begin{array}{|c|c|c|c|c|} \hline \mathtt{a} & \mathtt{b} & \mathtt{x} & \gg & \mathtt{e} \\ \hline \mathtt{a} & \mathtt{b} & \gg & \mathtt{c} & \gg \\ t_1 & t_2 & & t_3 & \\ \hline \end{array} \qquad \gamma_2^2 := \begin{array}{|c|c|c|} \hline \mathtt{a} & \gg & \mathtt{e} \\ \hline \mathtt{a} & \mathtt{d} & \mathtt{e} \\ t_1 & t_4 & t_5 \\ \hline \end{array}$$

Alignment γ_2^1 is one out of three optimal alignments between trace $\langle \mathtt{a},\mathtt{b},\mathtt{x},\mathtt{e}\rangle$ and S_1, while γ_2^2 is the only optimal alignment between trace $\langle \mathtt{a},\mathtt{e}\rangle$ and S_1. It holds that $c(\gamma_2^1) = 4$ and $c(\gamma_2^2) = 3$. The result of a naïve repair of S_1 based on optimal alignments γ_2^1 and γ_2^2 using repair recommendation R is the net system S_4 in Fig. 3(d). Let c' be the cost function on legal moves over S_4 induced by f. Then, it holds that $c' = c \cup \{((\mathtt{x},t_6),0), ((\gg,t_7),0)\}$. The optimal alignments between the two traces in event log L and net system S_4 as per cost function c' are listed below.

$$\gamma_3^1 := \begin{array}{|c|c|c|c|c|} \hline \mathtt{a} & \gg & \mathtt{b} & \mathtt{x} & \mathtt{e} \\ \hline \mathtt{a} & \tau & \gg & \gg & \mathtt{e} \\ t_1 & t_7 & & & t_5 \\ \hline \end{array} \qquad \gamma_3^2 := \begin{array}{|c|c|c|} \hline \mathtt{a} & \gg & \mathtt{e} \\ \hline \mathtt{a} & \tau & \mathtt{e} \\ t_1 & t_7 & t_5 \\ \hline \end{array}$$

Using these alignments, one can verify that $c'(\gamma_3^1) = 2$ and $c'(\gamma_3^2) = 0$. Thus, function $\psi := \{(\langle \mathtt{a},\mathtt{b},\mathtt{x},\mathtt{e}\rangle, \gamma_3^1), (\langle \mathtt{a},\mathtt{e}\rangle, \gamma_3^2)\}$ justifies the fact that L fits S_4 at least as 'good' as (and in fact 'better' than) it fits S_1. However, ψ does not justify the fact that S_4 fulfills R; note that alignment γ_3^1 contains move on trace $(\mathtt{x}, \gg)$ which is recommended for repair by R. Nevertheless, net system S_4 is the result of an alignment-based repair of S_1. This is justified by function $\psi' := \{(\langle \mathtt{a},\mathtt{b},\mathtt{x},\mathtt{e}\rangle, \gamma), (\langle \mathtt{a},\mathtt{e}\rangle, \gamma_3^2)\}$,

where $\gamma := \langle (\mathtt{a}, t_1), (\mathtt{b}, t_2), (\mathtt{x}, t_6), (\gg, t_3), (\mathtt{e}, \gg) \rangle$, $c'(\gamma) = 3$, is a *non-optimal* alignment between trace $\langle \mathtt{a}, \mathtt{b}, \mathtt{x}, \mathtt{e} \rangle$ and S_4; note that γ follows a path that is *similar* to the path of γ_2^1.

In fact, every naïve repair of a given net system based on a collection of optimal alignments always results in an alignment-based repair; one can justify this fact using a function that maps traces to alignments that follow *similar* paths as those followed by the optimal alignments. This observation is crystallized in the next statement, the proof of which can be found in Appendix A.

LEMMA 4.7 (ALIGNMENT-BASED REPAIR).
If a net system $S' \in \mathbb{S}$ is the result of a naïve repair of a net system $S \in \mathbb{S}$ based on the alignments in the image of a function ϕ, which maps every trace υ in an event log $L \in \mathcal{B}(\mathcal{A}^*)$ to an optimal alignment in $\mathbb{O}^{\upsilon}_{S,c}$, using a repair recommendation R, where c is the cost function on legal moves over S induced by a cost function on label moves f, then S' is the result of an alignment-based repair of S w.r.t. L as per f using R. ⌟

5. OPTIMAL MODEL REPAIR RECOMMENDATION

In this section, we define and motivate a new problem in process mining—the problem of *optimal model repair recommendation*, which refers to the following problem: Given a model of a distributed system S, e.g., a net system, and an event log L that does not fit S, among all possible repair recommendations, cf. Definition 4.1, that satisfy some given constraint C find a repair recommendation R such that a model S' obtained by repairing S using (and fulfilling) R is such that L fits S' at least as 'good' as it fits any other model obtained by repairing S using any other repair recommendation that satisfies C. Note that finding a solution to an optimal model repair recommendation problem is not about constructing a repaired version of a given model, but is rather about identifying a repair recommendation that when implemented in the given model, e.g., as some transformation of the model, will have the best *impact* on fitness. Furthermore, this section demonstrates that naïve repairs can be used to implement *optimal* alignment-based repairs when are based on optimal alignments as per a special cost function on legal moves. Finally, this section introduces the notion of an optimal repair recommendation, i.e., a recommendation that when used to perform an optimal alignment-based repair of a model leads to a repaired model that has the 'best' fit with the given event log.

The problem of optimal model repair recommendation is an optimization problem of finding the best repair recommendation from all feasible repair recommendations. One can often assess the feasibility of a solution to an optimization problem by associating it with a numeric value and comparing this value with a given threshold. We rely on the concept of a *repair constraint* to induce the set of all feasible repair recommendations.

DEFINITION 5.1 (REPAIR CONSTRAINT).
A *repair constraint* is a triple $C := (C_{ins}, C_{skp}, res)$, where $C_{ins} : \mathcal{A} \to \mathbb{N}_0$ and $C_{skp} : \mathcal{A} \to \mathbb{N}_0$ are functions that assign costs of *inserting* and *skipping* activities to labels, respectively, and $res \in \mathbb{N}_0$ is the amount of available *repair resources*. ⌟

Functions C_{ins} and C_{skp} specify the costs of including labels into a repair recommendation. A repair constraint is *standard* iff both images of C_{ins} and C_{skp} are equal to $\{1\}$.

Given a repair recommendation $R := (R_{ins}, R_{skp})$ and a repair constraint $C := (C_{ins}, C_{skp}, res)$, by $cost[R, C]$, we denote the *cost of R as per repair constraint C*, which is defined as $cost[R, C] := \sum_{r \in R_{ins}} C_{ins}(r) + \sum_{r \in R_{skp}} C_{skp}(r)$. Then, R is a *feasible* solution to an optimal model repair recommendation problem restricted by C if it holds that its cost does not exceed available repair resources, i.e., $cost[R, C] \leq res$. Thus, one can rely on the notion of a repair constraint to formulate an optimal model repair recommendation problem to search for a repair recommendation within the available repair resources that has the best impact on fitness between the event log and the repaired model.

We have observed that one can simulate the effect of an alignment-based repair of a net system $S \in \mathbb{S}$ w.r.t. an event log $L \in \mathcal{B}(\mathcal{A}^*)$ as per some cost function on label moves f using a repair recommendation R without performing actual transformations on S. Indeed, by setting costs of certain moves in the cost function on legal moves induced by f to be equal to zero one eliminates the

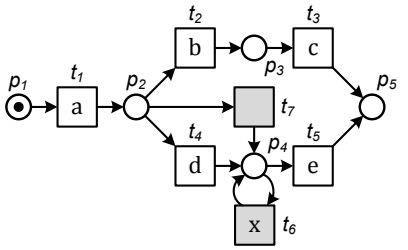


Fig. 5: Net system S_5.

effect of these moves on the costs of optimal alignments between traces in L and S in a similar way as if S has been transformed to cater for these moves. Consequently, in order to simulate the effect of performing an alignment-based repair of S using R, one needs to 'adjust' the costs of all the moves that are recommended for repair by R. This can be accomplished using the notion of the adjusted cost function proposed below.

DEFINITION 5.2 (ADJUSTED COST OF LABEL MOVES).
The *adjusted cost function on label moves* induced by a cost function on label moves f for events $X \subseteq \mathcal{A}$ and activities $Y \subseteq \mathcal{A}$ using a repair recommendation $R := (R_{ins}, R_{skp})$ is the restriction of f to $((X \setminus R_{ins}) \times \{\gg\}) \cup (\{\gg\} \times (Y \setminus R_{skp}))$, denoted by $f|_R$. ⌟

Let $L \in \mathcal{B}(\mathcal{A}^*)$ be an event log, let R be a repair recommendation, and let f be a cost function on label moves. An optimal alignment between a trace $\upsilon \in L$ and a net system $S \in \mathbb{S}$ as per the cost function c on legal moves over S induced by $f|_R$ hints at the fact that one can use R to repair S into a fresh net system S' such that the cost of optimal alignment between υ and S' as per the cost function on legal moves over S' induced by f is as good as the cost of optimal alignment between υ and S as per c. The next statement justifies this fact by pointing to one such repair.

THEOREM 5.3 (OPTIMAL ALIGNMENT-BASED REPAIR).
If a net system $S' \in \mathbb{S}$ is the result of a naïve repair of a net system $S \in \mathbb{S}$ based on the alignments in the image of a function ϕ using a repair recommendation R, where ϕ maps every trace υ in an event log $L \in \mathcal{B}(\mathcal{A}^*)$ to an alignment in $\mathbb{O}^{\upsilon}_{S,c}$, c is the cost function on legal moves over S induced by $f|_R$, f is a cost function on label moves, then S' is the result of an optimal alignment-based repair of S w.r.t. L as per f using R. ⌟

The proof of Theorem 5.3 can be found in Appendix A.

We exemplify implications of Theorem 5.3 by extending the discussion of the example proposed at the end of Section 4. Let $f := \{((\mathtt{a},\gg),1),((\mathtt{b},\gg),1),((\mathtt{e},\gg),1),((\mathtt{x},\gg),1),((\gg,\mathtt{a}),1),((\gg,\mathtt{b}),1),((\gg,\mathtt{c}),2),((\gg,\mathtt{d}),3),((\gg,\mathtt{e}),1),((\gg,\mathtt{x}),1)\}$ be a cost function on label moves. Net system S_5 in Fig. 5 is the result of a naïve repair of net system S_1 in Fig. 3(a) based on the alignments in the image of function $\phi := \{(\langle \mathtt{a},\mathtt{b},\mathtt{x},\mathtt{e}\rangle, \gamma_4^1), (\langle \mathtt{a},\mathtt{e}\rangle, \gamma_4^2)\}$ using repair recommendation $R := (\{\mathtt{x}\},\{\mathtt{d}\})$; the 'injected' transitions are highlighted with grey background in the figure.

$$\gamma_4^1 := \begin{array}{|c|c|c|c|c|} \hline \mathtt{a} & \gg & \mathtt{b} & \mathtt{x} & \mathtt{e} \\ \hline \mathtt{a} & \mathtt{d} & \gg & \gg & \mathtt{e} \\ t_1 & t_4 & & & t_5 \\ \hline \end{array} \qquad \gamma_4^2 := \begin{array}{|c|c|c|} \hline \mathtt{a} & \gg & \mathtt{e} \\ \hline \mathtt{a} & \mathtt{d} & \mathtt{e} \\ t_1 & t_4 & t_5 \\ \hline \end{array}$$

Alignments γ_4^1 and γ_4^2 shown above are two optimal alignments constructed between two traces $\langle \mathtt{a},\mathtt{b},\mathtt{x},\mathtt{e}\rangle$ and $\langle \mathtt{a},\mathtt{e}\rangle$ and the system S_1 in Fig. 3(a) as per the cost function c on legal moves over S_1 induced by $f|_R$. Note that it holds that $f|_R$ equals $\{((\mathtt{a},\gg),1),((\mathtt{b},\gg),1),((\mathtt{e},\gg),1),((\gg,\mathtt{a}),1),((\gg,\mathtt{b}),1),((\gg,\mathtt{c}),2),((\gg,\mathtt{e}),1),((\gg,\mathtt{x}),1)\}$. Thus, according to Definition 4.2, $c = \{((\mathtt{a},\gg),1),((\mathtt{b},\gg),1),((\mathtt{e},\gg),1),((\mathtt{x},\gg),0),((\gg,t_1),1),((\gg,t_2),1),((\gg,t_3),2),((\gg,t_4),0),((\gg,t_5),1),((\mathtt{a},t_1),0),((\mathtt{b},t_2),0),((\mathtt{c},t_3),0),((\mathtt{d},t_4),0),((\mathtt{e},t_5),0)\}$, and $c(\gamma_4^1) = 1$ and $c(\gamma_4^2) = 0$.

Note that net system S_5 in Fig. 5 is, indeed, the result of a naïve repair of $S_1 := (N, M_{ini}, M_{fin})$, $N := (P, T, F, \lambda)$, as it is the result of skip injections for transitions in $\{t_4\}$ in the net system that, in turn, is obtained as the result of self-loop injections in S_1 for label x at places in $\{p_4\}$. The self-loop injections for label x are performed at places in $\{p_4\}$ because there exist marking $[p_4]$ and only one alignment γ_4^1 in the image of ϕ with only one move on trace $(\mathtt{x}, \gg)$ at position 4 such that $\pi_2(\gamma_4^1)^{[3]}|_T = \langle t_1, t_4 \rangle$, $(N, M_{ini})[\langle t_1, t_4 \rangle\rangle (N, [p_4])$, $Set([p_4]) = \{p_4\}$, and $mhs(\{\{p_4\}\}) = \{p_4\}$. The skip injections are performed for transitions in $\{t_4\}$, because the two moves on system $(\gg, t_4)$, $\lambda(t_4) = \mathtt{d}$, both at position 2 in optimal alignments γ_4^1 and γ_4^2, are the only moves on system defined for label d in all the alignments in the image of ϕ.

Thus, according to Theorem 5.3, S_5 is the result of an *optimal* alignment-based repair of S_1 w.r.t. $L := [\langle \mathtt{a}, \mathtt{b}, \mathtt{x}, \mathtt{e} \rangle, \langle \mathtt{a}, \mathtt{e} \rangle]$ as per f using R. One can use function $\psi := \{(\langle \mathtt{a}, \mathtt{b}, \mathtt{x}, \mathtt{e} \rangle, \gamma_5^1), (\langle \mathtt{a}, \mathtt{e} \rangle, \gamma_5^2)\}$ to justify this claim, where γ_5^1 and γ_5^2 are optimal alignments between traces in L and S_5 listed below.

$$\gamma_5^1 := \begin{array}{|c|c|c|c|c|} \hline \mathtt{a} & \gg & \mathtt{b} & \mathtt{x} & \mathtt{e} \\ \hline \mathtt{a} & \tau & \gg & \mathtt{x} & \mathtt{e} \\ \hline t_1 & t_7 & & t_6 & t_5 \\ \hline \end{array} \qquad \gamma_5^2 := \begin{array}{|c|c|c|} \hline \mathtt{a} & \gg & \mathtt{e} \\ \hline \mathtt{a} & \tau & \mathtt{e} \\ \hline t_1 & t_7 & t_5 \\ \hline \end{array}$$

One can easily verify that (i) $g'(\gamma_5^1) < g(\gamma_2^1)$ because $1 < 4$ and $g'(\gamma_5^2) < g(\gamma_2^2)$ because $0 < 3$, where g and g' are the cost functions on legal moves over S_1 and S_5 induced by f, respectively, and, thus, L fits S_5 at least as 'good' as (and in fact 'better' than) it fits S_1, and (ii) S_5 fulfills R as neither γ_5^1 nor γ_5^2 contain moves that are recommended for repair by R. Note that alignments γ_5^1 and γ_5^2 follow paths in S_5 that are *similar* to those followed by alignment γ_4^1 and γ_4^2 in S_1, respectively.

As demonstrated above, one can use adjusted cost functions to 'guide' optimal alignment-based repairs of net systems. We also use adjusted cost functions to define the concept of an optimal alignment-based repair recommendation.

DEFINITION 5.4 (OPTIMAL REPAIR RECOMMENDATION).
An *optimal repair recommendation* for a net system $S \in \mathbb{S}$ w.r.t. an event log $L \in \mathcal{B}(\mathcal{A}^*)$ as per a cost function on label moves f restricted by a repair constraint $C := (C_{ins}, C_{skp}, res)$ is a repair recommendation R such that $cost[R, C] \leq res$ and for every repair recommendation X, $cost[X, C] \leq res$, it holds that $cost[L, S, c_R] \leq cost[L, S, c_X]$, where c_R and c_X are the cost functions on legal moves over S induced by $f|_R$ and $f|_X$, respectively. ⌟

An optimal repair recommendation $R := (R_{ins}, R_{skp})$ for S w.r.t. L as per f restricted by C is said to be *minimal* iff there exists no other optimal repair recommendation $X := (X_{ins}, X_{skp})$ for S w.r.t. L as per f restricted by C such that $X_{ins} \subseteq R_{ins}$ and $X_{skp} \subseteq R_{skp}$ and $X \neq R$. By $\mathbb{R}[S, L, f, C]$ and $\mathbb{R}_{min}[S, L, f, C]$, we denote the set of all optimal and the set of all minimal optimal repair recommendations for S w.r.t. L as per f restricted by C, respectively.

Let f be a cost function on label moves. A feasible repair recommendation R under the given constraint C is optimal for a net system S if there exist no other feasible repair recommendation X under C for which the cost of optimal alignment between the given event log L and S as per the cost function on legal moves over S induced by $f|_X$ is 'cheaper' than the cost of optimal alignment between L and S as per the cost function on legal moves over S induced by $f|_R$. Note that the cost of optimal alignment between L and S as per the cost function on legal moves induced by $f|_R$ can be used to determine the cost of optimal alignment between L and the repaired version of S as per the cost function on legal moves induced by f.

LEMMA 5.5 (OPTIMAL ALIGNMENT-BASED REPAIR).
If a net system $S' \in \mathbb{S}$ is the result of a naïve repair of a net system $S \in \mathbb{S}$ based on the alignments in the image of a function ϕ using a repair recommendation R, where ϕ maps every trace υ in an event log $L \in \mathcal{B}(\mathcal{A}^*)$ to an alignment in $\mathbb{O}^{\upsilon}_{S,c}$, c is the cost function on legal moves over S induced by $f|_R$, and f is a cost function on label moves, then it holds that $cost[L, S, c] = cost[L, S', c']$, where c' is the cost function on legal moves over S' induced by f. ⌟

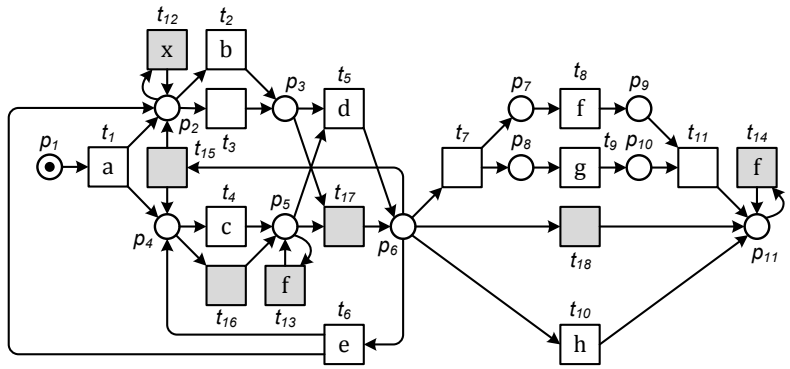


Fig. 6: A net system obtained as the result of a naïve repair of the net system in Fig. 2 using repair recommendation $R := (\{\mathtt{f},\mathtt{x}\},\{\mathtt{d},\mathtt{e},\mathtt{c},\mathtt{h}\})$.

The proof of Lemma 5.5 can be found in Appendix A.

Finally, if one performs an optimal alignment-based repair w.r.t. a given event log L using an optimal repair recommendation, then one obtains a repaired model that exhibits the best cost of optimal alignment with L among all the costs of optimal alignments between L and models that are obtained as alignment-based repairs using feasible repair recommendations.

COROLLARY 5.6 (OPTIMAL ALIGNMENT-BASED REPAIR).
If a net system $S' \in \mathbb{S}$ is the result of a naïve repair of a net system $S \in \mathbb{S}$ based on the alignments in the image of a function ϕ using a repair recommendation $R \in \mathbb{R}[S,L,f,C]$, where f is a cost function on label moves, ϕ maps every trace υ in an event log $L \in \mathcal{B}(\mathcal{A}^*)$ to an alignment in $\mathbb{O}^{\upsilon}_{S,c}$, c is the cost function on legal moves over S induced by $f|_R$, and $C := (C_{ins}, C_{skp}, res)$ is a repair constraint, then for every repair recommendation X such that $cost[X,C] \leq res$ it holds that $cost[L,S',c'] \leq cost[L,S'',c'']$, where $S'' \in \mathbb{S}$ is the result of a naïve repair of S based on ϕ using X, c' is the cost function on legal moves over S' induced by f, and c'' is the cost function on legal moves over S'' induced by f. ⌟

Corollary 5.6 follows immediately from Definition 5.4 and Lemma 5.5.

Let $C := (C_{ins}, C_{skp}, 6)$ be the standard repair constraint defined under the assumption that $\mathcal{A} = \{\mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{e},\mathtt{f},\mathtt{g},\mathtt{h},\mathtt{x}\}$. Then, $(\{\mathtt{f},\mathtt{x}\},\{\mathtt{d},\mathtt{e},\mathtt{c},\mathtt{h}\}) \in \mathbb{R}_{min}[S,L_3,f,C]$, where S is the net system in Fig. 2, L_3 is the event log from Section 2.3, and f is the standard cost function on label moves for events and activities $\mathcal{A}$.

Fig. 6 shows system S'' obtained as a result of a naïve repair of net system S in Fig. 2 using minimal optimal repair recommendation $R := (\{\mathtt{f},\mathtt{x}\},\{\mathtt{d},\mathtt{e},\mathtt{c},\mathtt{h}\})$; the 'injected' transitions are highlighted with grey background. This repair is performed based on optimal alignments between traces in L_3 and S as per the cost function on legal moves over S induced by f. Optimal alignments between traces in L_3 and S'' as per the cost function c on legal moves over S'' induced by f are listed below.

$\gamma_6^1 :=$

a	b	c	f	d	e	≫	f	≫	≫	≫
a	b	c	f	d	e	τ	f	τ	τ	τ
t_1	t_2	t_4	t_{13}	t_5	t_6	t_{16}	t_{13}	t_3	t_{17}	t_{18}

$\gamma_6^2 :=$

a	c	≫	d	≫	c	≫	≫	e	≫	≫	d	≫	g	f	≫
a	c	τ	d	τ	c	τ	τ	e	τ	τ	d	τ	g	f	τ
t_1	t_4	t_3	t_5	t_{15}	t_4	t_3	t_{17}	t_6	t_3	t_{16}	t_5	t_7	t_9	t_8	t_{11}

$\gamma_6^3 :=$

a	b	c	d	e	x	c	≫	≫	h	a
a	b	c	d	e	x	c	τ	τ	h	≫
t_1	t_2	t_4	t_5	t_6	t_{12}	t_4	t_3	t_{17}	t_{10}	

$\gamma_6^4 :=$

≫	c	≫	d	≫	≫	≫	d	≫	f	e	g	≫
a	c	τ	d	τ	τ	τ	d	τ	f	≫	g	τ
t_1	t_4	t_3	t_5	t_{15}	t_3	t_{16}	t_5	t_7	t_8		t_9	t_{11}

$$\gamma_6^5 := \begin{array}{|c|c|c|c|c|} \hline \texttt{a} & \texttt{b} & \gg & \gg & \gg \\ \hline \texttt{a} & \texttt{b} & \tau & \tau & \tau \\ t_1 & t_2 & t_{16} & t_{17} & t_{18} \\ \hline \end{array} \qquad \gamma_6^6 := \begin{array}{|c|c|c|c|c|c|c|c|c|c|} \hline \texttt{a} & \texttt{b} & \texttt{c} & \texttt{d} & \texttt{e} & \gg & \gg & \texttt{d} & \gg & \texttt{f} \\ \hline \texttt{a} & \texttt{b} & \texttt{c} & \texttt{d} & \texttt{e} & \tau & \tau & \texttt{d} & \tau & \texttt{f} \\ t_1 & t_2 & t_4 & t_5 & t_6 & t_3 & t_{16} & t_5 & t_{18} & t_{14} \\ \hline \end{array}$$

$$\gamma_6^7 := \begin{array}{|c|c|c|c|c|c|c|c|c|c|} \hline \texttt{a} & \texttt{b} & \texttt{c} & \texttt{d} & \texttt{e} & \texttt{b} & \texttt{c} & \texttt{d} & \gg & \texttt{g} \\ \hline \texttt{a} & \texttt{b} & \texttt{c} & \texttt{d} & \texttt{e} & \texttt{b} & \texttt{c} & \texttt{d} & \tau & \gg \\ t_1 & t_2 & t_4 & t_5 & t_6 & t_2 & t_4 & t_5 & t_{18} & \\ \hline \end{array}$$

One can verify that $c(\gamma_6^1) = 0$, $c(\gamma_6^2) = 0$, $c(\gamma_6^3) = 1$, $c(\gamma_6^4) = 2$, $c(\gamma_6^5) = 0$, $c(\gamma_6^6) = 0$, and $c(\gamma_6^7) = 1$, and, thus, it holds that $cost[L_3, S'', c] = 10 \times 0 + 9 \times 0 + 9 \times 1 + 7 \times 2 + 6 \times 0 + 2 \times 0 + 2 \times 1 = 25$. According to Corollary 5.6, there exists no feasible repair recommendation X under C such that for net system $\bar{S}$ obtained as the result of a naïve repair of S based on the same alignments that are employed to repair S'' using X it holds that L_3 fits $\bar{S}$ better than it fits S''. For example, as demonstrated in Section 4, the cost of optimal alignment between L_3 and net system S' in Fig. 4 that is repaired using repair recommendation $(\{\texttt{e},\texttt{f},\texttt{x}\},\{\texttt{c},\texttt{f},\texttt{g}\})$, which is a feasible repair recommendation under C, equals 47. Clearly, it holds that $(\{\texttt{e},\texttt{f},\texttt{x}\},\{\texttt{c},\texttt{f},\texttt{g}\}) \notin \mathbb{R}[S, L_3, f, C]$.

In the next section, we discuss issues concerning discovery and approximation of minimal optimal repair recommendations for net systems.

6. SEARCHING MODEL REPAIR RECOMMENDATIONS

This section discusses techniques that given a net system, an event log, a cost function on label moves and a repair constraint, discover (approximate) solutions to the optimal model repair recommendation problem stated in the previous section. The section starts with definitions of some algebraic operations and relations on repair recommendations, cf. Section 6.1. These introduced concepts are used to simplify subsequent presentations of the proposed search techniques. Section 6.2 presents the exhaustive search method for solving the optimal model repair recommendation problem, whereas Sections 6.3–6.5 devise techniques capable of discovering viable minimal optimal repair recommendations using efficient approximation schemes. Section 6.6 summarizes this section.

6.1. Algebra of Repair Recommendations

Let $R_1 := (R_{ins}^1, R_{skp}^1)$ and $R_2 := (R_{ins}^2, R_{skp}^2)$ be two repair recommendations. If every element in R_{ins}^1 is also an element in R_{ins}^2 and every element in R_{skp}^1 is also an element in R_{skp}^2, then we say that R_1 *is contained in* R_2, written $R_1 \sqsubseteq R_2$. Equivalently, one can write $R_2 \sqsupseteq R_1$; must be read as R_2 *contains* R_1. If R_1 is contained in R_2 and is not equal to R_2, then R_1 is *properly* contained in R_2, denoted by $R_1 \sqsubset R_2$. Equivalently, one can write $R_2 \sqsupset R_1$ to say that R_2 *properly* contains R_1. The relations between repair recommendations established by '$\sqsubseteq$' and '$\sqsubset$' are called *containment* and *proper containment*, respectively. The containment relation over a set of repair recommendations is a partial order, i.e., '$\sqsubseteq$' is reflexive, antisymmetric, and transitive. Let $\mathbb{R} := \mathcal{P}(\mathcal{A}) \times \mathcal{P}(\mathcal{A})$ be the set of all possible repair recommendations defined over $\mathcal{A}$. Then, $(\mathbb{R}, \sqsubseteq)$ is a lattice. Indeed, for every two repair recommendations $R_1, R_2 \in \mathbb{R}$, such that $R_1 := (R_{ins}^1, R_{skp}^1)$ and $R_2 := (R_{ins}^2, R_{skp}^2)$, it holds that $R_1 \sqcap R_2 := (R_{ins}^1 \cap R_{ins}^2, R_{skp}^1 \cap R_{skp}^2)$ and $R_1 \sqcup R_2 := (R_{ins}^1 \cup R_{ins}^2, R_{skp}^1 \cup R_{skp}^2)$ are the greatest lower bound and the least upper bound of R_1 and R_2, respectively.

Let $\mathcal{X} \subseteq \mathbb{R}$, by $\mathcal{X}_{min}$ and $\mathcal{X}_{max}$ we denote the set of all minimal and the set of all maximal (as per the containment relation) elements in $\mathcal{X}$, respectively.

6.2. Brute-force Search

Brute-force search, or an exhaustive search, is a problem solving technique that consists of systematically enumerating and checking all possible candidate solutions with the purpose of identifying those solutions that satisfy the problem statement.

ALGORITHM 1: *BruteForceRecommendationSearch*(S,L,f,C)

Input: A net system S, an event log L, a cost function on label moves f, and a repair constraint $C := (C_{ins}, C_{skp}, res)$.

Output: An ordered pair composed of the set of all minimal optimal repair recommendations for S w.r.t. L as per f restricted by C, i.e., $\mathbb{R}_{min}[S,L,f,C]$, and the cost of optimal alignment *optCost* between L and S such that $\forall R \in \mathbb{R}_{min}[S,L,f,C] : cost[L,S,c_R] = optCost$, where c_R is the cost function on legal moves over S induced by $f|_R$.

```
1 optCost := +∞; R := ∅;
2 foreach R ∈ {X ∈ P(labels(L)) × P(labels(S)) | cost[X,C] ≤ res} do
3     c_R is the cost function on legal moves over S induced by f|_R;
4     cost := cost[L,S,c_R];
5     if cost < optCost then
6         optCost := cost; R := ∅;
7     if cost = optCost then R := R ∪ {R};
8 return (R_min, optCost);
```

Let $S \in \mathbb{S}$, $S := (N, M_{ini}, M_{fin})$, $N := (P,T,F,\lambda)$, be a net system. By $labels(S)$, we denote the set of all labels assigned to observable transitions in T, i.e., $labels(S) := img(\lambda) \setminus \{\tau\}$.[4] Similarly, given an event log $L \in \mathcal{B}(\mathcal{A}^*)$, by $labels(L)$ we denote the set of all labels that are elements of traces in L, i.e., $labels(L) := \bigcup_{\upsilon \in L} Set(\upsilon)$.

It is easy to propose the brute-force approach for discovering $\mathbb{R}_{min}[S,L,f,C]$, i.e., the set of all minimal optimal repair recommendations for a net system $S \in \mathbb{S}$ w.r.t. an event log $L \in \mathcal{B}(\mathcal{A}^*)$ as per a cost function on label moves f restricted by a repair constraint $C := (C_{ins}, C_{skp}, res)$. One must simply check every feasible repair recommendation $R \in \mathcal{P}(labels(L)) \times \mathcal{P}(labels(S))$ for which it holds that $cost[R,C]$ is less than or equal to res and select those for which $cost[L,S,c_R]$ is the lowest, where c_R is the cost function on legal moves over S induced by $f|_R$, and which do not properly contain other feasible repair recommendations that lead to the lowest cost of optimal alignment between L and S. This intuition is formalized below.

$$\mathbb{F}[S,L,C] := \{X \in \mathcal{P}(labels(L)) \times \mathcal{P}(labels(S)) \mid cost[X,C] \leq res\}$$
$$\mathbb{R}_{min}[S,L,f,C] := \{Y \in \mathbb{F}[S,L,C] \mid \forall Z \in \mathbb{F}[S,L,C] : cost[L,S,c_Y] \leq cost[L,S,c_Z]\}_{min}$$

Here, c_Y and c_Z are the cost functions on legal moves over S induced by $f|_Y$ and $f|_Z$, respectively. Note that the feasible repair recommendations ($\mathbb{F}[S,L,C]$) are selected from repair recommendations in $\mathcal{P}(labels(L)) \times \mathcal{P}(labels(S))$. This is because it only makes sense to recommend those labels for insertion that are actually observed in the event log and to propose those labels for skipping that are actually used in the net system. Finally, the use of the adjusted cost functions in the above formulae is justified by Theorem 5.3 and Lemma 5.5, which imply that the costs of optimal alignments are, indeed, the costs of optimal alignments between net systems that are obtained via some optimal alignment-based repairs of S and event log L.

Algorithm 1 suggests an implementation (in pseudocode) of the brute-force technique for discovering the set of all minimal optimal repair recommendations.

The main logic of Algorithm 1 is encoded in the `foreach` loop of lines 2–7. This loop checks all the feasible repair recommendations (one per each iteration) to select those that 'lead' to the lowest cost of optimal alignment. Once all the optimal recommendations have been discovered, the minimal ones are returned at line 8. The algorithm also returns the value of *optCost* which reflects the cost of optimal alignment between the given event log and some net system obtained as a result of an optimal alignment-based repair of the given net system, refer to Theorem 5.3 and Lemma 5.5.

[4] By $img(f)$, we denote the image of function $f : X \to Y$ on the entire domain X, i.e., $img(f) := \{y \in Y \mid \exists x \in X : f(x) = y\}$.

ALGORITHM 2: *BruteForceRecommendationSearchWithOptimization*(S, L, f, C)

Input: A net system S, an event log L, a cost function on label moves f, and a repair constraint $C := (C_{ins}, C_{skp}, res)$.

Output: An ordered pair composed of the set of all minimal optimal repair recommendations for S w.r.t. L as per f restricted by C, i.e., $\mathbb{R}_{min}[S, L, f, C]$, and the cost of optimal alignment *optCost* between L and S such that $\forall R \in \mathbb{R}_{min}[S, L, f, C] : cost[L, S, c_R] = optCost$, where c_R is the cost function on legal moves over S induced by $f|_R$.

```
1 optCost := +∞; R := ∅;
2 foreach R ∈ {X ∈ P(labels(L)) × P(labels(S)) | cost[X, C] ≤ res}_max do
3     c_R is the cost function on legal moves over S induced by f|_R;
4     cost := cost[L, S, c_R];
5     if cost < optCost then
6         optCost := cost; R := ∅;
7     if cost = optCost then R := R ∪ {R};
8 return (MinimizeRecommendations(R, S, L, f, optCost), optCost);
```

For the optimal model repair recommendation problem, however, the number of feasible repair recommendations tends to grow very quickly as the size of the problem increases. If one uses a standard repair constraint $C := (C_{ins}, C_{skp}, res)$ as input to Algorithm 1, then the number of feasible repair recommendations to be checked in the iterations of the `foreach` loop of lines 2–7 is equal to[5]:

$$\sum_{i=0}^{res} \sum_{j=0}^{i} \binom{|labels(L)|}{j} \times \binom{|labels(S)|}{i-j}.$$

For example, for the input of the net system S in Fig. 2, event log L_3 from Section 2.3, and the standard repair constraint $C := (C_{ins}, C_{skp}, 6)$, Algorithm 1 performs 21 778 iterations of the `foreach` loop of lines 2–7; note that this number is independent of the cost function on label moves used in the input. Considering that each iteration of the loop performs one computation of the optimal alignment between each trace in the given event log and the net system, cf. line 4, the overall running time of Algorithm 1 is often intractable. For the above mentioned input and the standard cost function f on label moves, the algorithm took 57.216 seconds to discover all the five minimal optimal repair recommendations in $\mathbb{R}_{min}[S, L_3, f, C]$; these are recommendations $(\{\mathtt{f}\}, \{\mathtt{f}, \mathtt{d}, \mathtt{e}, \mathtt{c}, \mathtt{h}\})$, $(\{\mathtt{f}, \mathtt{a}\}, \{\mathtt{d}, \mathtt{e}, \mathtt{c}, \mathtt{h}\})$, $(\{\mathtt{f}\}, \{\mathtt{f}, \mathtt{g}, \mathtt{d}, \mathtt{e}, \mathtt{c}\})$, $(\{\mathtt{f}, \mathtt{x}\}, \{\mathtt{d}, \mathtt{e}, \mathtt{c}, \mathtt{h}\})$, and $(\{\mathtt{f}, \mathtt{g}\}, \{\mathtt{d}, \mathtt{e}, \mathtt{c}, \mathtt{h}\})$. Recall that the net system obtained after a naïve repair of S using repair recommendation $(\{\mathtt{f}, \mathtt{x}\}, \{\mathtt{d}, \mathtt{e}, \mathtt{c}, \mathtt{h}\})$ is shown in Fig. 6. Recall also that the cost of optimal alignment between event log L_3 and this repaired net system is equal to 25. Note that one will obtain the same cost of optimal alignment between L_3 and every net system obtained as the result of a naïve repair of S using any of the other four repair recommendations in $\mathbb{R}_{min}[S, L_3, f, C]$.

Interestingly, one can introduce an optimization to the logic of the brute-force approach of Algorithm 1. This optimization is due to the next observation.

PROPOSITION 6.1 (MONOTONICITY BETWEEN REPAIR RECOMMENDATIONS AND COSTS). *If R_1 and R_2 are two repair recommendations such that $R_1 \sqsubseteq R_2$, then it holds that $cost[\upsilon, S, c_{R_2}] \leq cost[\upsilon, S, c_{R_1}]$, where $\upsilon \in \mathcal{A}^*$ is a trace, $S \in \mathbb{S}$ is a net system, c_{R_1} and c_{R_2} are the cost functions on legal moves over S induced by $f|_{R_1}$ and $f|_{R_2}$, respectively, and f is a cost function on label moves.* ⌟

The proof of Proposition 6.1 is immediate as, clearly, given an alignment, its cost per c_{R_2} cannot be larger than that one per c_{R_1}, as c_{R_2} can be obtained from c_{R_1} by setting costs of some moves to zeros.

Every minimal optimal repair recommendation is (properly) contained in some optimal one. Thus, in order to discover all minimal optimal repair recommendations, one can first discover all maximal

[5] $\binom{n}{k}$ is the binomial coefficient indexed by n and k.

ALGORITHM 3: *MinimizeRecommendations*$(\mathcal{R}, S, L, f, cost)$

Input: A set of repair recommendations $\mathcal{R}$, a net system S, an event log L, a cost function on label moves f, and $cost \in \mathbb{N}_0$ such that for every $R \in \mathcal{R}$ it holds that $cost[L, S, c_R] = cost$, where c_R is the cost function on legal moves over S induced by $f|_R$.

Output: The set $\mathcal{X}$ of all repair recommendations such that for every $X \in \mathcal{X}$ it holds that $cost[L, S, c_X] = cost$, there exists a repair recommendation $Y \in \mathcal{R}$ such that X is contained in Y, i.e., $X \sqsubseteq Y$, and there exists no repair recommendation Z such that $Z \sqsubset X$ and $cost[L, S, c_Z] = cost$, where c_X and c_Z are the cost functions on legal moves over S induced by $f|_X$ and $f|_Z$, respectively.

1 $\mathcal{X} := \emptyset$; $visited := \emptyset$; $toVisit := \mathcal{R}$;
2 **while** $toVisit \neq \emptyset$ **do**
3 $\quad toVisit := toVisit \setminus \{(R_{ins}, R_{skp})\}$, where $(R_{ins}, R_{skp}) \in toVisit$;
4 $\quad visited := visited \cup \{(R_{ins}, R_{skp})\}$;
5 $\quad isMinimal :=$ **true**;
6 $\quad$ **foreach** $(X, Y) \in (\{\{x\} \mid x \in R_{ins}\} \times \{\emptyset\}) \cup (\{\emptyset\} \times \{\{y\} \mid y \in R_{skp}\})$ **do**
7 $\quad\quad R := (R_{ins} \setminus X, R_{skp} \setminus Y)$;
8 $\quad\quad c_R$ is the cost function on legal moves over S induced by $f|_R$;
9 $\quad\quad$ **if** $cost[L, S, c_R] = cost$ **then**
10 $\quad\quad\quad$ **if** $R \notin visited$ **then** $toVisit := toVisit \cup \{R\}$;
11 $\quad\quad\quad isMinimal :=$ **false**;
12 $\quad$ **if** $isMinimal$ **then** $\mathcal{X} := \mathcal{X} \cup \{(R_{ins}, R_{skp})\}$;
13 **return** $\mathcal{X}$;

ones and then use them to construct all the minimal repair recommendations. This intuition is captured in Algorithm 2.

The structure of Algorithm 2 resembles that of Algorithm 1. The only two differences between the algorithms are (i) in the sets of repair recommendations used to guide execution of the `foreach` loop of lines 2–7, and (ii) in the way the minimal optimal repair recommendations get constructed at lines 8 of both algorithms. Instead of checking every feasible repair recommendation, Algorithm 2 starts by selecting optimal repair recommendations from the set of maximal feasible recommendations. The discovered optimal repair recommendations are collected in set $\mathcal{R}$. Then, the minimal recommendations are constructed from those in $\mathcal{R}$ by iteratively removing one label and checking whether the resulting repair recommendations lead to the same cost of optimal alignment (the exact procedure for accomplishing this step is captured in Algorithm 3).

For the input of net system S in Fig. 2, event log L_3 from Section 2.3, and the standard repair constraint $C := (C_{ins}, C_{skp}, 6)$, Algorithm 2 performs 12 406 iterations of the `foreach` loop of lines 2–7. For the input of S, L_3, C, and the standard cost function f on label moves, Algorithm 2 took 32.594 seconds to discover all the five minimal optimal repair recommendations.

6.3. Knapsack Search

Every set of optimal alignments Γ between traces of an event log $L \in \mathcal{B}(\mathcal{A}^*)$ and a net system $S \in \mathbb{S}$ that can be used to compute the cost of optimal alignment between L and S, refer to Definitions 3.5 and 3.7, contains information on all the individual cost contributions of legal moves over S. The contribution of a legal move to the cost of optimal alignment can be computed as the sum of all the occurrences of the move in the optimal alignments in Γ multiplied by the cost of this move. Given a repair constraint $C := (C_{ins}, C_{skp}, res)$, one can repair S to cater for those legal moves in the alignments in Γ that impact the cost of optimal alignment between L and S the most while keeping the repair cost (as per cost functions C_{ins} and C_{skp}) within the amount of available repair resources res.

Interestingly, the above intuition is in line with the intuition of the Knapsack problem [Dantzig 1957]—a classical problem in combinatorial optimization. Given a set of items, each with a value and a weight, the Knapsack problem consists in determining the number of each item to include in a

ALGORITHM 4: *KnapsackApproximateRecommendationSearch*$(S, L, f, C, singleton)$

Input: A net system S, an event log L, a cost function on label moves f, a repair constraint $C := (C_{ins}, C_{skp}, res)$, and a Boolean parameter *singleton*.

Output: An ordered pair $(\mathcal{R}, optCost)$, where $\mathcal{R}$ is a set of *approximate* minimal optimal repair recommendations for S w.r.t. L as per f restricted by C, and *optCost* is the cost of optimal alignment between L and S such that $\forall R \in \mathcal{R} : cost[L, S, c_R] = optCost$, where c_R is the cost function on legal moves over S induced by $f|_R$.

c is the cost function on legal moves over S induced by f;
$X := \{(\upsilon, \gamma) \mid \upsilon \in L \wedge \gamma \in \mathbb{O}^{\upsilon}_{S,c}\}$;
$Y \subseteq X$ such that $\bigcup_{y \in Y} \{\pi_1(y)\} = Set(L)$ and $\forall\, y_1 \in Y\ \forall y_2 \in Y : (\pi_1(y_1) = \pi_1(y_2)) \Rightarrow y_1 = y_2$;
$optCost := \sum_{(\upsilon,\gamma) \in Y} (L(\upsilon) \times c(\gamma))$;
$weights := \{((\{x\}, \varnothing), C_{ins}(x)) \mid (x, \gg) \in dom(f)\} \cup \{((\varnothing, \{x\}), C_{skp}(x)) \mid (\gg, x) \in dom(f)\}$;
if $\forall\, w \in img(weights) : w > res$ **then return** $(\{(\varnothing, \varnothing)\}, optCost)$;
$values := \{((\{x\}, \varnothing), f((x, \gg)) \times count((x, \gg), Y, L, S)) \mid (x, \gg) \in dom(f)\} \cup$
$\{((\varnothing, \{x\}), f((\gg, x)) \times count((\gg, x), Y, L, S)) \mid (\gg, x) \in dom(f)\}$;
$\mathcal{K} := \varnothing$; $\mathcal{R} := \varnothing$;
if *singleton* **then** $\mathcal{K} :=$ *Knapsack01OneSolution*$(weights, values, res)$;
else $\mathcal{K} :=$ *Knapsack01AllSolutions*$(weights, values, res)$;
foreach $K \in \mathcal{K}$ **do**
    $R := \bigcup_{k \in K} k$;
    c_R is the cost function on legal moves over S induced by $f|_R$;
    $cost := cost[L, S, c_R]$;
    if $cost < optCost$ **then**
        $optCost := cost$; $\mathcal{R} := \varnothing$;
    if $cost = optCost$ **then** $\mathcal{R} := \mathcal{R} \cup \{R\}$;
return $(\mathcal{R}, optCost)$;

collection so that the total weight of items in the collection is less than or equal to a given capacity and the total value of the collection is as large as possible. The 0/1 Knapsack problem [Fayard and Plateau 1975; Martello et al. 2000] is a variant of the above stated problem. It requires solutions to be sets, i.e., no item can be included more than once in a solution to a 0/1 Knapsack problem.

A solution to the 0/1 Knapsack problem for the input of a set of repair recommendations, each with a value, i.e., its impact on the cost of optimal alignment between L and a repaired version of S obtained using this recommendation, and a weight, i.e., its cost as per C_{ins} and C_{skp}, 'recommends' repairs that address the most 'costly' differences between the traces and executions used to build Γ.

Let $S := (N, M_{ini}, M_{fin})$, $N := (P, T, F, \lambda)$, be a net system, let f be a cost function on label moves, let L be an event log, and let h be a function that maps every trace $\upsilon \in L$ to an alignment in $\mathbb{A}^{\upsilon}_{S}$. Then, by $count(m, h, L, S)$, $m \in dom(f)$, we denote the number of times that m gets 'reflected' in alignments (from the image of h) between traces in L and executions of S, i.e., $count(m, h, L, S) := \sum_{\upsilon \in L}(L(\upsilon) \times |h(\upsilon)|_{moves[m,S]}|)$, where $moves[m, S] := \{(x, y) \in \mathbb{M}_S \mid (x, y) = m \vee (y \in T \wedge x = \gg \wedge m = (x, \lambda(y)))\}$. Then, Algorithm 4 captures the above discussed reasoning in pseudocode.

Given a net system S, an event log L, a cost function on label moves f, a repair constraint $C := (C_{ins}, C_{skp}, res)$, Algorithm 4 discovers a set of *approximations* of the minimal optimal repair recommendations $\mathcal{R}$ for S w.r.t. L as per f restricted by C and the cost of optimal alignment *optCost* between a net system S' and L as per the cost function on legal moves over S' induced by f, where S' is some net system that can be obtained as the result of an optimal alignment-based repair of S w.r.t. L as per f using any repair recommendation in $\mathcal{R}$, cf. Definition 4.3. If the input parameter *singleton* is set to true, the algorithm discovers one repair recommendation, i.e., the set of discovered repair recommendations has exactly one element; otherwise it may contain multiple recommendations.

For every distinct trace υ in the input event log L, set Y (computed at line 3 of Algorithm 4) contains one ordered pair composed of υ and some optimal alignment between υ and S as per the cost function on legal moves c over S induced by f. The value of *optCost* (computed at line 4) is the cost of optimal alignment between L and S as per c, refer to Definition 3.7. Functions *weights* (line 5) and *values* (line 7), map repair recommendations to their weights and values, respectively. These functions are computed using the principles discussed above for the elementary repair recommendations, i.e., repair recommendations composed of a single label. If for every repair recommendation it holds that its weight exceeds the amount of available repair resources, i.e., the weight of every item exceeds the knapsack's capacity, then the solution to the problem is the empty knapsack,cf. line 6 of the algorithm.

If non-empty solutions to the 0/1 Knapsack problem exist, they are discovered at lines 10 and 11 of Algorithm 4. If the input Boolean parameter *singleton* is set to true, then a single solution is sought via a call to the *Knapsack01OneSolution* procedure at line 10. Otherwise, all solutions are computed by issuing a call to the *Knapsack01AllSolutions* procedure at line 11. The *Knapsack01OneSolution* procedure can be implemented using dynamic programming [Andonov et al. 2000] or branch and bound [Vance 1993] approaches, while the *Knapsack01AllSolutions* procedure can be carried out using techniques for computing k-best solutions of knapsack problems [Leão et al. 2014].

After executing lines 10 and 11 of Algorithm 4, set $\mathcal{K}$ contains all the discovered solutions to the knapsack problem. These solutions are then transformed into repair recommendations using the `foreach` loop of lines 12–18. For every discovered solution K, a repair recommendation R is constructed at line 13 as the union of elementary repair recommendations contained in the solution. The algorithm then computes the cost of optimal alignment between L and S as per the cost function on legal moves over S induced by $f|_R$, refer to line 15. Recall, that this cost represents the cost of optimal alignment between L and a net system obtained as a result of an optimal alignment-based repair of S, cf. Theorem 5.3 and Lemma 5.5. Repair recommendations that lead to the minimal cost are then collected in $\mathcal{R}$ at lines 16–18.

Note that Algorithm 4 is nondeterministic. This is because of multiple possible constructions of set Y at line 3 as, indeed, there can exist multiple optimal alignments between a trace and a net system.

Let, again, $C \coloneqq (C_{ins}, C_{skp}, 6)$ be the standard repair constraint defined under the assumption that $\mathcal{A} = \{\mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{e},\mathtt{f},\mathtt{g},\mathtt{h},\mathtt{x}\}$. Then, it holds that $(\{(\{\mathtt{f},\mathtt{e},\mathtt{x}\},\{\mathtt{g},\mathtt{d},\mathtt{c}\})\},40) = \mathit{KnapsackApproximateRecommendationSearch}(S, L_3, f, C, \mathit{true})$, where S is the net system in Fig. 2, L_3 is the event log from Section 2.3, and f is the standard cost function on label moves for events and activities $\mathcal{A}$. On this input, set Y computed at line 3 of Algorithm 4 can be composed of seven pairs where the first elements of the pairs are distinct traces in L_3 and each second element is the alignment between the trace at the first position in the pair and S shown on Page 15. Then, the 'weights' of repairs computed at line 5 of the algorithm are given by the set $\mathit{weights} \coloneqq \{((\{\mathtt{a}\},\varnothing),1),((\{\mathtt{b}\},\varnothing),1),((\{\mathtt{c}\},\varnothing),1),((\{\mathtt{d}\},\varnothing),1),((\{\mathtt{e}\},\varnothing),1),((\{\mathtt{f}\},\varnothing),1),((\{\mathtt{g}\},\varnothing),1),((\{\mathtt{h}\},\varnothing),1),((\{\mathtt{x}\},\varnothing),1),((\varnothing,\{\mathtt{a}\}),1),((\varnothing,\{\mathtt{b}\}),1),((\varnothing,\{\mathtt{c}\}),1),((\varnothing,\{\mathtt{d}\}),1),((\varnothing,\{\mathtt{e}\}),1),((\varnothing,\{\mathtt{f}\}),1),((\varnothing,\{\mathtt{g}\}),1),((\varnothing,\{\mathtt{h}\}),1),((\varnothing,\{\mathtt{x}\}),1)\}$, i.e., all the repairs have 'weight' of one unit because of the given standard repair constraint. The 'values' of the repairs computed at line 7 of the algorithm are given by the set $\mathit{values} \coloneqq \{((\{\mathtt{a}\},\varnothing),9),((\{\mathtt{b}\},\varnothing),0),((\{\mathtt{c}\},\varnothing),9),((\{\mathtt{d}\},\varnothing),7),((\{\mathtt{e}\},\varnothing),17),((\{\mathtt{f}\},\varnothing),10),((\{\mathtt{g}\},\varnothing),0),((\{\mathtt{h}\},\varnothing),0),((\{\mathtt{x}\},\varnothing),9),((\varnothing,\{\mathtt{a}\}),7),((\varnothing,\{\mathtt{b}\}),0),((\varnothing,\{\mathtt{c}\}),17),((\varnothing,\{\mathtt{d}\}),15),((\varnothing,\{\mathtt{e}\}),0),((\varnothing,\{\mathtt{f}\}),2),((\varnothing,\{\mathtt{g}\}),12),((\varnothing,\{\mathtt{h}\}),6),((\varnothing,\{\mathtt{x}\}),0)\}$. For example, the 'value' of repair recommendation $(\varnothing,\{\mathtt{c}\})$ is seventeen, because move $(\gg,\mathtt{c})$ occurs at positions eight, three, and seven in alignments γ_0^2, γ_0^5, and γ_0^6 on Page 15, respectively. Moreover, the corresponding traces $\langle\mathtt{a},\mathtt{c},\mathtt{d},\mathtt{c},\mathtt{e},\mathtt{d},\mathtt{g},\mathtt{f}\rangle$, $\langle\mathtt{a},\mathtt{b}\rangle$, and $\langle\mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{e},\mathtt{d},\mathtt{f}\rangle$ occur nine, six and two times in L_3, respectively, and, thus, it holds that $\mathit{count}((\gg,\mathtt{c}),Y,L_3,S) = 17$. Hence, one possible solution to the resulting 0/1 Knapsack problem at line 10 of the algorithm is $\mathcal{K} \coloneqq \{\{((\{\mathtt{e}\},\varnothing),17),((\{\mathtt{f}\},\varnothing),10),((\{\mathtt{x}\},\varnothing),9),((\varnothing,\{\mathtt{c}\}),17),((\varnothing,\{\mathtt{d}\}),15),((\varnothing,\{\mathtt{g}\}),12)\}\}$ which gives the highest possible value of 80 for a packed rucksack of capacity six. Note that $\{((\{\mathtt{e}\},\varnothing),17),((\{\mathtt{f}\},\varnothing),10),((\{\mathtt{a}\},\varnothing),9),((\varnothing,\{\mathtt{c}\}),17),((\varnothing,\{\mathtt{d}\}),15),((\varnothing,\{\mathtt{g}\}),12)\}$ and $\{((\{\mathtt{e}\},\varnothing),17),((\{\mathtt{f}\},\varnothing),10),((\{\mathtt{c}\},\varnothing),9),((\varnothing,\{\mathtt{c}\}),17),((\varnothing,\{\mathtt{d}\}),$

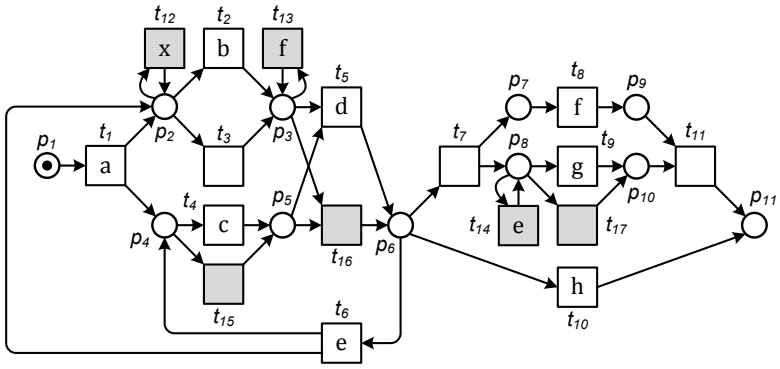


Fig. 7: A net system obtained as the result of a naïve repair of the net system in Fig. 2 using repair recommendation $R := (\{\mathtt{f},\mathtt{e},\mathtt{x}\},\{\mathtt{g},\mathtt{d},\mathtt{c}\})$ discovered using Algorithm 4.

$15),((\varnothing,\{\mathtt{g}\}),12)\}$ are two other solutions to the 0/1 Knapsack problem at hand, which can be discovered at line 11 of Algorithm 4 provided that the input *singleton* parameter is set to false. Finally, at line 13 the algorithm constructs repair recommendation $R := (\{\mathtt{f},\mathtt{e},\mathtt{x}\},\{\mathtt{g},\mathtt{d},\mathtt{c}\})$ by performing the union, refer to Section 6.1, of individual recommendations in the only set contained in $\mathcal{K}$, at line 15 it computes the cost of optimal alignment between L_3 and the repaired version of S, and at line 18 the algorithm saves the result of executing the algorithm, which is returned at line 19. Note that though the cost of optimal alignment between L_3 and the repaired version of S is $40 = 120 - 80$, where 120 is the cost of optimal alignment between L_3 and S and 80 is the 'value' of the discovered knapsack, in general the cost of optimal alignment between the given event log and the repaired version of the given net system can be lower than the difference of the two numbers due to the absence of control over the way the optimal alignments are constructed.

Fig. 7 shows net system S''' obtained as a result of a naïve repair of S using repair recommendation $R := (\{\mathtt{f},\mathtt{e},\mathtt{x}\},\{\mathtt{g},\mathtt{d},\mathtt{c}\})$; the 'injected' transitions are highlighted with grey background. This repair is performed based on optimal alignments between traces in L_3 and S as per the cost function on legal moves over S induced by f. Optimal alignments between traces in L_3 and S''' as per the cost function c on legal moves over S''' induced by f are listed below.

$$\gamma_7^1 := \begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{f}&\mathtt{d}&\gg&\mathtt{e}&\mathtt{f}&\gg&\gg\\ \hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{f}&\mathtt{d}&\tau&\mathtt{e}&\mathtt{f}&\tau&\tau\\ \hline t_1&t_2&t_4&t_{13}&t_5&t_7&t_{14}&t_8&t_{17}&t_{11}\\ \hline\end{array} \qquad \gamma_7^2 := \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{c}&\gg&\mathtt{d}&\mathtt{c}&\mathtt{e}&\gg&\gg&\mathtt{d}&\gg&\mathtt{g}&\mathtt{f}&\gg\\ \hline \mathtt{a}&\mathtt{c}&\tau&\mathtt{d}&\gg&\mathtt{e}&\tau&\tau&\mathtt{d}&\tau&\mathtt{g}&\mathtt{f}&\tau\\ \hline t_1&t_4&t_3&t_5&&t_6&t_3&t_{15}&t_5&t_7&t_9&t_8&t_{11}\\ \hline\end{array}$$

$$\gamma_7^3 := \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\mathtt{x}&\mathtt{c}&\gg&\gg&\mathtt{h}&\mathtt{a}\\ \hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\mathtt{x}&\mathtt{c}&\tau&\tau&\mathtt{h}&\gg\\ \hline t_1&t_2&t_4&t_5&t_6&t_{12}&t_4&t_3&t_{16}&t_{10}&\\ \hline\end{array} \qquad \gamma_7^4 := \begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline \gg&\mathtt{c}&\gg&\mathtt{d}&\gg&\mathtt{d}&\mathtt{f}&\mathtt{e}&\mathtt{g}&\gg\\ \hline \mathtt{a}&\mathtt{c}&\tau&\mathtt{d}&\tau&\gg&\mathtt{f}&\mathtt{e}&\mathtt{g}&\tau\\ \hline t_1&t_4&t_3&t_5&t_7&&t_8&t_{14}&t_9&t_{11}\\ \hline\end{array}$$

$$\gamma_7^5 := \begin{array}{|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{b}&\gg&\gg&\gg\\ \hline \mathtt{a}&\mathtt{b}&\tau&\tau&\mathtt{h}\\ \hline t_1&t_2&t_{15}&t_{16}&t_{10}\\ \hline\end{array} \qquad \gamma_7^6 := \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\gg&\gg&\mathtt{d}&\gg&\mathtt{f}&\gg&\gg\\ \hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\tau&\tau&\mathtt{d}&\tau&\mathtt{f}&\tau&\tau\\ \hline t_1&t_2&t_4&t_5&t_6&t_3&t_{15}&t_5&t_7&t_8&t_{17}&t_{11}\\ \hline\end{array}$$

$$\gamma_7^7 := \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\gg&\mathtt{g}&\gg&\gg\\ \hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\tau&\mathtt{g}&\mathtt{f}&\tau\\ \hline t_1&t_2&t_4&t_5&t_6&t_2&t_4&t_5&t_7&t_9&t_8&t_{11}\\ \hline\end{array}$$

Using these alignments, one can easily verify that $c(\gamma_7^1) = 0$, $c(\gamma_7^2) = 1$, $c(\gamma_7^3) = 1$, $c(\gamma_7^4) = 2$, $c(\gamma_7^5) = 1$, $c(\gamma_7^6) = 0$, and $c(\gamma_7^7) = 1$, and, thus, it holds that $cost[L_3, S''', c] = 10\times0 + 9\times1 + 9\times1 + 7\times2 + 6\times1 + 2\times0 + 2\times1 = 40$.

Algorithm 4 performs $1+n$ computations of optimal alignments between traces in the input event log and net system (as per different cost functions on legal moves), where n is the size of set $\mathcal{K}$ once the algorithm reaches the execution of line 12. Therefore, every call to $KnapsackApproximateRecommendationSearch(S, L_3, f, C, true)$ that reaches the execution of line 19 performs at least two costly alignment computations. Note that if the returned value of *optCost* is of no interest, the algorithm can be easily adapted to rely on a single computation.

6.4. Goldratt Search

The theory of constraints, introduced by Eliyahu M. Goldratt in his book entitled The Goal [Goldratt et al. 2012], studies approaches that help systems with managing the achievement of their goals. Goldratt argues that every system has at least one constraint, where a constraint is anything that prevents the system from achieving (more of) its goals. Given a system and methods for measuring accomplishments of its goal(s), the theory of constraints proposes the following procedure for improving performance of the system [Goldratt et al. 2012]:

(1) Identify the system's constraint(s).
(2) Decide how to exploit the system's constraint(s).
(3) Subordinate everything else to the above decision.
(4) Elevate the system's constraint(s).
(5) If in the previous steps a constraint has been broken, go back to step 1, but do not allow inertia to cause a system's constraint.

Given a system S, an event log L, a cost function on label moves f, and a repair constraint C, our goal is to repair S to obtain a net system S' such that L fits S' as much as possible. One can use the cost of optimal alignment between L and S' as per the cost function on legal moves over S' induced by f as a measure of fitness. We propose to adapt the above procedure to allow discovering approximations of the minimal optimal repair recommendations for S w.r.t. L as per f restricted by C as follows.

(1) Identify legal moves $X \subseteq \mathbb{M}_S$ that contribute to the cost of optimal alignment between L and S.
(2) Decide to exploit a legal move $x \in X$ that contributes the most to the cost of optimal alignment between L and S per one unit of cost required to carry out a repair of S that caters for x and is feasible given the amount of available repair resources.
(3) Subordinate all the available repair resources to repair S to cater x.
(4) Elevate x by repairing S to obtain net system S' that caters for x.
(5) If S' caters for x, go back to step 1 to repair S' using the remaining repair resources, but do not allow legal move x to contribute to the cost of optimal alignment between L and any of the net systems obtained by repairing S' in the subsequent iterations of the procedure.

Algorithm 5 formalizes the above devised procedure.

Line 1 of the algorithm initializes control variables and variables to be used for storing the result. The main logic of the algorithm is encoded in the `repeat` loop of lines 2–28. Set $\mathcal{X}$ contains repair recommendations to be explored for 'refinement' in the current iteration of the loop. The constructed refined repair recommendations are collected in set $\mathcal{R}$. The first iteration of the loop starts by initializing $\mathcal{X}$ with the set that only contains the empty recommendation $(\varnothing, \varnothing)$. This empty recommendation is gradually refined in the course of the execution of the algorithm into the result repair recommendation(s). Execution of the `repeat` loop terminates after its iteration completes without discovering any refined recommendation, i.e., $\mathcal{R} = \varnothing$ at line 28 of the algorithm. Otherwise, the discovered refined recommendations are passed to the next iteration of the loop to be explored for further refinements. After the `repeat` loop of lines 2–28 terminates, the `foreach` loop of lines 29–32 constructs the result of the algorithm by selecting each repair recommendation R in $\mathcal{X}$ (these are recommendations that were explored for refinement in the last iteration of the `repeat` loop of lines 2–28) that leads to the lowest cost of optimal alignment between the input event log and some net system obtained by performing an optimal alignment-based repair of the input system using R.

Every iteration of the `foreach` loop of lines 4–27 searches for feasible refinements of some repair recommendation $R \in \mathcal{X}$. Once execution of this loop reaches line 14, function w_2 maps each

ALGORITHM 5: *GoldrattApproximateRecommendationSearch*$(S, L, f, C, singleton)$

Input: A net system S, an event log L, a cost function on label moves f, a repair constraint $C := (C_{ins}, C_{skp}, res)$, and a Boolean value *singleton*.

Output: An ordered pair $(\mathcal{R}, optCost)$, where $\mathcal{R}$ is a set of *approximate* minimal optimal repair recommendations for S w.r.t. L as per f restricted by C, and *optCost* is the cost of optimal alignment between L and S such that $\forall R \in \mathcal{R} : cost[L, S, c_R] = optCost$, where c_R is the cost function on legal moves over S induced by $f|_R$.

$\mathcal{R} := \{(\varnothing, \varnothing)\}$; $\mathcal{X} := \varnothing$; $costs := \varnothing$; $optCost := +\infty$;
repeat
  $\mathcal{X} := \mathcal{R}$; $\mathcal{R} := \varnothing$; $mci := 0$; $mrc := 0$; $visited := \varnothing$;
  foreach $R := (R_{ins}, R_{skp}) \in \mathcal{X}$ **do**
    c_R is the cost function on legal moves over S induced by $f|_R$;
    $X := \{(\upsilon, \gamma) \mid \upsilon \in L \wedge \gamma \in \mathbb{O}^{\upsilon}_{S, c_R}\}$;
    $Y \subseteq X$ such that $\bigcup_{y \in Y} \{\pi_1(y)\} = Set(L)$ and $\forall\, y_1 \in Y\ \forall y_2 \in Y : (\pi_1(y_1) = \pi_1(y_2)) \Rightarrow y_1 = y_2$;
    $costs := costs \cup \{(R, \sum_{(\upsilon,\gamma) \in Y} (L(\upsilon) \times c_R(\gamma)))\}$;
    $w_1 := \{(((\{x\}, \varnothing), w) \mid x \in labels(L) \setminus R_{ins} \wedge w = f((x, \gg)) \times count((x, \gg), Y, L, S) \wedge w > 0\} \cup$
      $\{((\varnothing, \{x\}), w) \mid x \in labels(S) \setminus R_{skp} \wedge w = f((\gg, x)) \times count((\gg, x), Y, L, S) \wedge w > 0\}$;
    $free :=$ **false**; $w_2 := \varnothing$;
    if $\exists (r, w) \in w_1 : cost[r, C] = 0$ **then**
      $free :=$ **true**; $w_2 := w_1$;
    else $w_2 := \{(((\{x\}, \varnothing), w) \mid (\{x\}, \varnothing) \in dom(w_1) \wedge w = w_1((\{x\}, \varnothing))/C_{ins}(x)\} \cup$
      $\{((\varnothing, \{x\}), w) \mid (\varnothing, \{x\}) \in dom(w_1) \wedge w = w_1((\varnothing, \{x\}))/C_{skp}(x)\}$;
    foreach $(r, cci) \in w_2$ **do**
      $R' := R \cup r$; $crc := cost[r, C]$;
      if $R' \notin visited$ **and** $cost[R', C] \leq res$ **then**
        $visited := visited \cup \{R'\}$;
        if $crc = 0$ **and** $cci > mci$ **and** *free* **then**
          $\mathcal{R} := \{R'\}$; $mci := cci$;
        else if $crc = 0$ **and** $cci > 0$ **and** $cci = mci$ **and** *free* **and not** *singleton* **then**
          $\mathcal{R} := \mathcal{R} \cup \{R'\}$;
        else if $cci > mci$ **and not** *free* **then**
          $\mathcal{R} := \{R'\}$; $mrc := crc$; $mci := cci$;
        else if $crc > mrc$ **and** $cci > 0$ **and** $cci = mci$ **and not** *free* **then**
          $\mathcal{R} := \{R'\}$; $mrc := crc$;
        else if $crc > 0$ **and** $crc = mrc$ **and** $cci = mci$ **and not** *free* **and not** *singleton* **then**
          $\mathcal{R} := \mathcal{R} \cup \{R'\}$;
until $\mathcal{R} \neq \varnothing$;
foreach $R \in \mathcal{X}$ **do**
  if $costs(R) < optCost$ **then**
    $optCost := costs(R)$; $\mathcal{R} := \varnothing$;
  if $costs(R) = optCost$ **then** $\mathcal{R} := \mathcal{R} \cup \{R\}$;
return $(\mathcal{R}, optCost)$;

elementary repair recommendation that can be used to refine R to its weight. If some elementary recommendation r can be repaired for free, i.e., the cost of this elementary repair recommendation as per the input repair constraint C is equal to zero, weights are computed (and stored in w_1) as contributions of legal moves that are recommended for repair by r to the cost of optimal alignment between L and some net system obtained by performing an optimal alignment-based repair of S using

R, refer to lines 9–12 of the algorithm. Otherwise, the weight of an elementary recommendation r in w_2 is derived by dividing the weight of r in w_1 by the cost of r as per C, refer to line 13 of Algorithm 5, i.e., w_2 stores 'impacts' of elementary repair recommendations on the cost of optimal alignment per unit of repair resource.

The `foreach` loop of lines 14–27 is used to select elementary repair recommendation(s) with the highest impact on the cost of optimal alignment, as per w_2, to be used to refine R. The selection process is designed to account for different values of the input *singleton* parameter and correct handling of free and non-free recommendations. Execution of the body of the loop is controlled by the variables *cci*, *mci*, *crc*, and *mrc*. They are used to store values of weight of the currently analyzed repair recommendation (current cost improvement), maximal observed weight of repair recommendation (maximal cost improvement), cost of current repair recommendation (current repair cost), and maximal observed cost of repair recommendation (maximal repair cost), respectively.

Let $C := (C_{ins}, C_{skp}, 6)$ be the standard repair constraint defined for $\mathcal{A} = \{\mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{e},\mathtt{f},\mathtt{g},\mathtt{h},\mathtt{x}\}$. Then, $(\{(\{\mathtt{f},\mathtt{e},\mathtt{a},\mathtt{x}\},\{\mathtt{g},\mathtt{e}\})\},45) = \mathit{GoldrattApproximateRecommendationSearch}(S, L_3, f, C, \mathit{true})$, where S is the net system in Fig. 2, L_3 is the event log from Section 2.3, and f is the standard cost function on label moves for events and activities $\mathcal{A}$. The execution of Algorithm 5 on the above input starts with the empty repair recommendation $R := (\varnothing, \varnothing)$ at line 4. Then, the algorithm approximates which extensions of R will have the best impact on repair of S. It achieves this by constructing weights of all possible extensions. When the algorithm reaches line 14 for the first time, set w_2 is given by $\{((\{\mathtt{a}\},\varnothing),9),((\{\mathtt{c}\},\varnothing),9),((\{\mathtt{d}\},\varnothing),7),((\{\mathtt{e}\},\varnothing),17),((\{\mathtt{f}\},\varnothing),10),((\{\mathtt{x}\},\varnothing),9),((\varnothing,\{\mathtt{a}\}),7),((\varnothing,\{\mathtt{c}\}),17),((\varnothing,\{\mathtt{d}\}),15),((\varnothing,\{\mathtt{f}\}),2),((\varnothing,\{\mathtt{g}\}),12),((\varnothing,\{\mathtt{h}\}),6)\}$. Because the input parameter *singleton* is set to true, the repair recommendation $(\{\mathtt{e}\},\varnothing)$ is chosen nondeterministically by the `foreach` loop of lines 14–27 to be used as the only element of set $\mathcal{R}$ in the second iteration of the repeat loop of lines 2–28. Note that $(\{\mathtt{e}\},\varnothing)$, according to the data in w_2, has the best 'impact' of seventeen cost units on the cost of optimal alignment between $L3$ and S as per the cost function on legal moves over S induced by f; the impact is determined based on the optimal alignments γ_0^1–γ_0^7 between traces in L_3 and S shown on Page 15, i.e., it holds that $\mathit{count}((\mathtt{e},\gg),Y,L_3,S) = 17$. In the second iteration of the `repeat` loop of lines 2–28, the repair recommendation $(\{\mathtt{e}\},\varnothing)$ gets extended by $(\varnothing,\{\mathtt{g}\})$ to obtain $\mathcal{R} := \{(\{\mathtt{e}\},\{\mathtt{g}\})\}$ at the moment the algorithm reaches line 28 for the second time. The repair recommendation $(\varnothing,\{\mathtt{g}\})$ is chosen as it has the best impact on optimal alignments between traces in L_3 and the repaired version of S using the recommendation $(\{\mathtt{e}\},\varnothing)$; set w_2 at line 14 of the second iteration of the repeat loop of lines 2–28 is given by $\{((\{\mathtt{a}\},\varnothing),9),((\{\mathtt{c}\},\varnothing),9),((\{\mathtt{d}\},\varnothing),9),((\{\mathtt{f}\},\varnothing),10),((\{\mathtt{x}\},\varnothing),9),((\varnothing,\{\mathtt{a}\}),7),((\varnothing,\{\mathtt{c}\}),6),((\varnothing,\{\mathtt{d}\}),6),((\varnothing,\{\mathtt{e}\}),9),((\varnothing,\{\mathtt{f}\}),2),((\varnothing,\{\mathtt{g}\}),12),((\varnothing,\{\mathtt{h}\}),6)\}$. Note that these impact values are determined based on optimal alignment as per the cost function on legal moves over S induced by $f|_{(\{\mathtt{e}\},\varnothing)}$. The subsequent iterations of the repeat loop of lines 2–28 end with the value of $\mathcal{R}$ given by $\{(\{\mathtt{f},\mathtt{e}\},\{\mathtt{g}\})\}$, $\{(\{\mathtt{f},\mathtt{e},\mathtt{x}\},\{\mathtt{g}\})\}$, $\{(\{\mathtt{f},\mathtt{e},\mathtt{a},\mathtt{x}\},\{\mathtt{g}\})\}$, $\{(\{\mathtt{f},\mathtt{e},\mathtt{a},\mathtt{x}\},\{\mathtt{g},\mathtt{e}\})\}$, and $\varnothing$. The last iteration ends with the empty set value of $\mathcal{R}$ because the repair recommendation constructed in the previous iteration uses all the available repair resources, i.e., it holds that $\mathit{cost}[(\{\mathtt{f},\mathtt{e},\mathtt{a},\mathtt{x}\},\{\mathtt{g},\mathtt{e}\}),C] = 6$. This repair recommendation is returned as the result of the algorithm call together with the corresponding cost of optimal alignment between L_3 and the repaired version of S at line 33 after constructing the result in the `foreach` loop of lines 29–32.

Fig. 8 shows net system $\hat{S}$ that is obtained as a result of a naïve repair of net system S in Fig. 2 using repair recommendation $R := (\{\mathtt{f},\mathtt{e},\mathtt{a},\mathtt{x}\},\{\mathtt{g},\mathtt{e}\})$; the 'injected' transitions are highlighted with grey background. This repair is performed based on optimal alignments between traces in L_3 and S as per the cost function on legal moves over S induced by f. Optimal alignments between traces in L_3 and $\hat{S}$ as per the cost function c on legal moves over $\hat{S}$ induced by f are listed below.

$\gamma_8^1 :=$

a	b	c	f	d	e	≫	f	≫	≫
a	b	c	f	d	e	τ	f	τ	τ
t_1	t_2	t_4	t_{13}	t_5	t_{14}	t_7	t_8	t_{19}	t_{11}

$\gamma_8^2 :=$

a	c	≫	d	≫	c	≫	e	d	≫	g	f	≫
a	c	τ	d	τ	c	τ	e	d	τ	g	f	τ
t_1	t_4	t_3	t_5	t_{18}	t_4	t_3	t_{12}	t_5	t_7	t_9	t_8	t_{11}

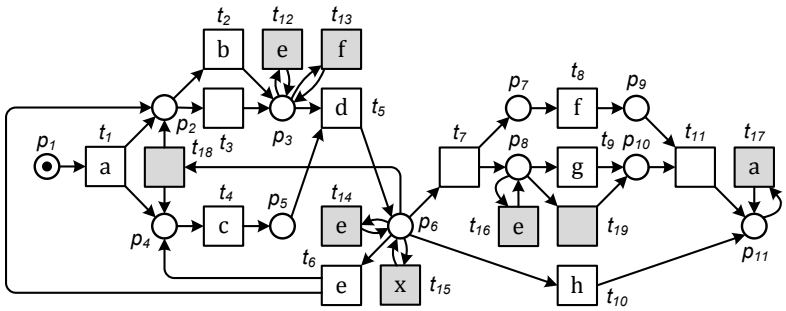


Fig. 8: A net system obtained as the result of a naïve repair of the net system in Fig. 2 using repair recommendation $R := (\{\mathtt{f},\mathtt{e},\mathtt{a},\mathtt{x}\},\{\mathtt{g},\mathtt{e}\})$ discovered using Algorithm 5.

$$\gamma_8^3 := \begin{array}{|c|c|c|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\mathtt{x}&\mathtt{c}&\mathtt{h}&\mathtt{a}\\ \hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\mathtt{x}&\gg&\mathtt{h}&\mathtt{a}\\ \hline t_1&t_2&t_4&t_5&t_{14}&t_{15}&&t_{10}&t_{17}\\ \hline\end{array} \qquad \gamma_8^4 := \begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline \gg&\mathtt{c}&\gg&\mathtt{d}&\gg&\mathtt{d}&\mathtt{f}&\mathtt{e}&\mathtt{g}&\gg\\ \hline \mathtt{a}&\mathtt{c}&\tau&\mathtt{d}&\tau&\gg&\mathtt{f}&\mathtt{e}&\mathtt{g}&\tau\\ \hline t_1&t_4&t_3&t_5&t_7&&t_8&t_{16}&t_9&t_{11}\\ \hline\end{array}$$

$$\gamma_8^5 := \begin{array}{|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{b}&\gg&\gg&\gg\\ \hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{h}\\ \hline t_1&t_2&t_4&t_5&t_{10}\\ \hline\end{array} \qquad \gamma_8^6 := \begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\gg&\mathtt{d}&\mathtt{f}&\gg&\gg\\ \hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\tau&\gg&\mathtt{f}&\tau&\tau\\ \hline t_1&t_2&t_4&t_5&t_{14}&t_7&&t_8&t_{19}&t_{11}\\ \hline\end{array}$$

$$\gamma_8^7 := \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\gg&\mathtt{g}&\gg&\gg\\ \hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\tau&\mathtt{g}&\mathtt{f}&\tau\\ \hline t_1&t_2&t_4&t_5&t_6&t_2&t_4&t_5&t_7&t_9&t_8&t_{11}\\ \hline\end{array}$$

One can verify that $c(\gamma_8^1) = 0$, $c(\gamma_8^2) = 0$, $c(\gamma_8^3) = 1$, $c(\gamma_8^4) = 2$, $c(\gamma_8^5) = 3$, $c(\gamma_8^6) = 1$, and $c(\gamma_8^7) = 1$, and, thus, it holds that $cost[L_3,\hat{S},c] = 10\times0+9\times0+9\times1+7\times2+6\times3+2\times1+2\times1 = 45$.

Note that it holds that $(\{(\{\mathtt{f},\mathtt{c},\mathtt{x}\},\{\mathtt{g},\mathtt{d},\mathtt{c}\}),(\{\mathtt{f},\mathtt{c},\mathtt{a}\},\{\mathtt{g},\mathtt{d},\mathtt{c}\}),(\{\mathtt{f},\mathtt{a},\mathtt{x}\},\{\mathtt{g},\mathtt{d},\mathtt{c}\})\},38) = \mathit{GoldrattApproximateRecommendationSearch}(S,L_3,f,C,\mathit{false})$.

Fig. 9 shows net system $\check{S}$ that is obtained as a result of a naïve repair of S using repair recommendation $R := (\{\mathtt{f},\mathtt{c},\mathtt{x}\},\{\mathtt{g},\mathtt{d},\mathtt{c}\})$; the 'injected' transitions are highlighted with grey background. Optimal alignments between traces in L_3 and $\check{S}$ as per the cost function c on legal moves over $\check{S}$ induced by f are listed below.

$$\gamma_9^1 := \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{f}&\mathtt{d}&\mathtt{e}&\gg&\gg&\gg&\gg&\mathtt{f}&\gg&\gg\\ \hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{f}&\mathtt{d}&\mathtt{e}&\tau&\tau&\tau&\tau&\mathtt{f}&\tau&\tau\\ \hline t_1&t_2&t_4&t_{12}&t_5&t_6&t_3&t_{15}&t_{16}&t_7&t_8&t_{17}&t_{11}\\ \hline\end{array} \qquad \gamma_9^2 := \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{c}&\gg&\mathtt{d}&\mathtt{c}&\mathtt{e}&\gg&\gg&\mathtt{d}&\gg&\mathtt{g}&\mathtt{f}&\gg\\ \hline \mathtt{a}&\mathtt{c}&\tau&\mathtt{d}&\mathtt{c}&\mathtt{e}&\tau&\tau&\mathtt{d}&\tau&\mathtt{g}&\mathtt{f}&\tau\\ \hline t_1&t_4&t_3&t_5&t_{14}&t_6&t_3&t_{15}&t_5&t_7&t_9&t_8&t_{11}\\ \hline\end{array}$$

$$\gamma_9^3 := \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\mathtt{x}&\mathtt{c}&\gg&\gg&\mathtt{h}&\mathtt{a}\\ \hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\mathtt{x}&\mathtt{c}&\tau&\tau&\mathtt{h}&\gg\\ \hline t_1&t_2&t_4&t_5&t_6&t_{13}&t_4&t_3&t_{16}&t_{10}&\\ \hline\end{array} \qquad \gamma_9^4 := \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|}\hline \gg&\mathtt{c}&\gg&\mathtt{d}&\gg&\mathtt{d}&\mathtt{f}&\mathtt{e}&\mathtt{g}&\gg\\ \hline \mathtt{a}&\mathtt{c}&\tau&\mathtt{d}&\tau&\gg&\mathtt{f}&\gg&\mathtt{g}&\tau\\ \hline t_1&t_4&t_3&t_5&t_7&&t_8&&t_9&t_{11}\\ \hline\end{array}$$

$$\gamma_9^5 := \begin{array}{|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{b}&\gg&\gg&\gg\\ \hline \mathtt{a}&\mathtt{b}&\tau&\tau&\mathtt{h}\\ \hline t_1&t_2&t_{15}&t_{16}&t_{10}\\ \hline\end{array} \qquad \gamma_9^6 := \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\gg&\gg&\mathtt{d}&\gg&\mathtt{f}&\gg&\gg\\ \hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\tau&\tau&\mathtt{d}&\tau&\mathtt{f}&\tau&\tau\\ \hline t_1&t_2&t_4&t_5&t_6&t_3&t_{15}&t_5&t_7&t_8&t_{17}&t_{11}\\ \hline\end{array}$$

$$\gamma_9^7 := \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\gg&\mathtt{g}&\gg&\gg\\ \hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\tau&\mathtt{g}&\mathtt{f}&\tau\\ \hline t_1&t_2&t_4&t_5&t_6&t_2&t_4&t_5&t_7&t_9&t_8&t_{11}\\ \hline\end{array}$$

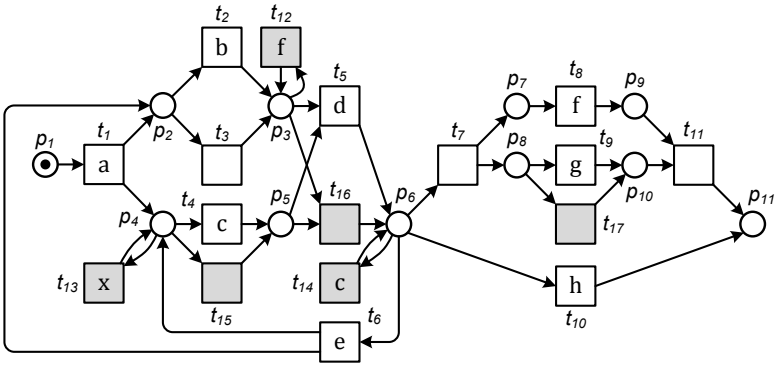


Fig. 9: A net system obtained as the result of a naïve repair of the net system in Fig. 2 using repair recommendation $R := (\{\mathtt{f},\mathtt{c},\mathtt{x}\},\{\mathtt{g},\mathtt{d},\mathtt{c}\})$ discovered using Algorithm 5.

Using the above alignments, one can easily verify that, indeed, $c(\gamma_9^1) = 0$, $c(\gamma_9^2) = 0$, $c(\gamma_9^3) = 1$, $c(\gamma_9^4) = 3$, $c(\gamma_9^5) = 1$, $c(\gamma_9^6) = 0$, and $c(\gamma_9^7) = 1$, and, thus, it holds that $cost[L_3,\check{S},c] = 10\times 0 + 9\times 0 + 9\times 1 + 7\times 3 + 6\times 1 + 2\times 0 + 2\times 1 = 38$.

If Algorithm 5 is invoked for the value of the *singleton* parameter of true and a standard repair constraint $C := (C_{ins}, C_{skp}, res)$, then it performs $1 + res$ computations of optimal alignments between traces in the input event log and net system (as per different cost functions); one per each iteration of the `repeat` loop of lines 2–28. Note that the same set of optimal alignments is used to compute cost of optimal alignment at line 8 and weights of elementary repair recommendations at line 9.

6.5. Greedy Search

A greedy algorithm is a heuristic that proceeds by taking the locally optimal intermediate decisions with the expectation of finding a globally optimal solution to the problem. A greedy strategy often fails to discover an optimal solution. However, it may yield locally optimal solutions to the problem that approximate a global optimal solution in a reasonable time. In this subsection, we propose a design of a greedy strategy for approximating minimal optimal repair recommendations. This design is captured in the pseudocode of Algorithm 6.

Algorithm 6 iteratively constructs repair recommendations by 'expanding' currently discovered ones with elementary repair recommendations that lead to the best improvement in the cost of optimal alignment with the expectation of finding a minimal optimal repair recommendation. The set that contains one empty repair recommendation and no other recommendations is used as a starting point for discovering the resulting recommendations, cf. initialization of set $\mathcal{X}$ at line 2 of Algorithm 6. During execution of the algorithm, set $\mathcal{X}$ contains currently discovered repair recommendations to be tested for expansions. The main logic of the algorithm is encoded in the `repeat` loop of lines 3–30, which terminates once the algorithm depletes elements in $\mathcal{X}$, i.e., $\mathcal{X} = \varnothing$.

Every iteration of the `foreach` loop of lines 7–28 visits and tests one repair recommendation for expansion by elementary repair recommendations (one in every iteration of the `foreach` loop of lines 8–27). The expansion takes place at line 9. If the expanded recommendation was not already visited and is feasible, refer to line 10 in the algorithm, it is tested to be included in set $\mathcal{X}$ and, thus, tested for further expansions in the next iteration of the `repeat` loop. The test relies on the use of control variables. These variables store values for maximal cost improvement (*mci*), current cost improvement (*cci*), maximal repair cost (*mrc*), current repair cost (*crc*), and current cost of optimal alignment (*cCost*). The Boolean variable *free* is used to control special handling of situations in which repair recommendation expansions do not lead to increases in repair costs.

Let, again, $C := (C_{ins}, C_{skp}, 6)$ be the standard repair constraint defined under the assumption that $\mathcal{A} = \{\mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{e},\mathtt{f},\mathtt{g},\mathtt{h},\mathtt{x}\}$. Then, one can employ Algorithm 6 to verify that $(\{(\{\mathtt{f},\mathtt{d},\mathtt{e},\mathtt{a}\},\{\mathtt{g},\mathtt{d}\})\},39) =$

ALGORITHM 6: *GreedyApproximateRecommendationSearch*$(S, L, f, C, singleton)$

Input: A net system S, an event log L, a cost function on label moves f, a repair constraint $C := (C_{ins}, C_{skp}, res)$, and a Boolean value *singleton*.

Output: An ordered pair $(\mathcal{R}, optCost)$, where $\mathcal{R}$ is a set of *approximate* minimal optimal repair recommendations for S w.r.t. L as per f restricted by C, and *optCost* is the cost of optimal alignment between L and S such that $\forall R \in \mathcal{R} : cost[L, S, c_R] = optCost$, where c_R is the cost function on legal moves over S induced by $f|_R$.

```
1  c is the cost function on legal moves over S induced by f;
2  R := ∅; X := {(∅,∅)}; optCost := cost[L,S,c]; cCost := optCost;
3  repeat
4      R := X;
5      if optCost = 0 then return (R, optCost);
6      X := ∅; mci := 0; mrc := 0; visited := ∅; free := false;
7      foreach (R_ins, R_skp) ∈ R do
8          foreach r ∈ {({x},∅) | x ∈ labels(L) \ R_ins} ∪ {(∅,{x}) | x ∈ labels(S) \ R_skp} do
9              R := (R_ins, R_skp) ∪ r;
10             if R ∉ visited and cost[R,C] ≤ res then
11                 aCost := optCost; crc := cost[r,C];
12                 c_R is the cost function on legal moves over S induced by f|_R;
13                 if crc = 0 or (not free and optCost/crc ≥ mci) then aCost := cost[L,S,c_R];
14                 cci := optCost − aCost;
15                 if crc > 0 then cci := cci/crc;
16                 if crc > 0 and cci > mci and not free then
17                     mci := cci; mrc := crc; X := {R}; cCost := aCost;
18                 else if crc > mrc and cci > 0 and cci = mci and not free then
19                     mrc := crc; X := {R}; cCost := aCost;
20                 else if crc > 0 and crc = mrc and cci > 0 and cci = mci and
                           not free and not singleton then
21                     X := X ∪ {R};
22                 else if crc = 0 and cci > mci and free then
23                     mci := cci; X := {R}; cCost := aCost;
24                 else if crc = 0 and cci = mci and free and not singleton then
25                     X := X ∪ {R};
26                 else if crc = 0 and cci > 0 and not free then
27                     free := true; mci := cci; X := {R}; cCost := aCost;
28             visited := visited ∪ {R};
29     optCost := cCost;
30 until X ≠ ∅;
31 return (R, optCost);
```

GreedyApproximateRecommendationSearch$(S, L_3, f, C, true)$, where S is the system in Fig. 2, L_3 is the event log from Section 2.3, and f is the standard cost function on label moves. Algorithm 6 starts with the empty repair recommendation and then proceeds, in the `foreach` loop of lines 8–28, by exploring all its possible extensions with elementary repair recommendations. In each iteration of the loop, one extended recommendation is constructed and the cost of optimal alignment between L_3 and the repaired version of S using the corresponding repair recommendation is computed at line 13 of the algorithm. Thus, in the first iteration of the `repeat` loop of lines 3–30, the following repair recommendation are explored: $(\{\texttt{a}\}, \varnothing)$ with the cost of optimal alignment between L_3 and the repaired version of S using this repair recommendation of 111, $(\{\texttt{b}\}, \varnothing)$ with the cost of 120, $(\{\texttt{c}\}, \varnothing)$ with the cost of 111, $(\{\texttt{d}\}, \varnothing)$ with the cost of 113, $(\{\texttt{e}\}, \varnothing)$ with the cost of 94, $(\{\texttt{f}\}, \varnothing)$

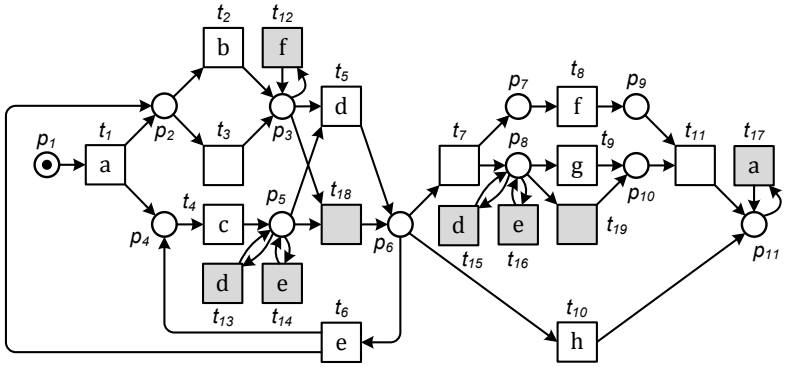


Fig. 10: A net system obtained as the result of a naïve repair of the net system in Fig. 2 using repair recommendation $R := (\{\mathtt{f},\mathtt{d},\mathtt{e},\mathtt{a}\},\{\mathtt{g},\mathtt{d}\})$ discovered using Algorithm 6.

with the cost of 110, $(\{\mathtt{g}\},\varnothing)$ with the cost of 120, $(\{\mathtt{h}\},\varnothing)$ with the cost of 120, $(\{\mathtt{x}\},\varnothing)$ with the cost of 111, $(\varnothing,\{\mathtt{a}\})$ with the cost of 113, $(\varnothing,\{\mathtt{b}\})$ with the cost of 120, $(\varnothing,\{\mathtt{c}\})$ with the cost of 103, $(\varnothing,\{\mathtt{d}\})$ with the cost of 105, $(\varnothing,\{\mathtt{e}\})$ with the cost of 111, $(\varnothing,\{\mathtt{f}\})$ with the cost of 118, $(\varnothing,\{\mathtt{g}\})$ with the cost of 108, and $(\varnothing,\{\mathtt{h}\})$ with the cost of 114. Because repair recommendation $(\{\mathtt{e}\},\varnothing)$ leads to the best cost of optimal alignment of 94, set $\mathcal{X}$ is equal to $\{(\{\mathtt{e}\},\varnothing)\}$ at the end of the first iteration of the repeat loop of lines 3–30. In the second iteration of this loop, sixteen distinct extensions of $(\{\mathtt{e}\},\varnothing)$ are explored and the best extension of $(\{\mathtt{e}\},\{\mathtt{g}\})$ that leads to the cost of optimal alignment of 82 is chosen. In the third iteration, fifteen further extensions are explored and $(\{\mathtt{e}\},\{\mathtt{g},\mathtt{d}\})$ with the cost of 67 is chosen. In the fourth iteration, fourteen extensions are explored and $(\{\mathtt{f},\mathtt{e}\},\{\mathtt{g},\mathtt{d}\})$ with the cost of 57 is chosen. In the fifth iteration, thirteen extensions are explored and $(\{\mathtt{f},\mathtt{e},\mathtt{a}\},\{\mathtt{g},\mathtt{d}\})$ with the cost of 48 is chosen. Finally, in the sixth iteration of the `foreach` loop of lines 3–30, twelve extensions are explored and $(\{\mathtt{f},\mathtt{d},\mathtt{e},\mathtt{a}\},\{\mathtt{g},\mathtt{d}\})$ with the cost of optimal alignment of 39 is chosen as the one that can be used to perform the best repair of S. This repair recommendation cannot be extended further as every its extensions will have the cost that exceeds the amount of available repair resources *res*. Finally, at line 31, this repair recommendation and the cost of 39 are returned as the result of the call to Algorithm 6.

Fig. 10 shows net system $\breve{S}$ that is obtained as a result of a naïve repair of S using repair recommendation $R := (\{\mathtt{f},\mathtt{d},\mathtt{e},\mathtt{a}\},\{\mathtt{g},\mathtt{d}\})$; the 'injected' transitions are highlighted with grey background. This repair is performed based on optimal alignments between traces in L_3 and S as per the cost function on legal moves over S induced by f. Optimal alignments between traces in L_3 and $\breve{S}$ as per the cost function c on legal moves over $\breve{S}$ induced by f are listed below.

$$\gamma^1_{10} := \begin{array}{|c|c|c|c|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{f}&\mathtt{d}&\gg&\mathtt{e}&\mathtt{f}&\gg&\gg\\ \hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{f}&\mathtt{d}&\tau&\mathtt{e}&\mathtt{f}&\tau&\tau\\ \hline t_1&t_2&t_4&t_{12}&t_5&t_7&t_{16}&t_8&t_{19}&t_{11}\\ \hline\end{array} \qquad \gamma^2_{10} := \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{c}&\mathtt{d}&\mathtt{c}&\mathtt{e}&\gg&\mathtt{d}&\gg&\mathtt{g}&\mathtt{f}&\gg\\ \hline \mathtt{a}&\mathtt{c}&\mathtt{d}&\gg&\mathtt{e}&\tau&\mathtt{d}&\tau&\mathtt{g}&\mathtt{f}&\tau\\ \hline t_1&t_4&t_{13}&&t_{14}&t_3&t_5&t_7&t_9&t_8&t_{11}\\ \hline\end{array}$$

$$\gamma^3_{10} := \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\mathtt{x}&\mathtt{c}&\gg&\gg&\mathtt{h}&\mathtt{a}\\ \hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\gg&\mathtt{c}&\tau&\tau&\mathtt{h}&\mathtt{a}\\ \hline t_1&t_2&t_4&t_5&t_6&&t_4&t_3&t_{18}&t_{10}&t_{17}\\ \hline\end{array} \qquad \gamma^4_{10} := \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|}\hline \gg&\mathtt{c}&\mathtt{d}&\gg&\mathtt{d}&\gg&\mathtt{f}&\mathtt{e}&\mathtt{g}&\gg\\ \hline \mathtt{a}&\mathtt{c}&\mathtt{d}&\tau&\mathtt{d}&\tau&\mathtt{f}&\mathtt{e}&\mathtt{g}&\tau\\ \hline t_1&t_4&t_{13}&t_3&t_5&t_7&t_8&t_{16}&t_9&t_{11}\\ \hline\end{array}$$

$$\gamma^5_{10} := \begin{array}{|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{b}&\gg&\gg&\gg\\ \hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\tau&\mathtt{h}\\ \hline t_1&t_2&t_4&t_{18}&t_{10}\\ \hline\end{array} \qquad \gamma^6_{10} := \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\mathtt{d}&\gg&\mathtt{f}&\gg&\gg\\ \hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\mathtt{d}&\tau&\mathtt{f}&\tau&\tau\\ \hline t_1&t_2&t_4&t_{13}&t_{14}&t_5&t_7&t_8&t_{19}&t_{11}\\ \hline\end{array}$$

$$\gamma^7_{10} := \begin{array}{|c|c|c|c|c|c|c|c|c|c|c|c|}\hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\gg&\mathtt{g}&\gg&\gg\\ \hline \mathtt{a}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\mathtt{e}&\mathtt{b}&\mathtt{c}&\mathtt{d}&\tau&\mathtt{g}&\mathtt{f}&\tau\\ \hline t_1&t_2&t_4&t_5&t_6&t_2&t_4&t_5&t_7&t_9&t_8&t_{11}\\ \hline\end{array}$$

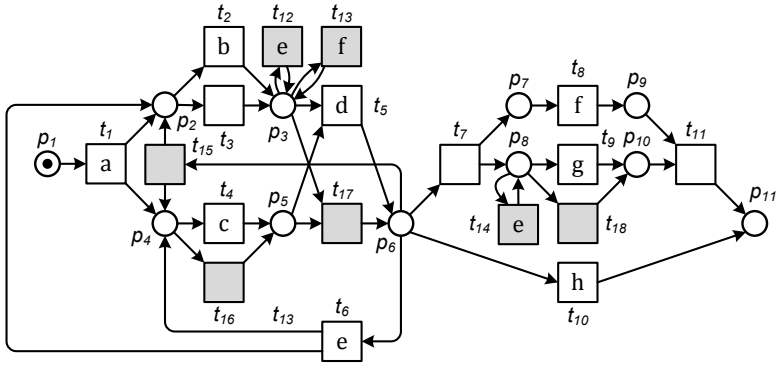


Fig. 11: A net system obtained as the result of a naïve repair of the net system in Fig. 2 using repair recommendation $R := (\{\mathtt{f},\mathtt{e}\},\{\mathtt{g},\mathtt{d},\mathtt{e},\mathtt{c}\})$ discovered using Algorithm 6.

One can use the above alignments to verify that $c(\gamma^1_{10}) = 0$, $c(\gamma^2_{10}) = 1$, $c(\gamma^3_{10}) = 1$, $c(\gamma^4_{10}) = 1$, $c(\gamma^5_{10}) = 2$, $c(\gamma^6_{10}) = 0$, and $c(\gamma^7_{10}) = 1$, and, thus, it holds that $cost[L_3,\check{S},c] = 10\times0+9\times1+9\times1+7\times1+6\times2+2\times0+2\times1 = 39$.

Note that one can obtain a better result if the input Boolean parameter *singleton* is set to false. In particular, it holds that $GreedyApproximateRecommendationSearch(S,L_3,f,C,false) = (\{(\{\mathtt{f},\mathtt{e}\},\{\mathtt{g},\mathtt{d},\mathtt{e},\mathtt{c}\}),33)$.

Fig. 11 shows net system $\tilde{S}$ that is obtained as a result of a naïve repair of net system S in Fig. 2 using repair recommendation $R := (\{\mathtt{f},\mathtt{e}\},\{\mathtt{g},\mathtt{d},\mathtt{e},\mathtt{c}\})$; the 'injected' transitions are highlighted with grey background. This repair is performed based on optimal alignments between traces in L_3 and S as per the cost function on legal moves over S induced by f. Optimal alignments between traces in L_3 and $\tilde{S}$ as per the cost function c on legal moves over $\tilde{S}$ induced by f are listed below.

$\gamma^1_{11}:=$

a	b	c	f	d	≫	e	f	≫	≫
a	b	c	f	d	τ	e	f	τ	τ
t_1	t_2	t_4	t_{13}	t_5	t_7	t_{14}	t_8	t_{18}	t_{11}

$\gamma^2_{11}:=$

a	c	≫	d	≫	c	≫	e	d	≫	g	f	≫
a	c	τ	d	τ	c	τ	e	d	τ	g	f	τ
t_1	t_4	t_3	t_5	t_{15}	t_4	t_3	t_{12}	t_5	t_7	t_9	t_8	t_{11}

$\gamma^3_{11}:=$

a	b	c	d	e	x	c	≫	≫	h	a
a	b	c	d	e	≫	c	τ	τ	h	≫
t_1	t_2	t_4	t_5	t_6		t_4	t_3	t_{17}	t_{10}	

$\gamma^4_{11}:=$

≫	c	≫	d	≫	≫	≫	d	≫	f	e	g	≫
a	c	τ	d	τ	τ	τ	d	τ	f	e	g	τ
t_1	t_4	t_3	t_5	t_{15}	t_3	t_{16}	t_5	t_7	t_8	t_{14}	t_9	t_{11}

$\gamma^5_{11}:=$

a	b	≫	≫	≫
a	b	τ	τ	h
t_1	t_2	t_{16}	t_{17}	t_{10}

$\gamma^6_{11}:=$

a	b	c	d	e	≫	≫	d	≫	f	≫	≫
a	b	c	d	e	τ	τ	d	τ	f	τ	τ
t_1	t_2	t_4	t_5	t_6	t_3	t_{16}	t_5	t_7	t_8	t_{18}	t_{11}

$\gamma^7_{11}:=$

a	b	c	d	e	b	c	d	≫	g	≫	≫
a	b	c	d	e	b	c	d	τ	g	f	τ
t_1	t_2	t_4	t_5	t_6	t_2	t_4	t_5	t_7	t_9	t_8	t_{11}

It holds that $c(\gamma^1_{11}) = 0$, $c(\gamma^2_{11}) = 0$, $c(\gamma^3_{11}) = 1$, $c(\gamma^4_{11}) = 1$, $c(\gamma^5_{11}) = 2$, $c(\gamma^6_{11}) = 0$, $c(\gamma^7_{11}) = 1$, and, thus, $cost[L_3,\tilde{S},c] = 10\times0+9\times0+9\times2+7\times1+6\times1+2\times0+2\times1 = 33$.

When Algorithm 6 is invoked for the value of the *singleton* parameter of true and a standard repair constraint $C := (C_{ins},C_{skp},res)$, it performs at most $1+\sum_{i=0}^{i<res}(|labels(L)|+|labels(S)|-i)$ computations of optimal alignments between traces in the input event log L and system S (as per different cost functions). The maximal number of optimal alignment computations is observed when

the given amount of repair resources *res* is not sufficient to discover a repair recommendation that can be used to repair S into a net system S' for which it holds that L fits S' perfectly.

6.6. Summary and Discussion

Table I summarizes the results of invoking all the algorithms from Subsections 6.2–6.5 for the input of the net system in Fig. 2, event log L_3 from Subsection 2.3, the standard cost function on label moves for events and activities $\mathcal{A} = \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}, \mathtt{h}, \mathtt{x}\}$, and the standard repair constraint $C := (C_{ins}, C_{skp}, 6)$. For different combinations of algorithms and values of the input *singleton* parameter (columns "Algorithm" and "Singleton", respectively), Table I lists the amount of employed repair resources (column "Res."), time in seconds of executing the respective algorithm (column "Time (s)"), number of discovered approximate minimal optimal repair recommendations (column "# Rec."), number of computations of optimal alignments between traces in the input event log and net system (column "# Align.Comp."), the cost of optimal alignment between the input event log and a net system that can be obtained as a result of an optimal alignment-based repair of the input net system using one of the discovered recommendations (column "Cost"), and a measure of fitness between the event log and the repaired model (column "Fitness").

The reported fitness values are the *relative fitness* values computed as per Definition 7.3.2 in [Adriansyah 2014]. Relative fitness values rely on alignments to quantify conformance. Intuitively, the relative fitness between a given event log and net system is computed as one minus one divided by the number of traces in the event log multiplied by the sum of normalized costs of optimal alignment between each trace in the even log and the net system; each cost is normalized by the highest possible cost of optimal alignment between the corresponding trace and the net system. The highest possible cost of optimal alignment between a trace and net system is determined by the sum of costs of moves on trace defined for each event in the trace and costs of moves on system defined for each transition in the shortest execution of the net system. A relative fitness measure takes a value between 0 and 1. The closer the fitness value is to 1, the better the event log fits the net system. Note that a relative fitness value of 1 denotes perfect fitness. Note also that the relation between the reported costs of optimal alignments and relative fitness values is not monotonic. This is due to the fact that naïve repairs have non-trivial effects on the shortest executions of the repaired systems. Nevertheless, one can clearly identify a trend that lower costs of optimal alignments imply better fitness values between the event log and the repaired net systems. It is worth mentioning that the relative fitness between L_3 and the original (non-repaired) net system shown in Fig. 2 is equal to 0.735.

Table I: Summary of results of searching (approximate) minimal optimal repair recommendations for the input of net system S in Fig. 2, event log L_3 from Section 2.3, the standard cost function on label moves for events and activities that coincide with the universe of labels $\mathcal{A} := \{\mathtt{a}, \mathtt{b}, \mathtt{c}, \mathtt{d}, \mathtt{e}, \mathtt{f}, \mathtt{g}, \mathtt{h}, \mathtt{x}\}$, and the standard repair constraint $C := (C_{ins}, C_{skp}, 6)$; recall from Section 4 that $cost[L_3, S, c] = 120$, where c is the standard cost function on legal moves over S.

Algorithm	Singleton	Res.	Time (s)	# Rec.	# Align.Comp.	Cost	Fitness
Knapsack	true	6	0.009	1	2	40	0.886
Goldratt	true	6	0.021	1	7	45	0.891
Goldratt	false	6	0.03	3	12	38	0.907
Greedy	true	6	0.248	1	88	39	0.879
Greedy	false	6	0.306	1	118	33	0.886
Brute-force (with opt.)	N/A	6	32.594	5	12 406	25	0.931
Brute-force	N/A	6	57.216	5	21 778	25	0.931

Table I confirms one of the expected trends, i.e., algorithms that perform more alignment computations tend to discover 'better' repair recommendations, i.e., repair recommendations that when used to perform optimal alignment-based repairs lead to repaired net systems that fit the given event log better. The greedy approximation scheme that relies on the value of the input Boolean parameter *singleton* of false performs 118 computations of optimal alignments between traces in the input

event log and net system and leads to better discovered repair recommendations as, for example, the Goldratt scheme invoked for the same input value of the *singleton* parameter. At the same time, Table I demonstrates that algorithms that rely on small numbers of alignment computations can produce usable results. For example, the Goldratt approximation schemes perform about ten times fewer alignment computations than the greedy schemes to discover repair recommendations of comparable quality. This phenomenon is studied in detail in the next section.

Note that one needs to increase the amount of available repair resources from six to nine to discover minimal optimal repair recommendations that can be used to completely repair the net system in Fig. 2 w.r.t. event log L_3 from Subsection 2.3. Table II summarizes the results of invoking all the algorithms from Subsections 6.2–6.5 for the input that was used to construct Table I with the only difference in the amount of available repair resources (nine instead of six).

Table II: Summary of results of searching (approximate) minimal optimal repair recommendations for the input of net system S in Fig. 2, event log L_3 from Section 2.3, the standard cost function on label moves for events and activities that coincide with the universe of labels $\mathcal{A} := \{\mathtt{a},\mathtt{b},\mathtt{c},\mathtt{d},\mathtt{e},\mathtt{f},\mathtt{g},\mathtt{h},\mathtt{x}\}$, and the standard repair constraint $C := (C_{ins}, C_{skp}, 9)$; recall from Section 4 that $cost[L_3, S, c] = 120$, where c is the standard cost function on legal moves over S.

Algorithm	Singleton	Res.	Time (s)	# Rec.	# Align.Comp.	Cost	Fitness
Knapsack	true	9	0.005	1	2	15	0.929
Goldratt	true	9	0.024	1	10	20	0.916
Goldratt	false	9	0.052	3	19	15	0.929
Greedy	true	9	0.3	1	118	13	0.947
Greedy	false	9	0.43	1	157	7	0.978
Brute-force (with opt.)	N/A	9	77.374	3	24 337	0	1.0
Brute-force	N/A	9	240.854	3	89 846	0	1.0

The three minimal optimal repair recommendations discovered by the brute-force search are $(\{\mathtt{f},\mathtt{a},\mathtt{x}\},\{\mathtt{f},\mathtt{g},\mathtt{d},\mathtt{e},\mathtt{c},\mathtt{a}\})$, $(\{\mathtt{f},\mathtt{a},\mathtt{x}\},\{\mathtt{f},\mathtt{d},\mathtt{e},\mathtt{c},\mathtt{a},\mathtt{h}\})$, and $(\{\mathtt{f},\mathtt{g},\mathtt{a},\mathtt{x}\},\{\mathtt{d},\mathtt{e},\mathtt{c},\mathtt{a},\mathtt{h}\})$. Note that it is still possible to perform 89 846 computations of optimal alignments between L_3 and the net system in Fig. 2 within reasonable time bounds; cf. the brute-force search row in the table. This is primarily due to the small size of L_3. The next section demonstrates that search schemes that rely on large amounts of alignment computations for event logs of practical interest are mostly intractable.

The brute-force search strategies do not make use of the *singleton* parameter, as usually one needs to check all the feasible repair recommendations to deliver the promised results. One, however, may decide to terminate the brute-force search once it discovers a recommendation that can be used to repair all the deviations between modeled and observed behaviors in the input net system and event log, respectively. This can be implemented by exiting the `foreach` loops of lines 2–7 in Algorithms 1 and 2 once variable *cost* is equal to zero at the end of some iteration. With these modifications, the brute-force strategies, with and without optimization, perform 6 294 and 15 123 alignment computations, respectively, to discover one out of the three existing minimal optimal repair recommendations (invoked for the same input).

Finally, it is worth mentioning that the Knapsack (true), Goldratt (true), Goldratt (false), greedy (true), and greedy (false) search schemes discover recommendations that can be used to repair the given net system completely only if the amount of repair resources is increased to twelve, thirteen, twelve, eleven, and ten, respectively.

7. EVALUATION

All the algorithms presented in this paper have been implemented and are publicly available.[6] Using this implementation we have conducted a series of experiments. This section reports on the results of these experiments. The experiments were performed on a computer with an Intel Core i7-3770

[6]https://svn.win.tue.nl/repos/prom/Packages/SelectivePRepair

CPU@3.40GHz, 8GB of memory, running Windows 7 Enterprise and SUN JVM 1.7. To eliminate load time from the measurements, each test was executed six times, and we recorded average times of the second to sixth executions.

The experiments were conducted on event logs that are publicly available from the Business Processing Intelligence Challenge (BPIC) initiative.[7,8] Net systems used in the experiments were 'mined' from these event logs using the Inductive Miner algorithm [Leemans et al. 2013; Leemans et al. 2015] that ships with ProM 6.3 [van Dongen et al. 2005; Verbeek et al. 2011].

The Inductive Miner algorithm takes as input an event log and a noise threshold and produces a *block-structured* [Polyvyanyy 2012; Polyvyanyy and Bussler 2013] process model, e.g., a net system. A process model is block-structured, or (well-)structured, if every node with multiple outgoing arcs (a split) has a corresponding node with multiple incoming arcs (a join), and vice versa, such that the part of the process model between the split and the join forms a single-entry-single-exit (SESE) component [Vanhatalo et al. 2009; Polyvyanyy et al. 2011]; otherwise the model is *unstructured* [Polyvyanyy 2012]. The noise threshold parameter is designed to allow filtering of infrequent traces and events in an input event log prior to construction of the resulting model [Leemans et al. 2013; Leemans et al. 2015]. The threshold parameter takes a value between zero and one inclusive. The noise threshold value of zero indicates that no filtering is to be applied and that the event log will fit perfectly the constructed model. The closer the value of the input noise threshold parameter is to the value of one, the more deviations will be present between traces in the input event log and executions of the model produced by the Inductive Miner algorithm.

Five net systems were mined from each of the studied event logs using the Inductive Miner algorithm. The net systems were produced using different values of the noise threshold parameter; the employed noise threshold values are 0.2, 0.4, 0.6, 0.8, and 1.0. The obtained net systems were then repaired using repair recommendations discovered by the algorithms proposed in Section 6. The repairs were conducted following the naïve repair method, cf. Definition 4.6, based on the optimal alignments between traces and executions of the respective event logs and net systems, which, according to Theorem 5.3, leads to optimal alignment-based repairs, cf. Definition 4.3.

Table III presents results of the experiments conducted on the BPIC 2012 event log taken from a Dutch Financial Institute. The top most sub-table contains information on costs of optimal alignments between the event log and repaired net systems. The sub-table is divided into five parts, cf. columns 0.2, 0.4, 0.6, 0.8, and 1.0, where each part presents information on costs of optimal alignment between the input event log and the repaired version of the input net system which is mined from the event log using the respective noise threshold value. Each mined net system was repaired using repair recommendations discovered via Algorithms 4, 5, and 6. The algorithms were invoked for the inputs of standard cost functions on label moves and standard repair constraints. Algorithm 4 was invoked for the value of the input *singleton* parameter of true, cf. column "K(T)" in the sub-table, whereas Algorithms 5 and 6 were invoked for the values of the *singleton* parameter of true and false. Consequently, columns "G(T)" and "G(F)" contain results of invoking Algorithm 5 for the values of the *singleton* parameter of true and false, respectively. Similarly, columns "GR(T)" and "GR(F)" contain the results of invoking Algorithm 6 for the true and false input *singleton* values, respectively. Each algorithm was invoked for 16 different values of available repair resources, i.e., from 0 to 15. The use of the repair resource value of zero corresponds to the situation of no repair and, therefore, the corresponding values represent the costs of optimal alignments between the BPIC 2012 event log and the original (non-repaired) net system, refer to row 0 in the sub-table. The three other sub-tables are organized in a similar way as the top most one. However, instead of the costs of optimal alignments between the event log and repaired net systems they present numbers of optimal alignment computations (between traces in the event log and the net systems) performed to discover repair recommendations, run times of the repair recommendation search algorithms (in seconds), and the numbers of discovered repair recommendations; refer to the headings of the sub-tables.

[7] http://dx.doi.org/10.4121/uuid:3926db30-f712-4394-aebc-75976070e91f
[8] http://dx.doi.org/10.4121/uuid:500573e6-accc-4b0c-9576-aa5468b10cee

Table III: Experimental results for the BPIC 2012 event log.

Cost of Optimal Alignment

noise		0.2					0.4					0.6					0.8					1.0				
algorithm		K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)
repair resources	0	46983	46983	46983	46983	46983	129189	129189	129189	129189	129189	148787	148787	148787	148787	148787	136462	136462	136462	136462	136462	172844	172844	172844	172844	172844
	1	38016	38016	38016	38016	38016	106351	106351	106351	106351	106351	126936	126936	126936	126936	126936	121595	121595	121595	121595	121595	149868	149868	149868	149868	149868
	2	33162	28932	28932	28932	28932	83719	83719	83719	83719	83719	104301	104301	104301	104301	104301	100204	100204	100204	100204	100204	127462	127462	127462	127462	127462
	3	29240	24078	24078	23909	23909	66640	66640	66640	66640	66640	88026	88026	88026	88026	88026	91766	91766	91766	91766	91766	110853	110853	110853	110853	110853
	4	20026	20026	20026	19055	19055	60816	60816	60816	60816	60816	66313	66313	66313	66313	66313	84687	82908	82908	82908	82908	94699	94699	94699	94699	94699
	5	16572	16572	16572	15601	15601	55793	55793	55793	55233	55233	56939	56939	56939	56939	56939	76850	76023	76023	76023	76023	83292	83292	83292	83292	83292
	6	13191	13191	13191	12220	12220	50210	50210	50210	50210	50210	46747	46747	46747	46747	46747	68219	68219	68219	68219	68219	71892	71892	71892	71892	71892
	7	10208	10208	10208	9127	9127	46282	46282	46282	46282	46282	38836	38836	38836	38836	38836	62809	63196	63196	62809	62809	63997	63997	63997	63997	63997
	8	8568	7108	7108	7331	7331	42151	42151	42151	42151	40387	33813	33813	33813	33813	33813	57786	57786	57786	57786	57786	56106	56106	56106	56106	56106
	9	7737	5339	5339	5339	5339	38697	38697	38697	36256	36256	30098	30098	30098	30098	30098	54610	53695	53695	53695	53695	50993	50993	50993	50993	50993
	10	7058	3699	3699	3699	3699	32802	32802	32802	32802	32802	26712	26644	26644	26644	26644	51372	50023	50023	50023	50023	45970	45970	45970	45970	45970
	11	6222	2868	2868	2868	2868	29421	29421	29421	29421	29421	23258	23258	23258	23258	23258	47993	46368	46368	46368	46368	42315	42315	42315	41764	41764
	12	5074	2066	2066	2066	2066	27207	26514	26514	26165	26165	20351	19935	19935	19935	19935	44933	42719	42719	42719	42719	38861	38861	38861	38109	38109
	13	3304	1670	1670	1670	1670	24392	23699	23699	23258	23258	17028	17028	17028	17009	17009	41163	39265	39265	39265	39265	35473	35473	35473	34655	34655
	14	1684	1273	1273	1273	1273	21485	20892	20892	20443	20443	14221	14221	14221	14102	14102	38523	36012	36012	36012	36012	29576	29576	29576	31848	31848
	15	1676	876	876	876	876	18678	18509	18509	17636	17636	11570	11295	11295	11295	11295	34851	32775	32775	32775	32775	26769	24805	24805	29327	29327

Number of Optimal Alignment Computations

noise		0.2					0.4					0.6					0.8					1.0				
algorithm		K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)
repair resources	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1	2	2	2	70	70	2	2	2	65	65	2	2	2	72	72	2	2	2	69	69	2	2	2	64	64
	2	2	3	3	138	138	2	3	3	128	128	2	3	3	142	142	2	3	3	136	136	2	3	3	126	126
	3	2	4	4	205	205	2	4	4	190	190	2	4	4	211	211	2	4	4	202	202	2	4	4	187	187
	4	2	5	5	271	271	2	5	5	251	373	2	5	5	279	279	2	5	5	267	267	2	5	5	247	247
	5	2	6	6	336	336	2	6	6	311	553	2	6	6	346	346	2	6	6	331	331	2	6	6	306	306
	6	2	7	7	400	400	2	7	7	370	730	2	7	7	412	412	2	7	7	394	394	2	7	7	364	364
	7	2	8	8	463	463	2	8	9	428	904	2	8	8	477	477	2	8	8	456	456	2	8	8	421	421
	8	2	9	9	525	525	2	9	10	485	1246	2	9	9	541	669	2	9	9	517	517	2	9	9	477	477
	9	2	10	10	586	586	2	10	11	541	1414	2	10	10	604	858	2	10	10	577	577	2	10	10	532	532
	10	2	11	11	646	646	2	11	12	596	1579	2	11	11	666	1044	2	11	11	636	636	2	11	11	586	586
	11	2	12	12	705	705	2	12	13	650	1741	2	12	12	727	1227	2	12	12	694	694	2	12	12	639	639
	12	2	13	13	763	763	2	13	14	703	1900	2	13	13	787	1407	2	13	13	751	751	2	13	13	691	691
	13	2	14	16	820	820	2	14	15	755	2056	2	14	14	846	1584	2	14	14	807	807	2	14	14	742	742
	14	2	15	17	876	988	2	15	16	806	2209	2	15	15	904	1758	2	15	15	862	862	2	15	15	792	792
	15	2	16	18	1043	1043	2	16	17	856	2359	2	16	16	961	1929	2	16	18	916	916	2	16	16	841	841

Computation Time (sec)

noise		0.2					0.4					0.6					0.8					1.0				
algorithm		K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)
repair resources	0	70.1	70.0	70.2	69.9	70.2	29.4	29.3	29.5	29.7	29.5	45.8	45.8	46.4	46.2	45.9	20.8	20.9	20.9	21.5	20.8	8.4	8.4	8.3	8.4	8.3
	1	139.0	138.1	139.0	4883	4891	58.8	59.1	58.8	1922	1916	91.8	91.6	92.0	3278	3284	42.0	41.7	41.7	1448	1450	16.4	16.4	16.5	539.1	536.8
	2	137.3	203.0	203.1	9561	9500	57.9	88.1	87.5	3772	3762	90.9	136.5	136.3	6481	6509	40.3	61.4	61.9	2849	2845	15.6	23.7	23.8	1038	1035
	3	132.2	265.5	266.9	13872	13848	56.3	114.7	113.7	5535	5522	90.6	181.4	180.5	9544	9506	40.4	80.8	81.0	4134	4109	15.8	31.2	31.3	1481	1479
	4	125.9	323.4	321.1	17321	17343	54.9	140.2	139.1	7163	10358	84.6	219.5	219.4	12501	12560	40.1	101.5	100.5	5393	5417	15.6	38.4	38.5	1927	1936
	5	126.5	379.3	380.3	20742	20708	54.2	164.2	164.0	8621	15007	85.0	259.2	260.7	15219	15080	39.0	118.5	119.7	6679	6691	15.5	45.5	45.5	2360	2355
	6	126.4	434.0	435.1	24108	24079	50.6	184.8	185.2	9908	18709	82.9	295.5	295.4	17773	17719	38.8	138.2	137.5	7871	7870	14.8	52.0	51.8	2771	2765
	7	122.3	486.8	485.1	27355	27531	50.7	206.8	227.5	11067	22342	85.2	335.6	334.8	20196	20199	38.4	155.2	154.5	9062	8908	14.1	57.9	57.7	3126	3126
	8	118.6	532.8	533.6	30368	30325	51.0	227.1	249.6	12304	29527	81.9	368.9	369.7	22621	27369	37.7	171.8	171.8	10051	10122	13.2	62.4	62.3	3453	3452
	9	115.7	573.8	576.8	33333	33083	51.0	249.0	271.8	13502	33104	80.8	404.1	403.1	24798	34111	38.2	187.0	186.1	11038	11091	13.0	67.0	67.4	3722	3716
	10	115.8	611.1	609.6	35525	35607	51.2	273.3	293.8	14879	36606	81.2	438.3	437.8	27013	40688	37.8	205.2	203.2	11947	11976	12.6	71.2	71.2	3969	3970
	11	115.0	647.7	641.0	37741	37633	50.4	290.9	316.8	15852	40212	79.5	471.3	468.7	28936	46944	38.1	221.0	219.8	12976	12937	12.5	75.5	75.3	4187	4184
	12	113.5	673.4	669.4	39615	39690	49.7	311.5	332.7	17068	43446	77.9	505.5	502.5	31052	52910	38.0	235.5	234.4	13788	13868	12.1	79.0	78.8	4431	4415
	13	106.7	710.2	771.2	41510	41414	47.8	331.5	352.8	18067	47300	74.8	528.7	531.6	32878	58604	38.3	249.6	251.0	14717	14770	12.1	82.4	82.5	4590	4587
	14	101.4	734.4	805.2	43117	47069	46.9	347.5	368.9	19242	49908	72.4	556.2	557.4	34491	63121	38.0	268.8	267.3	15645	15740	12.0	86.0	86.1	4764	4767
	15	101.6	768.4	834.2	44580	48727	44.3	366.0	388.1	20379	53314	69.9	580.7	580.0	35988	66956	37.8	280.9	314.4	16339	16485	12.0	89.6	89.5	4939	4937

Number of Repair Recommendations

noise		0.2					0.4					0.6					0.8					1.0				
algorithm		K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)
repair resources	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	3	1	1	1	1	1	1	1	1	1	3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	4	1	1	1	1	1	1	1	1	1	3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	5	1	1	1	1	1	1	1	1	1	3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	6	1	1	1	1	1	1	1	1	1	3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	7	1	1	1	1	1	1	1	2	1	6	1	1	1	1	3	1	1	1	1	1	1	1	1	1	1
	8	1	1	1	1	1	1	1	1	1	3	1	1	1	1	3	1	1	1	1	1	1	1	1	1	1
	9	1	1	1	1	1	1	1	1	1	3	1	1	1	1	3	1	1	1	1	1	1	1	1	1	1
	10	1	1	1	1	1	1	1	1	1	3	1	1	1	1	3	1	1	1	1	1	1	1	1	1	1
	11	1	1	1	1	1	1	1	1	1	3	1	1	1	1	3	1	1	1	1	1	1	1	1	1	1
	12	1	1	1	1	1	1	1	1	1	3	1	1	1	1	3	1	1	1	1	1	1	1	1	1	1
	13	1	1	3	1	3	1	1	1	1	3	1	1	1	1	3	1	1	1	1	1	1	1	1	1	1
	14	1	1	1	1	1	1	1	1	1	3	1	1	1	1	3	1	1	1	1	1	1	1	1	1	1
	15	1	1	1	1	1	1	1	1	1	3	1	1	1	1	3	1	1	3	1	3	1	1	1	1	1

Table IV: Experimental results for the BPIC 2013 event log.

Cost of Optimal Alignment

noise		0.2					0.4					0.6					0.8					1.0				
algorithm		K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)
repair resources	0	23552	23552	23552	23552	23552	42568	42568	42568	42568	42568	17974	17974	17974	17974	17974	16186	16186	16186	16186	16186	64215	64215	64215	64215	64215
	1	16519	16519	16519	16519	16519	19314	19314	19314	19314	19314	10421	10421	10421	10421	10421	8633	8633	8633	8633	8633	39225	39225	39225	39225	39225
	2	12837	12837	12837	12837	12837	12490	12490	12490	12490	12490	6245	6245	6245	6245	6245	5624	5624	5624	2868	2868	28124	28124	28124	28124	28124
	3	9794	9794	9794	9794	9794	9214	9214	9214	9214	9214	3911	3911	3911	3338	3338	3589	3589	3589	833	833	20571	20571	20571	20571	20571
	4	7760	7760	7760	7760	7760	7179	6108	6108	6108	6108	1876	1876	1876	1303	1303	1730	1730	1730	422	422	14778	15630	15630	14778	14778
	5	5780	5780	5780	5780	5780	4073	4073	4073	4073	4073	1384	1384	1384	811	811	1149	1149	1149	111	111	9350	10333	10333	9350	9350
	6	3996	3996	3996	3996	3996	2577	2577	2577	2577	2577	924	924	924	400	400	738	738	738	12	12	5574	6166	6166	5574	5574
	7	2463	2463	2463	2463	2463	2471	1378	1378	1378	1378	515	515	515	113	113	427	427	427	0	0	2425	2945	2945	2425	2425
	8	1071	1071	1071	1071	1071	1187	902	902	902	902	227	227	227	12	12	225	225	225			941	1412	1412	941	941
	9	579	579	579	579	579	1187	497	497	497	497	114	114	114	0	0	112	112	112			892	894	894	448	448
	10	266	266	266	266	266	711	187	187	187	187	13	13	13			13	13	13			448	401	401	138	138
	11	124	124	124	124	124	400	86	86	86	86	13	0	0			12	0	0			448	127	127	40	40
	12	23	23	23	23	23	182	5	5	5	5	0					0					207	57	57	22	22
	13	5	5	5	5	5	105	0	0	0	0											138	39	39	5	5
	14	0	0	0	0	0	4															57	22	22	0	0
	15						4															55	5	5		

Number of Optimal Alignment Computations

noise		0.2					0.4					0.6					0.8					1.0				
algorithm		K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)
repair resources	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1	2	2	2	25	25	2	2	2	26	26	2	2	2	26	26	2	2	2	26	26	2	2	2	27	27
	2	2	3	3	48	48	2	3	3	50	50	2	3	3	50	50	2	3	3	50	50	2	3	3	52	52
	3	2	4	4	70	70	2	4	4	73	73	2	4	4	73	73	2	4	4	73	73	2	4	4	76	76
	4	2	5	5	91	91	2	5	5	95	95	2	5	5	95	95	2	5	5	95	95	2	5	5	99	99
	5	2	6	6	111	111	2	6	6	116	116	2	6	6	116	116	2	6	6	116	137	2	6	6	121	121
	6	2	7	7	130	130	2	7	7	136	136	2	7	7	136	136	2	7	7	136	177	2	7	7	142	142
	7	2	8	8	148	148	2	8	8	155	155	2	8	8	155	155	2	8	8	155	215	2	8	8	162	162
	8	2	9	9	165	165	2	9	9	173	173	2	9	9	173	173	2	9	9	155		2	9	9	181	181
	9	2	10	10	181	181	2	10	10	190	190	2	10	10	190	190	2	10	10			2	10	10	199	199
	10	2	11	11	196	196	2	11	11	206	206	2	11	11			2	11	11			2	11	11	216	216
	11	2	12	12	210	210	2	12	12	221	221	2	12	12			2	12	12			2	12	12	232	232
	12	2	13	13	223	223	2	13	13	235	235	2					2					2	13	13	247	247
	13	2	14	14	235	235	2	14	14	248	248											2	14	15	261	275
	14	2	15	15	246	246	2															2	15	16	274	301
	15						2															2	16	17		

Computation Time (sec)

noise		0.2					0.4					0.6					0.8					1.0				
algorithm		K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)
repair resources	0	7.3	7.3	7.2	7.3	7.3	2.9	2.9	2.9	2.9	2.9	2.9	2.9	2.8	2.9	2.9	1.2	1.2	1.3	1.2	1.2	10.7	10.3	10.4	10.5	10.3
	1	14.6	14.5	14.5	181.1	181.2	6.0	5.8	5.8	76.5	76.5	5.2	5.3	5.2	74.7	74.5	2.3	2.2	2.2	31.4	31.4	21.8	21.5	21.6	277.6	277.7
	2	13.9	21.1	21.1	346.3	346.1	5.9	8.8	8.8	147.1	147.5	4.5	6.9	7.0	130.2	130.1	2.2	3.3	3.4	56.4	56.4	21.3	32.5	32.5	558.3	557.4
	3	14.0	27.9	27.8	491.7	493.3	5.6	11.5	11.5	215.5	215.1	4.5	8.4	8.4	169.7	169.5	2.2	4.2	4.3	81.7	81.5	20.5	42.6	42.5	818.2	821.6
	4	14.1	34.6	34.6	633.4	633.9	5.6	13.6	13.6	275.9	273.3	4.4	10.0	10.1	209.4	210.0	2.1	5.2	5.2	105.3	105.4	19.1	51.4	51.4	1045	1046
	5	14.0	41.3	41.6	769.0	767.0	5.0	15.7	15.8	318.9	318.7	4.4	11.4	11.5	248.1	248.4	2.1	6.1	6.4	129.0	152.2	17.4	58.6	58.5	1237	1236
	6	13.9	47.9	47.9	896.5	897.4	4.7	17.5	17.7	361.6	362.1	4.2	12.8	12.9	281.8	283.3	2.1	7.0	7.1	151.2	195.2	15.6	64.1	63.9	1389	1386
	7	13.2	53.8	54.0	1016	1013	5.0	19.3	19.3	397.1	396.6	4.2	14.4	14.1	314.7	313.9	2.1	7.9	7.9	171.0	237.2	13.7	67.2	67.2	1495	1494
	8	13.2	59.9	59.9	1119	1117	5.0	20.9	21.0	429.2	429.1	4.1	15.5	15.5	341.7	343.4	2.2	8.9	8.8			13.0	69.3	69.1	1594	1564
	9	12.9	67.3	66.9	1214	1221	5.6	22.5	22.5	460.3	458.4	4.2	16.8	16.8	369.1	369.0	2.1	9.8	9.8			13.3	72.8	72.9	1613	1635
	10	12.9	71.0	70.8	1330	1307	5.6	24.2	24.0	483.6	484.1	4.1	18.0	18.0			2.1	10.7	10.9			12.8	72.5	72.6	1658	1655
	11	13.2	77.9	78.2	1383	1397	5.6	25.4	25.7	507.0	508.2	4.0	19.4	19.3			2.2	11.7	11.6			13.0	74.0	74.2	1699	1695
	12	12.7	81.7	81.8	1449	1454	5.6	26.3	26.1	527.7	528.0	3.5					2.2					13.0	75.3	75.6	1731	1734
	13	12.9	87.3	87.5	1519	1520	5.6	26.8	27.0	536.0	535.7											12.8	76.8	77.6	1748	1780
	14	12.9	92.9	93.1	1587	1584	5.8															11.6	78.0	78.7	1754	1826
	15						5.6															11.6	78.9	80.1		

Number of Repair Recommendations

noise		0.2					0.4					0.6					0.8					1.0				
algorithm		K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)	K(T)	G(T)	G(F)	GR(T)	GR(F)
repair resources	0	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	2	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	3	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	4	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	1	1	1	1	1
	5	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	1	1	1	1	1
	6	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	1	1	1	1	1
	7	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	4	1	1	1	1	1
	8	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1			1	1	1	1	1
	9	1	1	1	1	1	1	1	1	1	1	1	1	1	1	2	1	1	1			1	1	1	1	1
	10	1	1	1	1	1	1	1	1	1	1	1	1	1			1	1	1			1	1	1	1	1
	11	1	1	1	1	1	1	1	1	1	1	1	1	1			1	1	1			1	1	1	1	1
	12	1	1	1	1	1	1	1	1	1	1	1					1					1	1	1	1	2
	13	1	1	1	1	1	1	1	1	1	2											1	1	1	1	2
	14	1	1	1	1	1	1															1	1	1	1	2
	15						1															1	1	1		

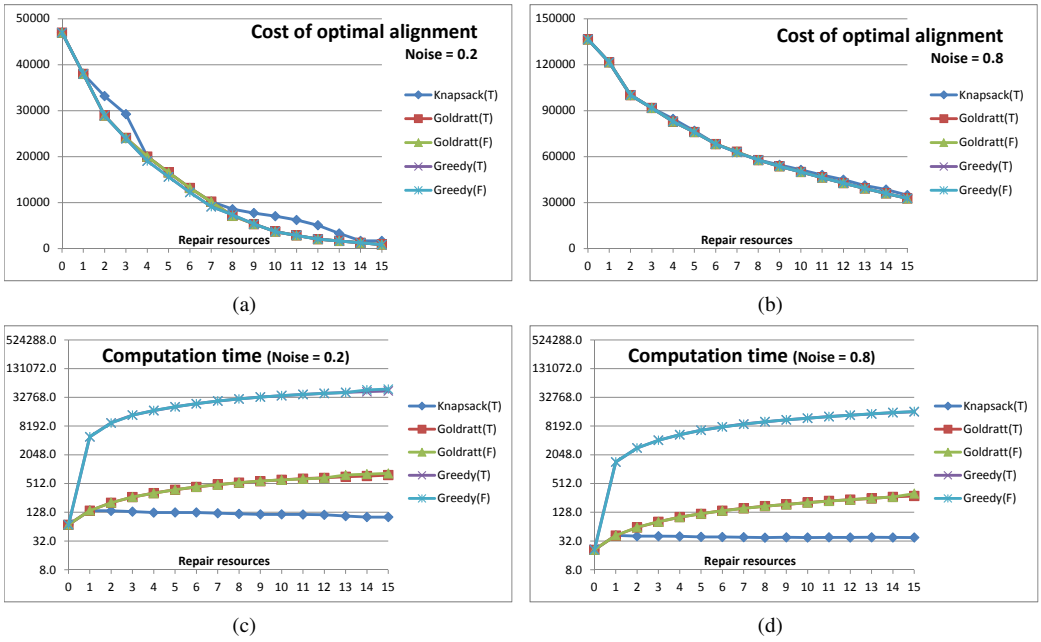


Fig. 12: (a) and (b) show costs of optimal alignment between the BPIC 2012 event log and the repaired versions of net systems constructed from the BPIC 2012 event log using the Inductive Miner algorithm for the noise threshold parameters of (a) 0.2 and (b) 0.8, whereas (c) and (d) show times in seconds spent to discover approximate minimal optimal repair recommendations that were used to perform the repairs in (a) and (b) for the noise threshold parameters of (c) 0.2 and (d) 0.8.

Table IV is organized in the same way as Table III and presents results of the experiments conducted on the BPIC 2013 event log taken from Volvo IT Belgium. For this setup, it was often sufficient to use less than 15 repair resources to discover feasible repair recommendations that can be used to obtain repaired net systems that fit the event log perfectly, which explains empty cells in the table.

Lessons learned from the above reported experiments are:

(Lesson 1) Search techniques that perform more computations of optimal alignments between traces in the input event log and net system (as per different cost functions) tend to discover 'better' repair recommendations, i.e., repair recommendations that can be used to obtain repaired models that fit the event log better.

(Lesson 2) To obtain a small improvement in the quality of discovered repair recommendations, a search technique needs to perform a large number of additional computations of optimal alignments between traces in the input event log and the net system (as per different cost functions).

(Lesson 3) If available repair resources are scarce, all the proposed techniques for searching approximate minimal optimal repair recommendation tend to discover identical repair recommendations.

(Lesson 4) If available repair resources are scarce, approximate minimal optimal repair recommendations discovered using the proposed techniques tend to coincide with the actual ones.

Lessons 1 to 3 can be easily reconfirmed by studying the results reported in Tables III and IV and the plots in Figs. 12 and 13. Figs. 12(a) and 12(b) show the costs of optimal alignment between the BPIC 2012 event log and repaired versions of net systems constructed from the BPIC 2012 event log using the Inductive Miner algorithm for the noise threshold parameters 0.2 and 0.8, respectively, plotted against the amount of repair resources used to discover the corresponding repair recommendations. Figs. 13(a) and 13(b) show similar plots but for the BPIC 2013 event log.

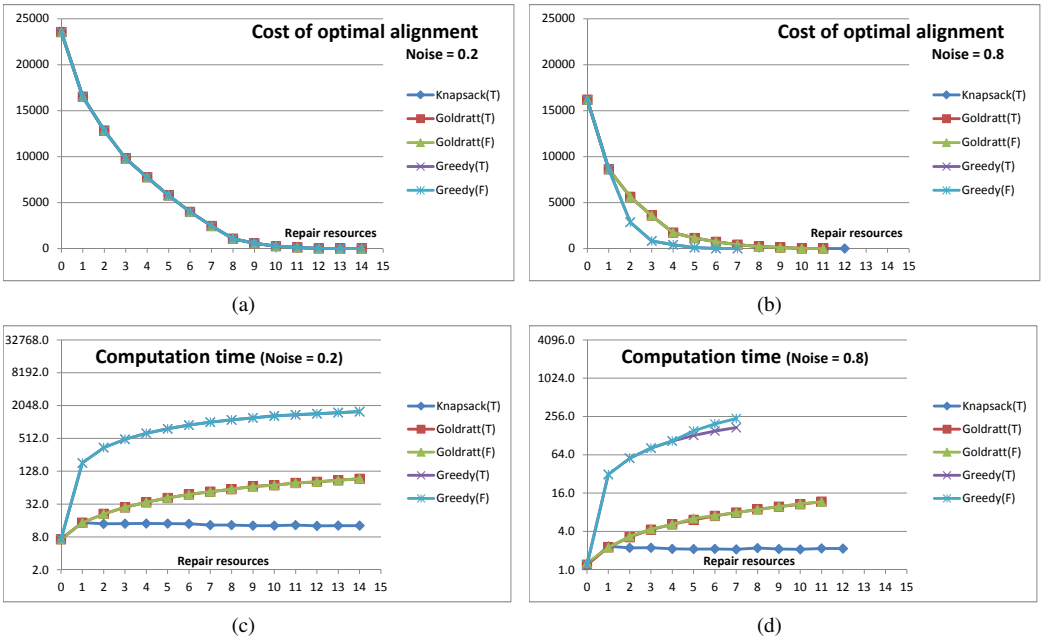


Fig. 13: (a) and (b) show costs of optimal alignment between the BPIC 2013 event log and the repaired versions of net systems constructed from the BPIC 2013 event log using the Inductive Miner algorithm for the noise threshold parameters of (a) 0.2 and (b) 0.8, whereas (c) and (d) show times in seconds spent to discover approximate minimal optimal repair recommendations that were used to perform the repairs in (a) and (b) for the noise threshold parameters of (c) 0.2 and (d) 0.8.

Using these figures, the reader can easily verify that all the five plotted curves, which represent data collected for five different search techniques, follow a similar trajectory; note slight deviations from the common trend in the results of Algorithm 4 in Fig. 12(a) and Algorithm 6 in Fig. 13(b). Indeed, those algorithms that perform more computations of optimal alignments tend to discover better repair recommendations (Lesson 1) and, when small amounts of repair resources are used, all the algorithms tend to discover similar results (Lesson 3). However, as suggested in sub-figures (c) and (d) of Figs. 12 and 13, in order to obtain small improvements in the quality of discovered repair recommendations, considerable computation efforts are required (Lesson 2). These figures report logarithm (base 2) of the times (in seconds) spent by different search techniques to discover the corresponding repair recommendations; plotted against the amount of given repair resources. For example, for the experiment reported in Figs. 12(b) and 12(d), all the proposed search techniques discovered repair recommendations of the same quality for the input repair resources parameter from 1 to 3. While Algorithm 4 invoked for the value of the input *singleton* parameter of true spent a bit over 2 minutes to discover all the three repair recommendations, Algorithm 6 spent 2 hours 20 minutes and 31 seconds to discover the same repair recommendations on the same inputs. For the same experiment, when the amount of repair resources was increased to 4, Algorithm 6 for the value of the input *singleton* parameter of true has discovered a repair recommendation that can be used to repair the input net system better than a repair recommendation discovered using Algorithm 4 for the same input. However, the costs of optimal alignment obtained between the input event log and each of the two repaired net systems using the discovered repair recommendations differ by only approximately 2%, while Algorithm 6 took 134 times longer than Algorithm 4 to complete.

Lesson 4 stems from the fact that for every repair recommendation search exercise conducted in this evaluation (up to an amount of available repair resources equal or less than 3), we have observed

that at least one of the proposed techniques for searching approximate repair recommendations was capable of discovering at least one minimal optimal repair recommendation. This observation was confirmed by comparing discovered approximate recommendations with those constructed using Algorithm 1. In addition, we ran Algorithm 2 on the inputs of the net system discovered using the Inductive Miner algorithm from the BPIC2013 event log for the noise threshold parameter 1.0, the BPIC 2013 event log, the standard cost function on label moves, and the five different standard repair constraints for the amounts of available repair resources ranging from 1 to 5. The discovered repair recommendations coincide with the best repair recommendations discovered on the same inputs by the search techniques proposed in Section 6. For example, for the amounts of repair resources, 1, 2, 3, 4, and 5, the repair recommendations discovered using Algorithm 2 can be used to repair the net system into systems that have the costs of optimal alignment with the BPIC 2013 event log of 39 225, 28 124, 20 571, 14 778, and 9 350, respectively. However, the times spent on executing the algorithm range from 297 seconds to 8 days 8 hours 22 minutes and 32 seconds, as the algorithm performed 65 785 computations of optimal alignments between the input event log and net system as per different cost functions on legal moves for the input value of available repair resources of 5.

8. RELATED WORK

The impact-driven process model repair approach presented in this paper relates to the process mining research studies conducted in the areas of process discovery, conformance checking, and process model repair, as well as to the service-oriented computing works on self-healing services.

Process discovery techniques are concerned with generating a 'good' process model that depicts observed behavior captured in an event log. Many discovery approaches exist in the literature [van der Aalst 2011]: the Alpha algorithm that makes use of ordering relations of activities [van der Aalst et al. 2004], the Heuristics Miner that discovers main behavior from noisy logs [Weijters et al. 2006], the Fuzzy Miner that caters for less structured processes and supports different levels of abstractions [Günther and van der Aalst 2007], the Genetic Miner that balances different quality criteria [de Medeiros et al. 2007], the Inductive Miner that discovers sound, fit, and block-structured models [Leemans et al. 2013; Leemans et al. 2015], the discovery technique over precedence constraints [Greco et al. 2015], etc. These algorithms vary in terms of the types of logs they can effectively handle and the quality of discovered models. There are four main quality dimensions (fitness, precision, simplicity and generalization) that can provide insights into how good a process model represents reality [Buijs et al. 2012; van der Aalst 2011]. Often, trade-offs between the different quality dimensions also need to be taken into account when assessing quality of a discovered process model. Most discovery algorithms start from scratch. i.e., the algorithms only consider event data and ignore existing models. Thus, it is possible to obtain models from such algorithms that are very different to existing process models, which could potentially limit the usability of discovered models. However, there have been research studies to incorporate the similarity of process models as an additional quality dimension. For example, Buijs et al. take into account the similarity of a discovered model to a given process model as the fifth quality dimension [Buijs et al. 2013]. These recent efforts highlight how process knowledge extracted from event logs can be supplemented with knowledge captured in reference process models to produce high-quality and representative models and, thus, further motivate our work in the area of process model repair. In [Buijs et al. 2013], the authors show that there may exist several good discovered models that adequately represent behavior recorded in a given event log. By constructing a Pareto front of discovered models, one can identify collections of non-dominating (as per some quality dimensions) process models. Similarly, one can construct a Pareto front of repaired models obtained using different amounts of repair resources and different repair techniques to prioritize them based on different quality criteria.

Conformance checking techniques are concerned with the detection and diagnosis of the differences between modeled and observed behaviors of a process [Rozinat and van der Aalst 2008; Adriansyah et al. 2011; van der Aalst 2011]. These techniques address how to determine the degree of conformance between modeled and observed behaviors by comparing 'allowable moves' by a model against 'observed moves' in an event log [Adriansyah 2014]. Adriansyah et al. developed a technique

to align activities in models and transitions executed in event logs and showed how to use these alignments to replay an event log over a process model [Adriansyah et al. 2011]. Deviations between moves on model and moves on event log are detected and the severity of deviations is quantified using likelihood cost functions for skipped activities (activities that should be carried out according to the model, but do not occur in the event log) and inserted activities (activities that should not be carried out according to the model, but occur in the event log). Our technique relies on alignments such as those proposed in [Adriansyah 2014] to solve the conformance checking problem. The main quality dimension of interest for a conformance checking algorithm is replay fitness. In [Munoz-Gama et al. 2013; Munoz-Gama et al. 2013], hierarchical conformance checking approaches that decompose processes using the Refined Process Structure Trees (RPSTs) [Vanhatalo et al. 2009; Polyvyanyy et al. 2011] and localize conformance problems while taking into account fitness and precision measures are presented. In [Molka et al. 2014], the authors propose a conformance checking approach between a process modeled using Business Process Model and Notation (BPMN) and an event log with conformance measures in terms of precision (the fraction of model behavior supported by the event log) and recall (the fraction of event log behavior supported by the model).

The conformance checking approach by Adriansyah [Adriansyah 2014] focuses on the control-flow behavior conformance while abstracting from other causes for potential variations between modeled and observed behaviors. The works in [de Leoni and van der Aalst 2013b; Mannhardt et al. 2015] describe multi-perspective conformance checking approaches which make use of a Petri net enriched with data variables to capture the different process perspectives. De Leoni & van der Aalst propose to align models and event logs so that the control-flow alignment is considered first and other process perspectives (i.e., data, time and resources) are subsequently taken into account within the objective function of an ILP problem [de Leoni and van der Aalst 2013a]. The approach presented in [Mannhardt et al. 2015] balances different process perspectives in a customizable manner to identify explanations for deviations. One can extend our work to consider using alignments based on multiple process dimensions when searching optimal repair recommendations.

Techniques to improve the quality of a process model have been researched. The works in [Li et al. 2009; Li et al. 2010; Li et al. 2011] develop an approach to transform a process model by taking into account the similarity of process model variants based on edit distance. In order for a process model to better reflect reality, knowledge obtained through conformance checking analysis could be used for process model repairs. In [Gaaloul et al. 2009], the authors propose to discover a transactional workflow model from an event log and to repair the initially designed workflow based on the discrepancies between the two models. In [Adriansyah 2014], the alignment-based conformance checking approach was extended to detect high-level deviation patterns such as the activity replacement pattern, the activity reordering pattern and the activity repetition pattern. By appending the original model with such patterns, a better alignment can be observed between a given event log and the repaired model. The work by Fahland et al. [Fahland and van der Aalst 2015] describes an approach to repair a process model such that it can replay (most of) the event log, i.e., to increase the fitness of the model, while staying structurally close to the original model. This is achieved by first computing alignments to identify non-fitting traces, then grouping traces from the event log according to the identified misalignments, then discovering a sub-process for each group of traces, and inserting such a sub-process into the original model to enable the replay of the corresponding non-fitting traces. This is repeated until a repaired process model is obtained that can replay the observed behavior perfectly. The resulting repaired model contains new loops and sub-processes to minimize misalignments between the event log and the original model. Our approach has the same aim as the repair approach presented in [Fahland and van der Aalst 2015], i.e., to minimize misalignments, but instead of fixing every single misalignment, we identify and prioritize those repairs that have the highest impact on model fitness for a given repair constraint. Hence, our approach can be used to suggest incremental process repair recommendations by making adjustments to the cost of repairs and/or the number of available resources for repair.

Service-oriented computing (SOC) [Papazoglou 2003] proposes to compose new value-added services from the existing component services. A service composition may yield unreliable during

its execution as certain component services may happen to be unavailable and/or faulty. To tackle this problem, the notion of a self-healing service was proposed. The idea of a self-healing service is to supply a service composition with a mechanism to allow its automatic repair at run time. A self-healing service can repair itself by retrying, compensating, and/or substituting certain faulty component service invocations. The reader can refer to [Eder et al. 2009; Friedrich et al. 2010; Islam et al. 2010; Gaaloul et al. 2010] for approaches on managing self-healing services. In contrast to these approaches, we repair service compositions (which are usually captured as process models) at design time to allow the repaired models to better reflect the so far observed executions.

9. CONCLUSION

Process model repair guided by event data can be positioned in-between process discovery (discovering a control-flow model based on just event data) and conformance checking (taking a predefined model as the "norm"). To repair a model there is a trade-off between (1) staying close to the modeled behavior and (2) better fitting the observed behavior. This paper focuses on *impact-driven process model repair* where costs are assigned to model changes and repair resources are limited. Within the space of possible repairs, we seek repaired models that maximize fitness (minimize mismatches between repaired model and log) constrained by the repair resources, i.e., different repairs may have different costs. This way model repair becomes an optimization problem that can be controlled through easy-to-understand parameters. The trade-offs considered in this paper were not investigated earlier and, overall, surprising few process mining papers considered model repair.

In this paper, various algorithms have been proposed and implemented. A brute-force algorithm, and a slightly more efficient variant of it, ensure finding optimal repair recommendations, but may be too time consuming for larger models and event logs. Hence, approximative algorithms have been developed. These do not guarantee finding optimal repair recommendations, but require fewer alignment computations. Experiments show that the approximations are quite good, i.e., often many additional alignment computations are needed to achieve further small improvements in fitness. Using the approximations, it is possible to repair real-life models in a satisfactory manner.

Future work aims at extending and improving impact-driven process model repair in various ways.

First of all, we would like to conduct more case studies to better understand the requirements of model repair. For example, how to elicit repair constraints?

Second, we are interested in providing guarantees for our approximative algorithms. Bounds can be set on the difference between the alignment costs of the unknown optimal repair recommendation and the result of the approximative algorithm. This way end-users can better judge the quality of the repair and decide whether an exhaustive search is needed. We hope that this will give us new ideas on faster algorithms for discovering optimal repair recommendations.

Third, repairs are considered at the level of activity labels rather than model structures. The naïve repair that relies on self-loop and/or skip injections (Definition 4.6) can be improved in various ways. Loops are introduced even if activities happen only once. Skips are always atomic. Hence, the naïve repair approach leaves ample room for improvement and model refactoring. However, this is orthogonal to determining repairs at the level of activity labels as described in this paper.

Fourth, the techniques proposed in this work aim at repairing a process model using information on conformance analysis between the model and a given event log. The ultimate goal is to repair the model into one that fits the event log perfectly, i.e., certain behavioral equivalence guarantees between the repaired model and the event log are achieved. One can adapt the problem of process model repair to account for other notions of behavioral equivalence. A process model can be repaired to become ('more') bisimilar [van Glabbeek 1990] or ('more') isotactic [Polyvyanyy et al. 2012] with some given normative process model, e.g., an implementation of a system can be repaired to better conform with its specification. To this end, one can reuse the existing quantitative techniques for measuring behavioral equivalence of two process models [Desharnais et al. 2004; de Medeiros et al. 2008; de Alfaro et al. 2009].

Fifth, future work will aim at considering different criteria when searching for optimal repair recommendations. In this paper, the focus was on fitness (i.e., reducing alignment costs) and we did

not consider the other three classical conformance dimensions: simplicity, precision, and generalization [van der Aalst 2011; van der Aalst et al. 2012]. This problem was already considered in [Buijs et al. 2013]. However, instead of using a genetic approach (with no guarantees and possibly time consuming), we would like to use the framework presented in this paper. The effects of inserting or skipping activities on simplicity, precision, and generalization can be (partly) anticipated and quantified. At the same time there should be new operators to remove unused behavior.

Last but not least, one can design new cost schemes to guide model repairs. For example, one may want to give different penalties to the same unmatched label depending on whether it occurred at the beginning or at the end of an alignment. Accordingly, we would like to extend the framework proposed in this paper to account for more sophisticated cost schemes.

Acknowledgments. This research was partly supported by the Australian Research Council's Discovery Projects funding scheme (project number DP120101624).

A. PROOFS

This appendix contains proofs of mathematical statements introduced in this paper.

COROLLARY 4.4 (ALIGNMENT-BASED REPAIR).
If a net system $S' \in \mathbb{S}$ is the result of an alignment-based repair of a net system $S \in \mathbb{S}$ w.r.t. an event log $L \in \mathcal{B}(\mathcal{A}^*)$ as per a cost function on label moves f using a repair recommendation, then it holds that $cost[L,S',c'] \leq cost[L,S,c]$, where c and c' are the cost functions on legal moves over S and S' induced by f, respectively.

PROOF. It suffices to show that for every trace υ in L it holds that $cost[\upsilon,S',c'] \leq cost[\upsilon,S,c]$. According to Definition 4.3, there exists a function ψ that justifies the fact that S' is the result of an alignment-based repair of S. Let $\alpha \in L$ be a trace. Then, because L fits S' at least as 'good' as it fits S, cf. Definition 4.3, it holds that $(c' \circ \psi)(\alpha) \leq cost[\alpha,S,c]$. Let γ be an optimal alignment between α and S' as per c'; note that γ always exists. Then, according to Definition 3.6, it holds that $c'(\gamma) \leq (c' \circ \psi)(\alpha)$. Finally, note that $c'(\gamma) = cost[\alpha,S',c']$. ∎

LEMMA 4.7 (ALIGNMENT-BASED REPAIR).
If a net system $S' \in \mathbb{S}$ is the result of a naïve repair of a net system $S \in \mathbb{S}$ based on the alignments in the image of a function ϕ, which maps every trace υ in an event log $L \in \mathcal{B}(\mathcal{A}^*)$ to an optimal alignment in $\mathbb{O}^{\upsilon}_{S,c}$, using a repair recommendation R, where c is the cost function on legal moves over S induced by a cost function on label moves f, then S' is the result of an alignment-based repair of S w.r.t. L as per f using R.

PROOF. Let ψ be a function that maps every trace in L to an alignment between this trace and an execution of S' for which it holds that $\psi(\upsilon) = similar[\phi(\upsilon),S,S']$, $\upsilon \in L$; note that the fact that $similar[\phi(\upsilon),S,S'] \in \mathbb{A}^{\upsilon}_{S'}$ follows from the definition of the *similar* construction on alignments, refer to Section 4. Then, ψ justifies the fact that S' is the result of an alignment-based repair of S. Both statements that (i) L fits S' at least as 'good' as it fits S and (ii) S' fulfills R follow immediately from Definition 4.6 and the *similar* construction on alignments. Let $\upsilon \in L$ be a trace. It holds that $(c' \circ \psi)(\upsilon) \leq cost[\upsilon,S,c]$, where c' is the cost function on legal moves over S' induced by f. Note that $cost[\upsilon,S,c] = (c \circ \phi)(\upsilon)$ and $\psi(\upsilon)$ is constructed from $\phi(\upsilon)$ by replacing some moves with moves of zero costs as per c', while the cost of every other move of $\psi(\upsilon)$ as per c' is equal to the cost of this move as per c. Finally, S' fulfills $R := (R_{ins}, R_{skp})$ because $similar[\phi(\upsilon),S,S']$ guarantees the absence of moves $(\alpha, \gg)$, $\alpha \in R_{ins}$, and $\{(\gg, y) \in \{\gg\} \times T \mid \lambda(y) \in R_{skp}\}$ in the resulting alignment, where T and λ are the set of transitions of S' and the function that assigns labels to transitions in S', respectively. ∎

The next proposition captures the statement that will be used as part of proofs of subsequent results.

PROPOSITION A.1 (SIMILAR ALIGNMENTS).
If a net system $S' \in \mathbb{S}$ is the result of a naïve repair of a net system $S \in \mathbb{S}$ based on the alignments in the

image of a function ϕ using a repair recommendation R, where ϕ maps every trace υ in an event log $L \in \mathcal{B}(\mathcal{A}^*)$ to an alignment in $\mathbb{A}^{\upsilon}_{S}$, then it holds that $c(\gamma) = c'(similar[\gamma, S, S'])$, where $\gamma \in img(\phi)$, c is the cost function on legal moves over S induced by $f|_R$, c' is the cost function on legal moves over S' induced by f, and f is a cost function on label moves. ⌟

The proof of Proposition A.1 follows immediately from the *similar* construction on alignments and Definitions 4.2, 4.6 and 5.2.

THEOREM 5.3 (OPTIMAL ALIGNMENT-BASED REPAIR).
If a net system $S' \in \mathbb{S}$ is the result of a naïve repair of a net system $S \in \mathbb{S}$ based on the alignments in the image of a function ϕ using a repair recommendation R, where ϕ maps every trace υ in an event log $L \in \mathcal{B}(\mathcal{A}^*)$ to an alignment in $\mathbb{O}^{\upsilon}_{S,c}$, c is the cost function on legal moves over S induced by $f|_R$, f is a cost function on label moves, then S' is the result of an optimal alignment-based repair of S w.r.t. L as per f using R. ⌟

PROOF. Let us assume that S' is the result of a naïve repair of S based on the alignments in the image of ϕ using $R := (R_{ins}, R_{skp})$, but S' is *not* the result of an optimal alignment-based repair of S w.r.t. L as per f using R. Then, according to Definition 4.3, there exists no function that maps every trace $\upsilon \in L$ to an alignment in $\mathbb{O}^{\upsilon}_{S',c'}$ and justifies the fact that S' is the result of an alignment-based repair of S, where c' is the cost function on legal moves over S' induced by f. Let ψ be the function that maps every trace $\upsilon \in L$ to the alignment $similar[\phi(\upsilon), S, S']$ between υ and an execution of S', i.e., $\psi(\upsilon) := similar[\phi(\upsilon), S, S']$, $\upsilon \in L$. Let $\upsilon \in L$ be a trace.

- It holds that $cost[\upsilon, S, c] \leq cost[\upsilon, S, c'']$, where c'' is the cost function on legal moves over S induced by f. Let us assume that $cost[\upsilon, S, c] > cost[\upsilon, S, c'']$. Then, there exists $\gamma \in \mathbb{O}^{\upsilon}_{S,c''}$ such that for all $\eta \in \mathbb{A}^{\upsilon}_{S}$ it holds that $c''(\gamma) < c(\eta)$. Note that $\gamma \in \mathbb{A}^{\upsilon}_{S}$ and, thus, it must hold that $c''(\gamma) < c(\gamma)$. For every legal move $m \in \mathbb{M}_S$ it holds that $c(m) = 0$ if $m \in (R_{ins} \times \{\gg\}) \cup \{(\gg, y) \in \{\gg\} \times T \mid \lambda(y) \in R_{skp}\}$, and $c(m) = c''(m)$ otherwise; this follows immediately from Definitions 4.2 and 5.2. Therefore, for every $\eta \in \mathbb{A}^{\upsilon}_{S}$ it holds that $c(\eta) \leq c''(\eta)$. We have reached a contradiction. Note that $(c \circ \psi)(\upsilon) = cost[\upsilon, S, c]$ because $\psi(\upsilon) \in \mathbb{O}^{\upsilon}_{S,c}$. It also holds that $(c' \circ \psi)(\upsilon) = (c \circ \psi)(\upsilon)$, cf. Proposition A.1. Hence, it holds that $(c' \circ \psi)(\upsilon) \leq cost[\upsilon, S, c'']$.
- It follows from the definition of the *similar* construction on alignments that $\psi(\upsilon)$ contains no move from the set $(R_{ins} \times \{\gg\}) \cup \{(\gg, y) \in \{\gg\} \times T \mid \lambda(y) \in R_{skp}\}$, where T and λ are the set of transitions of S' and the function that assigns labels to transitions in S', respectively.
- Let us assume that $\psi(\upsilon) \notin \mathbb{O}^{\upsilon}_{S',c'}$. Then, there exists an alignment $\chi \in \mathbb{A}^{\upsilon}_{S'}$ such that $c'(\chi) < (c' \circ \psi)(\upsilon)$. Let $\eta \in \mathbb{A}^{\upsilon}_{S}$ be such that $\chi = similar[\eta, S, S']$. It holds that $c(\eta) = c'(\chi)$ and $(c \circ \phi)(\upsilon) = (c' \circ \psi)(\upsilon)$, cf. Proposition A.1. Therefore, it holds that $c(\eta) < (c \circ \phi)(\upsilon)$, which means that $\phi(\upsilon) \notin \mathbb{O}^{\upsilon}_{S,c}$. We have reached a contradiction and, thus, it holds that $\psi(\upsilon) \in \mathbb{O}^{\upsilon}_{S',c'}$.

We have reached a contradiction as, clearly, ψ maps every trace $\upsilon \in L$ to an alignment in $\mathbb{O}^{\upsilon}_{S',c'}$ and justifies the fact that S' is the result of an alignment-based repair of S. ∎

LEMMA 5.5 (OPTIMAL ALIGNMENT-BASED REPAIR).
If a net system $S' \in \mathbb{S}$ is the result of a naïve repair of a net system $S \in \mathbb{S}$ based on the alignments in the image of a function ϕ using a repair recommendation R, where ϕ maps every trace υ in an event log $L \in \mathcal{B}(\mathcal{A}^*)$ to an alignment in $\mathbb{O}^{\upsilon}_{S,c}$, c is the cost function on legal moves over S induced by $f|_R$, and f is a cost function on label moves, then it holds that $cost[L, S, c] = cost[L, S', c']$, where c' is the cost function on legal moves over S' induced by f. ⌟

PROOF. Note that if for every trace $\upsilon \in L$ it holds that $cost[\upsilon, S, c] = cost[\upsilon, S', c']$, then it also holds that $cost[L, S, c] = cost[L, S', c']$. Let $\upsilon \in L$ be a trace and let $\gamma := similar[\phi(\upsilon), S, S']$. Then, according to Proposition A.1, it holds that $(c \circ \phi)(\upsilon) = c'(\gamma)$. It also holds that $\gamma \in \mathbb{O}^{\upsilon}_{S',c'}$, refer to the proof of Theorem 5.3. Therefore, it holds that $cost[\upsilon, S, c] = cost[\upsilon, S', c']$. ∎

B. GLOSSARY

This appendix contains a glossary of the specific mathematical notations used in this paper.

Table V: Glossary of the specific mathematical notations used in this paper. The 'Notation' column exemplifies the notation. The 'Page' column specifies pages at which the corresponding notation is defined. The 'Parameters' column explains the parameters of the notation. Finally, the 'Meaning' column states the meaning of the notation in natural language.

Notation	Page	Parameters	Meaning
$\mathbb{A}_S^\upsilon$	10	υ – a trace S – a net system	The set that for every execution σ of S contains all the alignments between υ and σ, and contains nothing else.
$\mathcal{B}(A)$	4	A – a set	The set of all finite multisets over A.
$cost[R,C]$	18	R – a repair recommendation C – a repair constraint	The cost of R as per C.
$cost[\upsilon,S,c]$	11	υ – a trace S – a net system c – a cost function on legal moves over S	The cost of some optimal alignment between υ and S as per c.
$cost[L,S,c]$	11	L – an event log S – a net system c – a cost function on legal moves over S	The cost of some optimal alignment between L and S as per c.
$f\|_R$	19	f – a cost function on label moves R – a repair recommendation	The adjusted cost function on label moves induced by f using R.
$\mathbb{E}_S$	7	S – a net system	The set of all executions of S.
$labels(L)$	23	L – an event log	The set of all labels that are elements of. traces in L
$labels(S)$	23	S – a net system	The set of all labels assigned to observable transitions of S.
$\mathbb{M}_S$	9	S – a net system	The set of all legal moves over S.
$mhs(A)$	14	A – a set of sets	A minimal hitting set of A.
$\mathbb{O}_{S,c}^\upsilon$	11	υ – a trace S – a net system c – a cost function on legal moves over S	The set of all optimal alignments between υ and S as per c.
$R_1 \sqsubseteq R_2$	22	R_1 – a repair recommendation R_2 – a repair recommendation	R_1 is contained in R_2.
$R_1 \sqsubset R_2$	22	R_1 – a repair recommendation R_2 – a repair recommendation	R_1 is properly contained in R_2.
$\mathcal{R}_{max}$	22	$\mathcal{R}$ – a set of repair recommendations	The set of all maximal (as per the containment relation) repair recommendations in $\mathcal{R}$.
$\mathcal{R}_{min}$	22	$\mathcal{R}$ – a set of repair recommendations	The set of all minimal (as per the containment relation) repair recommendations in $\mathcal{R}$.
$\mathbb{R}[S,L,f,C]$	20	S – a net system L – an event log f – a cost function on label moves C – a repair constraint	The set of all optimal repair recommendations for S w.r.t. L as per f restricted by C.
$\mathbb{R}_{min}[S,L,f,C]$	20	S – a net system L – an event log f – a cost function on label moves C – a repair constraint	The set of all minimal optimal repair recommendations for S w.r.t. L as per f restricted by C.
$\mathbb{S}$	6		The universe of net systems.

REFERENCES

Arya Adriansyah. 2014. *Aligning Observed and Modeled Behavior*. Ph.D. Dissertation. Technische Universiteit Eindhoven. http://dx.doi.org/10.6100/IR770080

Arya Adriansyah, Boudewijn F. van Dongen, and Wil M.P. van der Aalst. 2011. Conformance Checking Using Cost-Based Fitness Analysis. In *Proceedings of the 15th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2011, Helsinki, Finland, August 29–September 2, 2011*. IEEE Computer Society, 55–64. http://dx.doi.org/10.1109/EDOC.2011.12

Rumen Andonov, Vincent Poirriez, and Sanjay V. Rajopadhye. 2000. Unbounded knapsack problem: Dynamic programming revisited. *European Journal of Operational Research (EOR)* 123, 2 (2000), 394–407.

Thomas Baier, Claudio Di Ciccio, Jan Mendling, and Mathias Weske. 2015a. Matching of Events and Activities—An Approach Using Declarative Modeling Constraints. In *Enterprise, Business-Process and Information Systems Modeling—16th International Conference, BPMDS 2015, 20th International Conference, EMMSAD 2015, Held at CAiSE 2015, Stockholm, Sweden, June 8–9, 2015, Proceedings (Lecture Notes in Business Information Processing)*, Vol. 214. Springer, 119–134. http://dx.doi.org/10.1007/978-3-319-19237-6_8

Thomas Baier, Andreas Rogge-Solti, Jan Mendling, and Mathias Weske. 2015b. Matching of events and activities: an approach based on behavioral constraint satisfaction. In *Proceedings of the 30th Annual ACM Symposium on Applied Computing, Salamanca, Spain, April 13–17, 2015*. ACM, 1225–1230.

Joos C.A.M. Buijs, Marcello La Rosa, Hajo A. Reijers, Boudewijn F. van Dongen, and Wil M.P. van der Aalst. 2013. Improving business process models using observed behavior. In *Data-Driven Process Discovery and Analysis*. Springer, 44–59.

Joos C.A.M. Buijs, Boudewijn F. van Dongen, and Wil M.P. van der Aalst. 2012. On the Role of Fitness, Precision, Generalization and Simplicity in Process Discovery. In *On the Move to Meaningful Internet Systems: OTM 2012*, Robert Meersman, Hervé Panetto, Tharam Dillon, Stefanie Rinderle-Ma, Peter Dadam, Xiaofang Zhou, Siani Pearson, Alois Ferscha, Sonia Bergamaschi, and IsabelF. Cruz (Eds.). Lecture Notes in Computer Science, Vol. 7565. Springer Berlin Heidelberg, 305–322. http://dx.doi.org/10.1007/978-3-642-33606-5_19

Joos C.A.M. Buijs, Boudewijn F. van Dongen, and Wil M.P. van der Aalst. 2013. Discovering and Navigating a Collection of Process Models Using Multiple Quality Dimensions. In *Business Process Management Workshops—BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers (Lecture Notes in Business Information Processing)*, Vol. 171. Springer, 3–14. http://dx.doi.org/10.1007/978-3-319-06257-0_1

George B. Dantzig. 1957. Discrete-Variable Extremum Problems. *Operations Research* 5, 2 (1957), 266–277.

Luca de Alfaro, Marco Faella, and Mariëlle Stoelinga. 2009. Linear and Branching System Metrics. *IEEE Transactions on Software Engineering (TSE)* 35, 2 (2009), 258–273. http://dx.doi.org/10.1109/TSE.2008.106

Massimiliano de Leoni and Wil M.P. van der Aalst. 2013a. Aligning Event Logs and Process Models for Multi-perspective Conformance Checking: An Approach Based on Integer Linear Programming. In *Business Process Management—11th International Conference, BPM 2013, Beijing, China, August 26–30, 2013. Proceedings (Lecture Notes in Computer Science)*, Vol. 8094. Springer, 113–129. http://dx.doi.org/10.1007/978-3-642-40176-3_10

Massimiliano de Leoni and Wil M.P. van der Aalst. 2013b. Data-aware Process Mining: Discovering Decisions in Processes Using Alignments. In *Proceedings of the 28th Annual ACM Symposium on Applied Computing (SAC'13)*. ACM, New York, NY, USA, 1454–1461. http://dx.doi.org/10.1145/2480362.2480633

Ana Karla Alves de Medeiros, Wil M.P. van der Aalst, and A.J.M.M. Weijters. 2008. Quantifying process equivalence based on observed behavior. *Data & Knowledge Engineering (DKE)* 64, 1 (2008), 55–74. http://dx.doi.org/10.1016/j.datak.2007.06.010

Ana Karla Alves de Medeiros, A.J.M.M. Weijters, and Wil M.P. van der Aalst. 2007. Genetic process mining: an experimental evaluation. *Data Mining and Knowledge Discovery* 14, 2 (2007), 245–304.

Josee Desharnais, Vineet Gupta, Radha Jagadeesan, and Prakash Panangaden. 2004. Metrics for labelled Markov processes. *Theoretical Computer Science* 318, 3 (2004), 323–354. http://dx.doi.org/10.1016/j.tcs.2003.09.013

Johann Eder, Juergen Mangler, Enrico Mussi, and Barbara Pernici. 2009. Using Stateful Activities to Facilitate Monitoring and Repair in Workflow Choreographies. In *2009 IEEE Congress on Services, Part I, SERVICES I 2009, Los Angeles, CA, USA, July 6–10, 2009*. IEEE Computer Society, 219–226. http://dx.doi.org/10.1109/SERVICES-I.2009.49

Dirk Fahland and Wil M.P. van der Aalst. 2015. Model repair—aligning process models to reality. *Information Systems (IS)* 47 (2015), 220–243. http://dx.doi.org/10.1016/j.is.2013.12.007

Didier Fayard and Gérard Plateau. 1975. Resolution of the 0-1 knapsack problem: Comparison of methods. *Mathematical Programming (MP)* 8, 1 (1975), 272–307.

Gerhard Friedrich, Mariagrazia Fugini, Enrico Mussi, Barbara Pernici, and Gaston Tagni. 2010. Exception Handling for Repair in Service-Based Processes. *IEEE Transactions on Software Engineering (TSE)* 36, 2 (2010), 198–215. http://dx.doi.org/10.1109/TSE.2010.8

Walid Gaaloul, Sami Bhiri, and Mohsen Rouached. 2010. Event-Based Design and Runtime Verification of Composite Service Transactional Behavior. *IEEE Transactions on Services Computing (TSC)* 3, 1 (2010), 32–45. http://dx.doi.org/10.1109/TSC.2010.1

Walid Gaaloul, Khaled Gaaloul, Sami Bhiri, Armin Haller, and Manfred Hauswirth. 2009. Log-based transactional workflow mining. *Distributed and Parallel Databases (DPD)* 25, 3 (2009), 193–240. http://dx.doi.org/10.1007/s10619-009-7040-0

E.M. Goldratt, J. Cox, and D. Whitford. 2012. *The Goal: A Process of Ongoing Improvement*. North River Press.

Gianluigi Greco, Antonella Guzzo, Francesco Lupia, and Luigi Pontieri. 2015. Process Discovery under Precedence Constraints. *ACM Transactions on Knowledge Discovery from Data (TKDD)* 9, 4 (2015), 32. http://dx.doi.org/10.1145/2710020

Christian W. Günther and Wil M.P. van der Aalst. 2007. Fuzzy Mining—Adaptive Process Simplification Based on Multi-perspective Metrics. In *Business Process Management, 5th International Conference, BPM 2007, Brisbane, Australia, September 24–28, 2007, Proceedings (Lecture Notes in Computer Science)*, Vol. 4714. Springer, 328–343. http://dx.doi.org/10.1007/978-3-540-75183-0_24

Farhana Islam, Meherun Nesa Lucky, and Barbara Pernici. 2010. Business Analysis of Web Service Repairability. In *Proceedings of the Fourth IEEE International Conference on Research Challenges in Information Science, RCIS 2010, Nice, France, May 19–21, 2010*. IEEE, 463–472. http://dx.doi.org/10.1109/RCIS.2010.5507407

Richard M. Karp. 1972. Reducibility Among Combinatorial Problems. In *Proceedings of a symposium on the Complexity of Computer Computations, held March 20–22, 1972, at the IBM Thomas J. Watson Research Center, Yorktown Heights, New York. (The IBM Research Symposia Series)*, Raymond E. Miller and James W. Thatcher (Eds.). Plenum Press, New York, 85–103. http://www.cs.berkeley.edu/~luca/cs172/karp.pdf

Aline A. S. Leão, Luiz H. Cherri, and Marcos N. Arenales. 2014. Determining the K-best solutions of knapsack problems. *Computers & OR* 49 (2014), 71–82.

Sander J.J. Leemans, Dirk Fahland, and Wil M.P. van der Aalst. 2013. Discovering Block-Structured Process Models from Event Logs Containing Infrequent Behaviour. In *Business Process Management Workshops—BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers (Lecture Notes in Business Information Processing)*, Niels Lohmann, Minseok Song, and Petia Wohed (Eds.), Vol. 171. Springer International Publishing, 66–78. http://dx.doi.org/10.1007/978-3-319-06257-0_6

Sander J. J. Leemans, Dirk Fahland, and Wil M.P. van der Aalst. 2015. Scalable Process Discovery with Guarantees. In *Enterprise, Business-Process and Information Systems Modeling (BPMDS 2015) (Lecture Notes in Business Information Processing)*, K. Gaaloul, R. Schmidt, S. Nurcan, S. Guerreiro, and Q. Ma (Eds.), Vol. 214. Springer International Publishing, 85–101.

Chen Li, Manfred Reichert, and Andreas Wombacher. 2009. Discovering Reference Models by Mining Process Variants Using a Heuristic Approach. In *Business Process Management*, Umeshwar Dayal, Johann Eder, Jana Koehler, and Hajo A. Reijers (Eds.). Lecture Notes in Computer Science, Vol. 5701. Springer Berlin Heidelberg, 344–362. http://dx.doi.org/10.1007/978-3-642-03848-8_23

Chen Li, Manfred Reichert, and Andreas Wombacher. 2010. The Minadept Clustering Approach for Discovering Reference Process Models Out of Process Variants. *International Journal of Cooperative Information Systems* 19, 3–4 (2010), 159–203. http://dx.doi.org/10.1142/S0218843010002139

Chen Li, Manfred Reichert, and Andreas Wombacher. 2011. Mining business process variants: Challenges, scenarios, algorithms. *Data & Knowledge Engineering (DKE)* 70, 5 (2011), 409–434. http://dx.doi.org/10.1016/j.datak.2011.01.005

Felix Mannhardt, Massimiliano de Leoni, Hajo A. Reijers, and Wil M.P. van der Aalst. 2015. Balanced multi-perspective checking of process conformance. *Computing* (2015). http://dx.doi.org/10.1007/s00607-015-0441-1

Silvano Martello, David Pisinger, and Paolo Toth. 2000. New trends in exact algorithms for the 0-1 knapsack problem. *European Journal of Operational Research (EOR)* 123, 2 (2000), 325–332.

Thomas Molka, David Redlich, Marc Drobek, Artur Caetano, Xiao-Jun Zeng, and Wasif Gilani. 2014. Conformance Checking for BPMN-based Process Models. In *Proceedings of the 29th Annual ACM Symposium on Applied Computing (SAC'14)*. ACM, New York, NY, USA, 1406–1413. http://dx.doi.org/10.1145/2554850.2555061

Jorge Munoz-Gama and Josep Carmona. 2011. Enhancing precision in Process Conformance: Stability, confidence and severity. In *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining, CIDM 2011, part of the IEEE Symposium Series on Computational Intelligence 2011, April 11–15, 2011, Paris, France*. IEEE, 184–191. http://dx.doi.org/10.1109/CIDM.2011.5949451

Jorge Munoz-Gama, Josep Carmona, and Wil M.P. van der Aalst. 2013. Conformance Checking in the Large: Partitioning and Topology. In *Business Process Management—11th International Conference, BPM 2013, Beijing, China, August 26–30, 2013. Proceedings (Lecture Notes in Computer Science)*, Vol. 8094. Springer, 130–145. http://dx.doi.org/10.1007/978-3-642-40176-3_11

Jorge Munoz-Gama, Josep Carmona, and Wil M.P. van der Aalst. 2013. Hierarchical conformance checking of process models based on event logs. In *Application and Theory of Petri Nets and Concurrency*. Springer, 291–310.

Mike P. Papazoglou. 2003. Service-Oriented Computing: Concepts, Characteristics and Directions. In *4th International Conference on Web Information Systems Engineering, WISE 2003, Rome, Italy, December 10–12, 2003*. IEEE Computer Society, 3–12. http://dx.doi.org/10.1109/WISE.2003.1254461

Artem Polyvyanyy. 2012. *Structuring Process Models*. Ph.D. Dissertation. University of Potsdam. http://nbn-resolving.de/urn/resolver.pl?urn:nbn:de:kobv:517-opus-59024

Artem Polyvyanyy and Christoph Bussler. 2013. The Structured Phase of Concurrency. In *Seminal Contributions to Information Systems Engineering, 25 Years of CAiSE*, Janis A. Bubenko Jr., John Krogstie, Oscar Pastor, Barbara Pernici, Colette Rolland, and Arne Sølvberg (Eds.). Springer, 257–263. http://dx.doi.org/10.1007/978-3-642-36926-1_20

Artem Polyvyanyy, Jussi Vanhatalo, and Hagen Völzer. 2011. Simplified Computation and Generalization of the Refined Process Structure Tree. In *Web Services and Formal Methods—7th International Workshop, WS-FM 2010, Hoboken, NJ, USA, September 16–17, 2010. Revised Selected Papers (Lecture Notes in Computer Science)*, Mario Bravetti and Tevfik Bultan (Eds.), Vol. 6551. Springer, 25–41. http://dx.doi.org/10.1007/978-3-642-19589-1_2

Artem Polyvyanyy, Matthias Weidlich, and Mathias Weske. 2012. Isotactics as a Foundation for Alignment and Abstraction of Behavioral Models. In *Business Process Management—10th International Conference, BPM 2012, Tallinn, Estonia, September 3–6, 2012. Proceedings (Lecture Notes in Computer Science)*, Alistair Barros, Avigdor Gal, and Ekkart Kindler (Eds.), Vol. 7481. Springer, 335–351. http://dx.doi.org/10.1007/978-3-642-32885-5_26

Wolfgang Reisig. 1998. *Elements of Distributed Algorithms: Modeling and Analysis with Petri Nets*. Springer-Verlag Berlin Heidelberg New York. http://www.springer.com/computer/theoretical+computer+science/book/978-3-540-62752-4

Anne Rozinat and Wil M.P. van der Aalst. 2008. Conformance checking of processes based on monitoring real behavior. *Information Systems (IS)* 33, 1 (2008), 64–95. http://dx.doi.org/10.1016/j.is.2007.07.001

Wil M.P. van der Aalst. 1997. Verification of Workflow Nets. In *Application and Theory of Petri Nets 1997, 18th International Conference, ICATPN '97, Toulouse, France, June 23–27, 1997, Proceedings (Lecture Notes in Computer Science)*, Pierre Azéma and Gianfranco Balbo (Eds.), Vol. 1248. Springer Berlin Heidelberg, 407–426. http://dx.doi.org/10.1007/3-540-63139-9_48

Wil M.P. van der Aalst. 2011. *Process Mining: Discovery, Conformance and Enhancement of Business Processes* (1st ed.). Springer-Verlag Berlin Heidelberg. http://www.springer.com/gp/book/9783642193446

Wil M.P. van der Aalst. 2013. Decomposing Petri nets for process mining: A generic approach. *Distributed and Parallel Databases* 31, 4 (2013), 471–507. http://dx.doi.org/10.1007/s10619-013-7127-5

Wil M.P. van der Aalst, Arya Adriansyah, and Boudewijn F. van Dongen. 2012. Replaying history on process models for conformance checking and performance analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery (WIDM)* 2, 2 (2012), 182–192. http://dx.doi.org/10.1002/widm.1045

Wil M.P. van der Aalst, Kees M. van Hee, Jan Martijn E.M. van der Werf, and Marc Verdonk. 2010. Auditing 2.0: Using Process Mining to Support Tomorrow's Auditor. *IEEE Computer (COMPUTER)* 43, 3 (2010), 90–93. http://dx.doi.org/10.1109/MC.2010.61

Wil M.P. van der Aalst, A.J.M.M. Weijters, and Laura Maruster. 2004. Workflow mining: discovering process models from event logs. *IEEE Transactions on Knowledge and Data Engineering* 16, 9 (Sept 2004), 1128–1142. http://dx.doi.org/10.1109/TKDE.2004.47

Boudewijn F. van Dongen, Ana Karla Alves de Medeiros, H.M.W. Verbeek, A.J.M.M. Weijters, and Wil M.P. van der Aalst. 2005. The ProM Framework: A New Era in Process Mining Tool Support. In *Applications and Theory of Petri Nets 2005, 26th International Conference, ICATPN 2005, Miami, USA, June 20–25, 2005, Proceedings (Lecture Notes in Computer Science)*, Gianfranco Ciardo and Philippe Darondeau (Eds.), Vol. 3536. Springer Berlin Heidelberg, 444–454. http://dx.doi.org/10.1007/11494744_25

Rob J. van Glabbeek. 1990. The Linear Time-Branching Time Spectrum (Extended Abstract). In *CONCUR'90, Theories of Concurrency: Unification and Extension, Amsterdam, The Netherlands, August 27–30, 1990, Proceedings (Lecture Notes in Computer Science)*, Vol. 458. Springer, 278–297. http://dx.doi.org/10.1007/BFb0039066

Pamela H. Vance. 1993. Knapsack Problems: Algorithms and Computer Implementations. *SIAM Rev.* 35, 4 (1993), 684–685. http://dx.doi.org/10.1137/1035174

Jussi Vanhatalo, Hagen Völzer, and Jana Koehler. 2009. The refined process structure tree. *Data & Knowledge Engineering (DKE)* 68, 9 (2009), 793–818. http://dx.doi.org/10.1016/j.datak.2009.02.015

H.M.W. Verbeek, Joos C.A.M. Buijs, Boudewijn F. van Dongen, and Wil M.P. van der Aalst. 2011. XES, XESame, and ProM 6. In *Information Systems Evolution—CAiSE Forum 2010, Hammamet, Tunisia, June 7–9, 2010, Selected Extended Papers (Lecture Notes in Business Information Processing)*, Vol. 72. Springer, 60–75. http://dx.doi.org/10.1007/978-3-642-17722-4_5

A.J.M.M. Weijters, Wil M.P. van der Aalst, and Ana Karla Alves de Medeiros. 2006. Process mining with the heuristics miner-algorithm. *Technische Universiteit Eindhoven, Tech. Rep. WP* 166 (2006), 1–34.

Franz Wotawa. 2001. A variant of Reiter's hitting-set algorithm. *Information Processing Letters (IPL)* 79, 1 (2001), 45–51. http://dx.doi.org/10.1016/S0020-0190(00)00166-6