Single-Entry Single-Exit Decomposed Conformance Checking

Jorge Munoz-Gama[a], Josep Carmona[a], Wil M.P. van der Aalst[b,c]

[a] Universitat Politecnica de Catalunya, Barcelona (Spain)
[b] Eindhoven University of Technology, Eindhoven (The Netherlands)
[c] PAIS Lab, Higher School of Economics, Moscow (Russia)

Abstract

An exponential growth of event data can be witnessed across all industries. Devices connected to the internet ("internet of things"), social interaction, mobile computing, and cloud computing provide new sources of event data and this trend will continue. The omnipresence of large amounts of event data is an important enabler for process mining. Process mining techniques can be used to discover, monitor and improve real processes by extracting knowledge from observed behavior. However, unprecedented volumes of event data also provide new challenges with which state-of-the-art process mining techniques often cannot cope. This paper focuses on "conformance checking in the large" and presents a novel decomposition technique that partitions larger process models and event logs into smaller parts that can be analyzed independently. The so-called Single-Entry Single-Exit (SESE) decomposition not only helps to speed up conformance checking, but also provides improved diagnostics. The analyst can zoom in on the problematic parts of the process. Importantly, the conditions under which the conformance of the whole can be assessed by verifying the conformance of the SESE parts are described, which enables the decomposition and distribution of large conformance checking problems. All the techniques have been implemented in ProM, and experimental results are provided.

Keywords: Process Mining, Conformance Checking, Decomposition, Process Diagnosis

1. Introduction

In the last decade process mining emerged as a novel discipline for addressing challenges related to Business Process Management (BPM) and "Big Data" [1]. Information systems (and many other computer-supported systems) record overwhelming amounts of event data. These can be seen as the "footprints" left by the process. For example, Boeing jet engines may produce up to 10 terabytes of operational information every 30 minutes, and Walmart logs may store one million customer transactions per hour [2].

Event logs can be used to conduct three types of process mining [1].The first and most prominent is *discovery*. A discovery technique takes an event log and produces a model without using a priori information. For many organizations it is surprising that existing techniques are indeed able to discover real processes based only on example behaviors recorded in event logs. The second type is *conformance* where an existing process model is compared with an event log of the same process. Conformance checking can be used to check if reality, as recorded in the log, conforms to the model and vice versa. The third type is *enhancement* where the idea is to extend or improve an existing process model using information about the actual process recorded in an event log. Whereas conformance checking measures alignment between model and reality, this third type of process mining aims to change or extend the a priori model; for instance, using timestamps in the event log, one can extend the model to show bottlenecks, service levels, throughput times, and frequencies.

In conformance checking, the seminal work by Rozinat *et al.*[3] was the first in formalizing the problem and enumerating the four dimensions to consider for determining the adequacy of a model in describing a log: *fitness*, *precision*, *generalization* and *simplicity*. In this paper, we will focus on evaluating fitness, that measures the capability of a model in reproducing the traces of a log. As modeling notation, we will focus on the Petri net formalism [4], although most of the conclusions of the paper can be generalized to similar process formalisms [5].

Email addresses: `jmunoz@lsi.upc.edu` (Jorge Munoz-Gama), `jcarmona@lsi.upc.edu` (Josep Carmona), `w.m.p.v.d.aalst@tue.nl` (Wil M.P. van der Aalst)

In real-life situations, event logs often do not fit its corresponding models, i.e., some log traces cannot be fully reproduced in the model. These non-fitting situations should be communicated to the stakeholders, in order to take decisions on the process object of study. However, in reality process models can be non-deterministic, which complicates the analysis. Non-determinism may arise when the model contains *silent* or *duplicate* activities, which is often the case in practice. Moreover, the presence of *noise* in the log (rare or infrequent behavior that has been recorded in the log) complicates even more the algorithmic detection of non-fitting situations. Due to this, the initial proposal from Rozinat *et al.* to *replay* log traces in a model in order to assess whether a trace can fit a model has been recently reconsidered, giving rise to the notion of *alignment*. Alignment techniques relate execution sequences of the model and traces in the event log. The techniques can cope with deviations and models with duplicate/invisible activities [6, 7, 8]. However, alignment techniques are extremely challenging from a computational point of view. Traces in the event log need to be mapped on paths in the model. A model may have infinitely many paths and the traces may have an arbitrary amount of deviating events. Hence, although the algorithms have demonstrated to be of great value for undertaking small or medium-sized problem instances [1, 9], they are often unable to handle problems of industrial size. We believe that decomposition techniques are an important means to tackle much large and more complex process mining problems. Therefore, this paper addresses this problem through *decomposition and distribution*.

There is a trivial way to decompose the conformance checking problem. One can simply split the event log into sublogs such that every trace appears in precisely one of these sublogs. Note that the conformance is still checked on the whole model. Linear speed-ups are possible using such a simple decomposition. However, the real complexity is in the size of the model and the number of different activities in the event log. Therefore, we propose a different approach. Instead of trying to assess the conformance of the whole event log and the complete Petri net, conformance checking is only performed for selected subprocesses (subnets of the initial Petri net and corresponding sublogs). Subprocesses are identified as subnets of the Petri net that have a single-entry and a single-exit node (*SESE*), thus representing an isolated part of the model with a well-defined interface to the rest of the net. SESEs can be efficiently computed and hierarchically represented in a tree-like manner into the Refined Process Structured Tree (RPST) [10].

Experiments (cf. Sec.6) show a *considerable reduction (orders of magnitude) in the time required to perform fitness checking*. Moreover the techniques presented in this paper allow for identifying those subnets that have fitness problems, allowing the process owner to focus on the problematic parts of a large model. Importantly, we have performed analytical comparisons of the conformance problem when decomposition is considered or not, related to the size of the components and the average length of the log traces. In terms of complexity, those studies reveal a clear superiority of the methods proposed in this paper, being more robust for these important matters (size of the components and length of log traces). Remarkably, this significant complexity alleviation comes without any penalty on the capability of the method: by applying decomposition techniques conformance checking of the whole can still be assessed.

The SESE decomposition is not only used for efficiency reasons. We also use it to provide *diagnostics* that help the analyst in localizing conformance problems. We create a topological structure of SESEs in order to detect the larger connected components that have fitness problems. Moreover, problematic parts can be analyzed in isolation. Finally, a hierarchical perspective of the conformance checking problem is presented, which may open the door for zoom-in zoom-out analysis, and also focus the analysis of the hierarchy into particular subprocesses that have common features.

This paper extends and generalizes two recent conference papers [11, 12]. The extensions and generalizations can be summarized as follows. First, we present a strategy to compute fitness by adapting a partitioning of the RPST in order to satisfy the valid decomposition requirements from [13]. Second, we have extended the decomposition approach of [12] in order to deal with silent and duplicate activities. Third, different perspectives to the conformance problem are presented which aim to provide more flexibility during analysis. Three alternatives to the traditional conformance checking practice are proposed: (1) subprocess, (2) hierarchical and (3) filtered conformance checking. The filtered conformance checking perspective is based on the initial one presented in [11], but a new data perspective is proposed in this paper. Fourth, we have reimplemented the initial architecture of [11, 12] in order to address problems encountered when analyzing large event logs. The new implementation results in some cases in speed-ups of orders of magnitude. Fifth, we have extended considerably the empirical evaluation from [11, 12], incorporating studies that relate the performance of the decomposed technique with respect to the log trace length and the size of the subprocesses.

The paper is structured as follows: Sect. 2 introduces preliminaries needed in the remainder. In Sect. 3 the SESE-decomposition is presented formally. Section 4 describes the high-level use of the RPST structure for diagnostics of conformance checking problems, by means of a topology of SESE components. Various applications of the decomposition technique are presented in Sect. 5. Section 6 presents the experimental evaluation of the techniques described in this paper. Related work is discussed in Sect. 7. Section 8 concludes the paper.

2. Preliminaries

2.1. Mathematical Preliminaries

Definition 1 (Multisets) *Given a set X, a multiset M of X is a mapping $M : X \to \mathbb{N}$. $\mathcal{B}(X)$ denotes the set of all multisets over X.*

Multisets can be represented in vector format, i.e., $[x^3, y, z^2]$ is the multiset that has three occurrences of x, one of y and two occurrences of z. $M_1 \leq M_2$ if $M_1(x) \leq M_2(x)$ for all $x \in X$. (Domains are extended if needed.) For example, $[y, z] \leq [x^3, y, z^2]$ and $[y^2] \nleq [x^3, y, z^2]$. The difference $(M_1 - M_2)$ and union $(M_1 + M_2)$ are defined as usual. For example, $[x^3, y, z^2] - [y, z] = [x^3, z]$.

Definition 2 (Projection) *Let X be a set and $Q \subseteq X$ one of its subsets. $\sigma\upharpoonright_Q$ denotes the projection of $\sigma \in W^*$ on Q, e.g., $aabc\upharpoonright_{\{a,c\}}= aac$. The projection can also be applied to multisets, e.g., $[x^3, y, z^2]\upharpoonright_{\{x,y\}}= [x^3, y]$.*

2.2. Petri Nets and Logs

For a deeper introduction of Petri nets the reader is refereed to [4].

Definition 3 (Petri Net, Workflow Net) *A Petri net is a tuple $PN = (P, T, A)$, being P the set of places, T the set of transitions, where $P \cap T = \emptyset$, and $A \subseteq (P \times T) \cup (T \times P)$ the flow relation. For a node n (place or transition) of a Petri net, $\bullet n$ ($n\bullet$) is the predecessor (successor) set of n in A.*

A workflow net WF-net = $(P, T, A, start, end)$ is a particular type of Petri net where the net has one source place 'start' *and one sink place* 'end'*, and all the other nodes are in a path between them.*

A *marking* in a Petri net defines the global state, which is distributed among its places. Formally:

Definition 4 (Marking, Firability) *Let $PN = (P, T, A)$ be a Petri net. A marking M is a multiset of places, i.e., $M \in \mathcal{B}(P)$. A transition $t \in T$ is* enabled *in a marking M iff $\bullet t \leq M$.* Firing *transition t in M results in a new marking $M' = M - \bullet t + t\bullet$, i.e., tokens are removed from $\bullet t$ and added to $t\bullet$. A marking M' is* reachable *from M if there is a sequence of firings $\sigma = t_1 t_2 \ldots t_n$ that transforms M into M', denoted by $M[\sigma\rangle M'$.*

Definition 5 (System Net) *A system net is a tuple $SN = (PN, M_{ini}, M_{fin})$, where PN is a Petri net and M_{ini}, M_{fin} define the initial and final marking of the net, respectively.*

A system net defines a set of sequences, each one starting from the initial marking and ending in the final marking.

Definition 6 (Full Firing Sequences, Boundedness) *Let $SN = (PN, M_{ini}, M_{fin})$ be a system net. The set $\{\sigma \mid (PN, M_{ini})[\sigma\rangle(PN, M_{fin})\}$ denotes all the full firing sequences of SN. A Petri net is said to be* k-bounded *or simply* bounded *if all the possible markings in the set of full firing sequences are bounded by k (e.g., no place contains more than k tokens). When k is 1 the Petri net is called* safe.

An event log is a collection of traces, where a trace may appear more than once. Formally:

Definition 7 (Trace, Event Log) *Let T^* be a* trace*. An* event log *$L \in \mathcal{B}(T^*)$ is a multiset of traces.*

In this simple definition of an event log, an event refers to just an activity. Often event logs store additional information about events, e.g., resource, timestamp, or additional data elements. In this paper, we abstract from such information. However, the results presented can easily be extended to event logs containing additional information.

Process discovery is concerned with learning a process model (e.g., a Petri net) from an event log. The focus of this paper is however on conformance checking, i.e., comparing observed and modeled behavior. There are four quality dimensions for comparing model and log: (1) *replay fitness*, (2) *simplicity*, (3) *precision*, and (4) *generalization* [1]. A model with good *replay fitness* allows for most of the behavior seen in the event log. A model has a perfect fitness if all traces in the log can be replayed by the model from beginning to end. The *simplest* model that can explain the behavior seen in the log is the best model. This principle is known as Occam's Razor. Fitness and simplicity alone are not sufficient to judge the quality of a discovered process model. For example, it is very easy to construct an extremely simple Petri net ("flower model") that is able to replay all traces in an event log (but also any other event log referring to the same set of activities). Similarly, it is undesirable to have a model that only allows for the exact behavior seen in the event log. Remember that the log contains only example behavior and that many traces that are possible may not have been seen yet. A model is *precise* if it does not allow for "too much" behavior. Clearly, the "flower model" lacks precision. A model that is not precise is "underfitting"[14]. Underfitting is the problem that the model over-generalizes the example behavior in the log (i.e., the model allows for behaviors very different from what was seen in the log). At the same time, the model should generalize and not restrict behavior to just the examples seen in the log. A model that does not *generalize* is "overfitting" [15]. Overfitting is the problem that a very specific model is generated whereas it is obvious that the log only holds example behavior (i.e., the model explains the particular sample log, but there is a high probability that the model is unable to explain the next batch of cases).

In the remainder, we will focus on replay fitness which we will simply refer to as *fitness*. The following definition states whether a trace or log fits the model or not.

Definition 8 (Fitting Trace) *A trace $\sigma \in T^*$ fits $SN = (PN, M_{ini}, M_{fin})$ if $(PN, M_{ini})[\sigma\rangle(PN, M_{fin})$, i.e., σ corresponds to a full firing sequence of SN. An event log $L \in \mathcal{B}(T^*)$ fits SN if $(PN, M_{ini})[\sigma\rangle(PN, M_{fin})$ for all $\sigma \in L$.*

Note that different metrics to quantify fitness are possible. For example, we can look at the percentage of fitting traces. Using alignments we can go one step further and look at the event level. Finally, although in this paper we focus on fitness, other dimensions such as precision [7, 8] and generalization [15] can be considered at the level of subnets. However, the computation of the overall precision and generalization in a decomposed way will require revisiting both dimensions because the current definitions are defined globally and cannot be reformulated in a decomposed manner easily.

2.3. SESE and RPST

The intuitive idea behind the decomposition technique in the following section is to find subgraphs that have a simple interface with respect to the rest of the net. The following set of definitions formalize the idea of *Single-Entry Single-Exit* (SESE) subnet and the corresponding decomposition. The underlying theory dates back to the seminal work of Hopcroft and Tarjan in the seventies [16], but recent studies have made considerable progress into making the algorithms practical when applied to process models [17]. We start defining the graph structure used for decomposing a process model:

Definition 9 (Workflow Graph) *Given a Petri net $PN = (P, T, A)$, we define its* workflow graph *simply as the directed graph $G = (V, E)$ where no distinctions are made between places and transitions, i.e., $V = P \cup T$ and $E = A$.*

In the remainder, the following context is assumed: Let G be a workflow graph of a given WF-net, and let $G_S = (V_S, S)$ be a connected subgraph of G formed by a set of edges S and the vertices $V_S = \Pi(S)$ induced by S.[1]

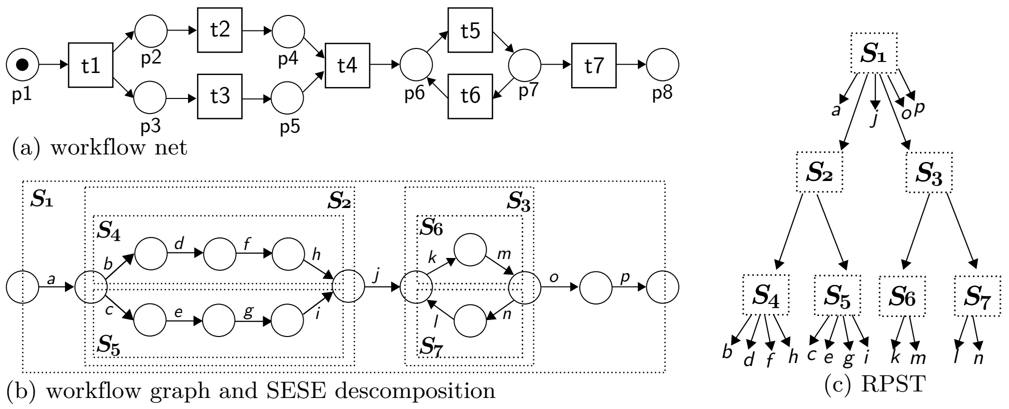
Definition 10 (Subnet nodes [10]) *A node $x \in V_S$ is* interior *with respect to G_S iff it is connected only to nodes in V_S; otherwise x is a* boundary *node of G_S. A boundary node y of G_S is an* entry *of G_S iff no incoming edge of y belongs to S or if all outgoing edges of y belong to S. A boundary node y of G_S is an* exit *of G_S iff no outgoing edge of y belongs to S or if all incoming edges of y belong to S.*

As next definition formalizes, a SESE is a special type of subgraph with a very restricted interface with respect to the rest of the graph:

Definition 11 (SESE [10]) *A set of edges $S \subseteq E$ is a* SESE (Single-Exit-Single-Entry) *of graph $G = (V, E)$ iff G_S has exactly two boundary nodes: one entry and one exit. A SESE is* trivial *if it is composed of a single edge. S is a* canonical *SESE of G if it does not partially overlap with any other SESE of G, i.e., given any other SESE S' of G, they are nested ($S \subseteq S'$ or $S' \subseteq S$) or they are disjoint ($S \cap S' = \emptyset$). By definition, the source of a WF-net is an entry to every fragment it belongs to and the sink of the net is an exit from every fragment it belongs to.*

The decomposition based on canonical SESEs is a well studied problem in the literature, and can be computed in linear time. In [18], the authors proposed the algorithm for constructing the *Refined Process Structure*

[1] $\Pi(R) = \bigcup_{(a,b) \in R} \{a, b\}$ is the set of elements referred to by relation $X \subseteq A \times B$.



(a) workflow net

(b) workflow graph and SESE descomposition

(c) RPST

Figure 1: A WF-net, its workflow graph and the RPST and SESE decomposition.

Tree (RPST), i.e., a hierarchical structure containing all the canonical SESEs of a model. In [10], the computation of the RPST is considerably simplified and generalized by introducing a pre-processing step that reduces the implementation effort considerably.

Definition 12 (RPST [10]) *Let G be the workflow graph of a given WF-net.*[2] *The* Refined Process Structured Tree (RPST) *of G is the tree composed by the set of all its canonical SESEs, such that, the parent of a canonical SESE S is the smallest canonical SESE that contains S. The root of the tree is the entire graph, and the leaves are the trivial SESEs. The set of all the nodes of the tree is denoted as $\mathbb{S}$.*

In the remainder of the paper, we will refer to canonical SESEs resulting from the RPST decomposition simply as SESEs. Also note that the SESEs are defined as a set of edges (i.e., S) over the workflow graph (not as subgraphs, i.e., G_S). However, for simplicity and when the context is clear, we will use the term SESE to refer also to the subgraph induced by those edges. We will denote as $PN^S = (P^S, T^S, A^S)$ the Petri net determined by the SESE S, i.e., $PN^S = (P \cap \Pi(S), T \cap \Pi(S), A \cap S)$. The nodes (either transitions or places) determined by S are denoted as N^S, i.e., $(P \cup T) \cap \Pi(S)$.

2.4. Running Example

Conformance checking techniques investigate how well an event log L and a system net SN fit together. Note that SN may have been discovered through process mining or may have been made by hand. In any case, it is interesting to compare the observed example behavior in L with the potential behavior of SN.

As indicated before there are four quality dimensions for comparing model and log (fitness, simplicity, precision, and generalization), but here we focus on fitness. A model with good fitness allows for most of the behavior seen in the event log. We use the model shown in Fig. 2 to illustrate the notion of fitness and the role of alignments when comparing modeled and observed behavior.

The model in Fig. 2 was inspired by a similar model presented in [19] and represents the possible situations to handle claims in a insurance company. In the following, we will use the letters in each transition to refer to the corresponding activity, e.g., a refers to the "register claim" event.

Consider the following event log $L_1 = [\sigma_1 = abdfgehmnpqs, \sigma_2 = abijlmnpqnpqs]$. An *optimal alignment* between a log trace and a model is a pair of traces denoting what is the best way the log trace can be reproduced by the model. For trace σ_1 of L_1, the optimal alignment is:

a	b	d	f	g	e	h	m	n	p	q	s
a	b	d	f	g	e	h	m	n	p	q	s

The top row of each alignment corresponds to "moves in the log" and the bottom row correspond to "moves in the model". If a move in the model cannot be mimicked by a move in the log, then a "$\gg$" ("no move") appears in the top row. The symmetric situation (a move in the log that cannot be mimicked by a move in the model) can also happen and is denoted analogously. Any of the two aforementioned situations reveal fitness problems between the model and the log trace. When both log and model can execute the same activity (in other words, they move synchronously), it denotes a fitting step between log and model. Since the optimal alignment for

[2]Although the approach presented in this paper can be generalized to graphs with several sources and sinks, for the sake of clarity in this paper we restrict to the case with only one source and only one sink [10].

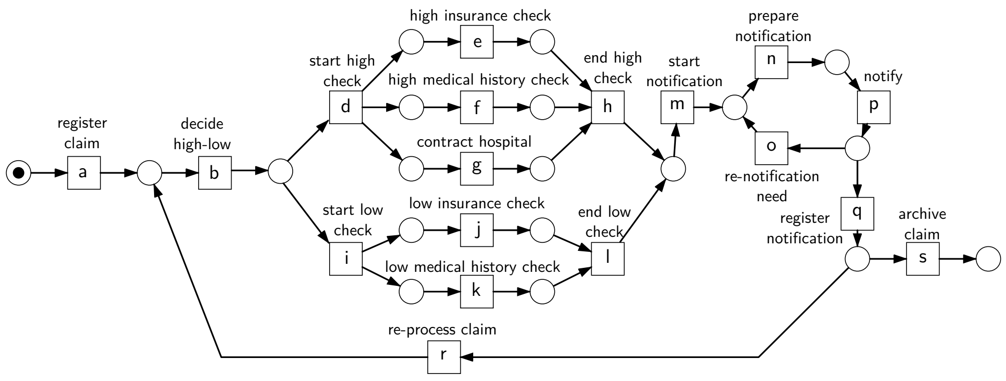


Figure 2: Running example: claims in a insurance company.

trace σ_1 has only fitting steps, the trace is fitting. An optimal alignment for trace σ_2 is:

a	b	i	j	$\gg$	l	m	n	p	$\gg$	q	n	p	q	s
a	b	i	j	k	l	m	n	p	o	$\gg$	n	p	q	s

This optimal alignment for trace σ_2 has several fitness problems represented by asynchronous moves of log or model. For instance, when the claim is low, the model considers it mandatory to check the medical history, while in reality this step did not happen, which is manifested in the alignment above by an asynchronous model move in the fifth position of the alignment: $(\gg, k)$. On the other hand, the notification cannot be registered if there are still pending notifications: the asynchronous log move in the eleventh position of the alignment $(q, \gg)$ reveals this situation.

In general, costs can be assigned to the different types of misalignments and global or individual fitness values can be defined and computed [6]. For the example above, and assuming that each misalignment has equal cost (say 1), the fitness of the model with respect to the log for trace σ_1 is 1.0 (perfect fitness), while the fitness for trace σ_2 is 0.9.

3. Decomposing Conformance Checking using SESEs

It is well known that checking conformance of large logs and models is a challenging problem. The size of log and model and the complexity of the underlying process strongly influence the time needed to compute fitness and to create optimal alignments. *Divide-and-conquer* strategies are a way to address this problem [11, 12]. As indicated before, we do not just want to partition the traces in the event log (providing a trivial way to distribute conformance checking). The potential gains are much higher if also the model is decomposed and traces are split into smaller ones. To decompose conformance checking problems, the overall system net *SN* is broken down into a collection of subnets $\{SN^1, SN^2, \ldots SN^n\}$ such that the union of these subnets yields the original system net.

Definition 13 (Decomposition) *Let* $SN = (PN, M_{ini}, M_{fin})$ *be a system net where* $PN = (P, T, A)$*.* $D = \{SN^1, SN^2, \ldots SN^n\}$ *is a decomposition of SN if and only if:*

- $P = \bigcup_{1 \leq i \leq n} P^i$,
- $T = \bigcup_{1 \leq i \leq n} T^i$,
- $A = \bigcup_{1 \leq i \leq n} A^i$ *where* $A^i \cap A^j = \emptyset$ *for* $1 \leq i < j \leq n$.

Note that each place or transition can be shared among different subnets, while each arc resides in just one subnet.

Any decomposition that satisfies Def. 13 may be considered for decomposing a conformance problem, e.g., subnets containing only one arc, or subnets randomly grouping distant nodes on the net. However, given that the ultimate goal of a decomposition is to be able to diagnose, comprehend and understand conformance problems, the use of meaningful decompositions is preferred. In [11] *SESEs* fragments were used to decompose conformance: given the structure of a SESE where a unique single entry and a unique single exit exist, a SESE becomes an appropriate unit of decomposition. Intuitively, each SESE may represent a subprocess within the main process (i.e., the interior nodes are not connected with the rest of the net) , and the analysis of every SESE can be performed independently. The *RPST* of a net can then be used to select a possible set of SESEs forming a decomposition. As it shown in Prop. 1, any *transverse cut* over the RPST defines a decomposition.

Proposition 1 (SESE decomposition) *Let SN* =

($WF\text{-}net, M_{ini}, M_{fin}$) be the system net of the workflow net $WF\text{-}net = (P, T, A, start, end)$. Consider the RPST decomposition of WF-net, where $\mathbb{S}$ represents all the SESEs in the RPST. We define a transverse-cut *over the RPST as a set of SESEs $\mathbb{D} \subseteq \mathbb{S}$ such that any path from the root to a leaf of RPST contains one and only one SESE in $\mathbb{D}$. Given a transverse-cut $\mathbb{D} = \{S_1, S_2, \ldots S_n\}$, let the decomposition $D_{\mathbb{D}}$ be defined as $D_{\mathbb{D}} = \{SN^{S_1}, SN^{S_2}, \ldots SN^{S_n}\}$, where $SN^{S_i} = (PN^{S_i}, M_{ini}\restriction_{P^{S_i}}, M_{fin}\restriction_{P^{S_i}})$, i.e., the Petri net determined by the SESE S_i, and the projection of the initial and final markings on the places of the subnet. The decomposition $D_{\mathbb{D}}$ derived from the SESEs satisfies the definition of decomposition given in Def. 13*

Proof. By definition of RPST, the arcs of each SESE in the RPST are contained in one, and only one, of its children (unless it is a trivial SESE). Therefore, any transverse-cut set of SESEs contains all the arcs, where each arc only appears in only one SESE. □

Proposition 2 (A SESE decomposition from RPST exists) *Given any RPST, a decomposition always exists.*

Proof. Given any RPST, the root (i.e., the whole net) defines a decomposition. In addition, the set of all the leaves (i.e., the trivial SESEs with only one arc) also defines a decomposition. □

As it is claimed in Prop. 2, the overall net is, by definition, a decomposition by itself. But it is obvious to see that this trivial way of decomposition does not alleviate the initial conformance problem. On the other hand, a decomposition formed only by trivial SESEs will produce meaningless components, and at the same time, the posterior analysis will have to deal with the analysis overhead produced by the creation of the numerous components. A decomposition which lays in between the aforementioned extremes seems more interesting from the practical point of view, i.e., to generate components large enough to become meaningful subprocesses, but whose size can be handled in practice. Hence, the algorithm proposed in Alg. 1 can generate a decomposition which limits the maximum size of each component to k in order to control the size and complexity of individual components.

Algorithm 1 shows how to compute a k-decomposition, for any k such that $1 \leq k \leq |A|$ (where $|A|$ stands for the number of arcs of the overall net). The algorithm keeps a set of nodes that conform the decomposition (D) and a set of nodes to consider (V). Initially V contains the root of the RPST, i.e., the overall net. Then, the algorithm checks, for each node v to consider, if v satisfies the k property, i.e., the number of arcs of SESE v is less or equal than k. If this is the case, v is included in the decomposition. If not, it discards v and includes the RPST children of v into the nodes to consider. Note that, given any RPST, a k-decomposition always exists, i.e., in worst case, the decomposition formed by all the leaves of the RPST will satisfy the definition. The algorithm proposed has linear complexity with respect to the size of the RPST, and termination is guaranteed by the fact that the size of the component is reduced in every iteration.

Algorithm 1 k-decomposition algorithm
```
procedure k-DEC(RPST,k)
    $V = \{root(RPST)\}$
    $D = \emptyset$
    while $V \neq \emptyset$ do
        $v \leftarrow pop(V)$
        if $|v.arcs()| \leq k$ then $D = D \cup \{v\}$
        else $V = V \cup \{children(v)\}$
    return $D$
```

A SESE is a component that only interfaces with the rest of the net through the single entry and single exit boundary nodes, which may be shared among different components. The rest of nodes of a SESE (i.e., the interior nodes) have no connection with other components. Given that the SESE computation is performed over the workflow graph (i.e., where there is no distinction between places and transitions), we distinguish two possible cases for the boundary nodes: *transition boundary* and *place boundary*.

The transition boundary case occurs when the node determined to be the entry or the exit of a SESE is a transition. Figure 3 shows an example of a transition boundary. In the example, the overall net is decomposed into two subnets that correspond to the SESEs S_1 and S_2, being d the boundary transition shared between them.

As it is proven in [13], a *transition boundary* decomposition represents no problem from a conformance point of view, i.e., given a decomposition with only transition boundaries, a log trace fits the overall net if and only if it fits all the subnets. The reason for that is that when a transition is shared among subnets, the label of the transition is used to synchronize the subnets that contain that transition on their boundaries, ensuring that the decisions on model's ability to reproduce that label are done jointly. Consider the decomposition $D_{\mathbb{D}} = \{SN^{S_1}, SN^{S_2}\}$ from the example of Fig. 3, where $SN^{S_1} = (PN^{S_1}, [start], [])$ and $SN^{S_2} = (PN^{S_2}, [], [end])$

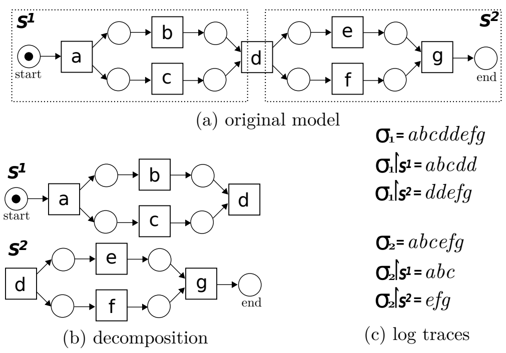


Figure 3: Example of decomposition with transition boundary.

are the systems nets derived from the SESEs S_1 and S_2. Consider the trace $\sigma_1 = abcddefg$ shown in Fig. 3c. Such trace does not fit the overall net due to the double d. The projection of that trace on SN^{S_1} and SN^{S_2} results in $\sigma_1\upharpoonright_{T^{S_1}} = abcdd$ and $\sigma_1\upharpoonright_{T^{S_2}} = ddefg$ respectively (cf. Fig. 3c). Note that, although $\sigma_1\upharpoonright_{T^{S_2}}$ fits SN^{S_2} (on SN^{S_2}, the preset of d is empty hence it can fire more than once), $\sigma_1\upharpoonright_{T^{S_1}}$ does not fit SN^{S_1}. Hence, the trace σ_1 that does not fit the overall net, does not fit all the subnets (at least there is one that is not fitting). A similar situation happens with the trace $\sigma_2 = abcefg$ (where no d appears), i.e., trace σ_2 does not fit the overall net, hence $\sigma_2\upharpoonright_{T^{S_1}}$ does not fit SN^{S_1} or $\sigma_2\upharpoonright_{T^{S_2}}$ does not fit SN^{S_2}. In the latter example, actually both do not fit.

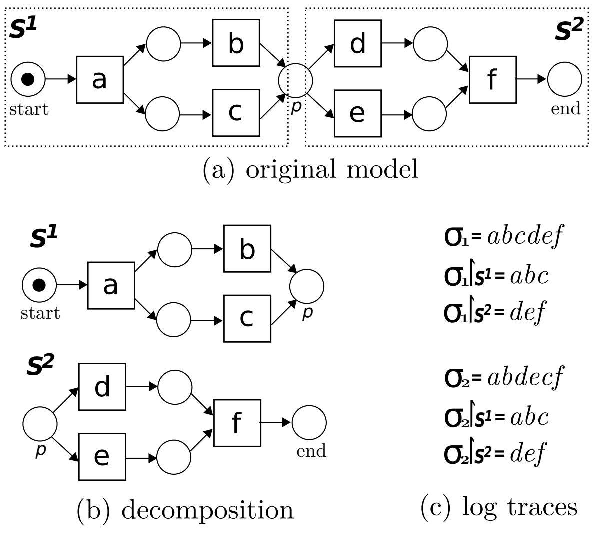


Figure 4: Example of decomposition with place boundary.

On the other hand, the case of *place boundary* is different. When the boundary (entry or exit) is a place, it is shared among two or more subnets. However, the arcs connected to the place (the ones in charge of producing and consuming tokens) are split among the subnets. This makes the place unable to synchronize, and therefore, it is impossible to analyze the different subnets in isolation. The example in Fig. 4 reflects this situation. The original net is decomposed into two subnets, $D_{\mathbb{D}} = \{SN^{S_1}, SN^{S_2}\}$, corresponding with the SESEs S_1 and S_2, and being p the boundary place shared by both subnets. It can be seen that the arcs that produce tokens in p and the ones that consume tokens from p are distributed into different subnets. Consider now the log traces $\sigma_1 = abcdef$ and $\sigma_2 = abdecf$ of Fig. 4. While σ_1 fits the overall net, σ_2 does not. However, the projections of both traces on T^{S_1} and T^{S_2} are the same (cf. Fig. 4). This problem materializes when we analyze the subnets. Firstly, given that any arc that produces tokens in p is contained in PN^{S_1}, we need to consider an initial marking for SN^{S_2} different than [] (otherwise, the subnet would be deadlocked initially). If we consider the initial marking $[p]$, $\sigma_1\upharpoonright_{T^{S_2}}$ does not fits SN^{S_2}. Therefore the fitness correctness is not preserved, i.e., a trace that fits the overall net like σ_1 must fit all the subnets. On the other hand, if we consider the initial marking with two (or more) tokens on p (i.e., $[p^2]$), $\sigma_2\upharpoonright_{T^{S_2}}$ fits SN^{S_2} (similarly, $\sigma_2\upharpoonright_{T^{S_1}}$ fits SN^{S_1}). However σ_2 is a non-fitting trace of the overall net, and consequently, it must not fit all the subnets. Therefore, when the decomposition contains place boundaries, the preservation of the fitness correctness is not guaranteed.

In [13] the definition of decomposition is revisited to propose the so called *valid decomposition*, i.e., a decomposition that only shares transitions (but not places nor arcs).

Definition 14 (Valid Decomposition [13]) *Let $SN = (PN, M_{ini}, M_{fin})$ be a system net where $PN = (P, T, A)$. $D = \{SN^1, SN^2, \ldots SN^n\}$ is a* valid *decomposition of SN if and only if:*

- $T = \bigcup_{1 \le i \le n} T^i$,
- $P = \bigcup_{1 \le i \le n} P^i$ *where* $P^i \cap P^j = \emptyset$ *for* $1 \le i < j \le n$,
- $A = \bigcup_{1 \le i \le n} A^i$ *where* $A^i \cap A^j = \emptyset$ *for* $1 \le i < j \le n$.

In [13, Theorem 2] it is proven that all valid decomposition preserves the fitting correctness, i.e., a log is fitting a system net if and only if fits all the subnets.

As has been illustrated in the previous examples, a decomposition based directly on SESEs is not necessarily a valid decomposition, i.e., boundary places may be shared among subnets. However, in the remainder of this section an approach to transform a SESE decomposition into a valid decomposition is presented, which tries to preserve the underlying semantics behind the SESE decomposition. This technique is called *bridging*, and consists of: (1) transforming each place boundary found into a transition boundary (i.e., boundary place is removed) and (2) creating explicit subnets (called *bridges*) for each boundary place. The bridges contain all the transitions connected with the boundary place, and they are in charge of keeping the place synchronized among subnets. In addition, the boundary places together with the arcs connected to them are removed from the original subnets. Formally:

Definition 15 (Bridging a SESE decomposition)
Let $\mathbb{D} = \{S_1, \dots S_n\}$ *be SESE decomposition of the WF-net* $(P, T, A, start, end)$*. Let* $I_{\mathbb{D}} = \{i_1, \dots, i_n\}$ *and* $O_{\mathbb{D}} = \{o_1, \dots, o_n\}$ *be the set of all entry and exit nodes of the SESEs in* $\mathbb{D}$*.* $B = \{p_1, \dots, p_k\} = ((I_{\mathbb{P}} \cup O_{\mathbb{P}}) \cap P) \setminus \{start, end\} = (I_{\mathbb{P}} \cap O_{\mathbb{P}}) \cap P$ *is the set of boundary places, i.e., entry and exit nodes of the SESEs that are places but not the source or sink place of the WF-net WN. The decomposition after applying* bridging $\mathbb{D}' = \{S'_1, \dots S'_n, B_1 \dots B_k\}$ *of* $\mathbb{D}$ *is constructed as follows:*

- *For all* $1 \le i \le n$*:* $S'_i = \{(x, y) \in S_i \mid \{x, y\} \cap B = \emptyset\}$ *(boundary places are removed from the SESEs).*
- *For* $1 \le j \le k$*:* $B_j = \{(x, y) \in A \mid p_j \in \{x, y\}\}$ *(bridges are added).*

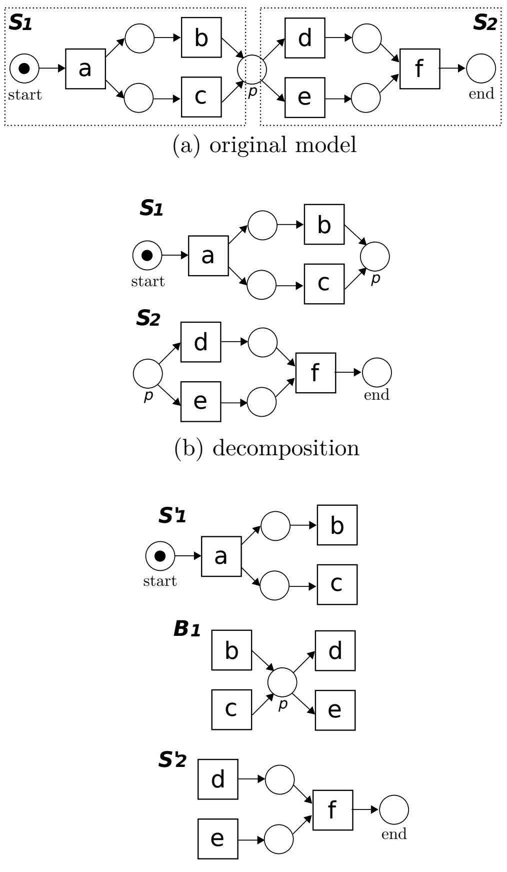
$D_{\mathbb{D}'} = \{SN^{S'_1}, \dots SN^{S'_n}, SN^{B_1} \dots SN^{B_k}\}$ *represents the decomposition constructed from* $\mathbb{D}'$*.*

Figure 5 illustrates the effects of the bridging on the example previously shown in Fig. 4. In this case, the boundary place p (and its arcs) are removed from S_1 and S_2, and a bridge B_1 is created. Note that now, the transitions connected to p (i.e., b, c, d and e) are shared (instead of p), keeping the synchronization among components, and making $D_{\mathbb{D}'}$ a valid decomposition.

Proposition 3 shows that the decomposition derived from applying SESE decomposition and then bridging results in a valid decomposition, according to Def. 14.

Proposition 3 (Bridging results in valid decomposition)
Let $\mathbb{D}' = \{S'_1, \dots S'_n, B_1 \dots B_k\}$ *be obtained from a SESE decomposition after applying bridging. The decomposition* $D_{\mathbb{D}'} = \{SN^{S'_1}, \dots SN^{S'_n}, SN^{B_1} \dots SN^{B_k}\}$ *is a valid decomposition according to Def. 14.*

Proof. By construction, a SESE decomposition only shares transitions and places. After applying the bridging, all the shared places are removed, creating explicit components with only one instance of these places. □



(a) original model

(b) decomposition

(c) decomposition and bridging

Figure 5: Example of decomposition with bridging.

Moreover, given that the bridging produces a valid decomposition, it also preserves the property that a trace in the log fits the overall process model if and only if each subtrace fits the corresponding process fragment. Hence, fitness checking can be decomposed using SESEs and bridges.

Proposition 4 (Fitness Checking can be decomposed) *Let L be a log and* $SN = (WF\text{-}net, M_{ini}, M_{fin})$ *be a system net where WF-net is a workflow net. Let* $D_{\mathbb{D}'} = \{SN^1, SN^2, \dots SN^n\}$ *be a valid decomposition resulting of the application of the SESE decomposition and bridging over WF-net. Let* $SN^i = (PN^i, M^i_{ini}, M^i_{fin})$*, where* $PN^i = (P^i, T^i, A^i)$*.*

A trace $\sigma \in L$ fits *SN (i.e.,* $(WF\text{-}net, M_{ini})[\sigma\rangle(WF\text{-}net, M_{fin})$*) if and only if it fits all the parts, i.e., for all* $SN^i \in D_{\mathbb{D}'}$*,* $(PN^i, M^i_{ini})[\sigma\upharpoonright_{T^i}\rangle(PN^i, M^i_{fin})$*.*

Proof. Special case of the more general Theorem 2 in [13]. If the overall trace σ fits SN, then each of the pro-

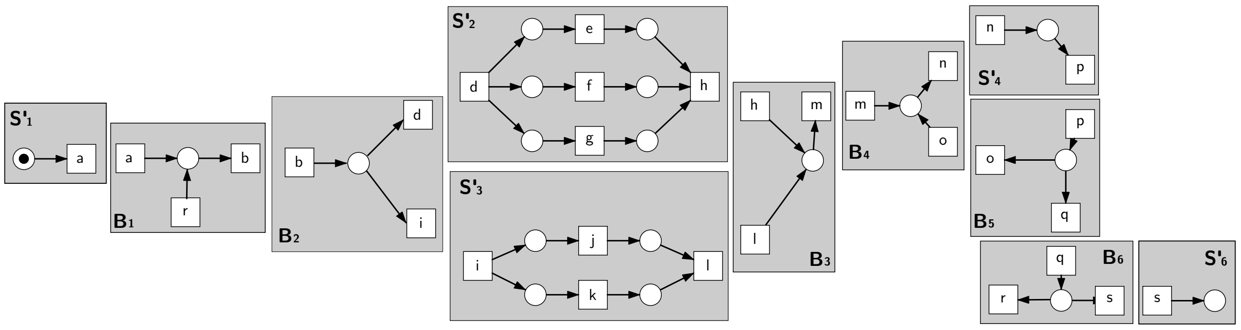


Figure 6: Components resulting from 15-decomposition and bridging for the running example of Fig. 2.

jected traces $\sigma\restriction_{T^i}$ fits the corresponding subnet. If this is not the case, then at least there exist one projected trace $\sigma\restriction_{T^i}$ that does not fit. But this is impossible because, by construction, each subnet is a relaxation of the behavior of the overall net. If the projected traces $\sigma\restriction_{T^i}$ fit the corresponding subnets, then these traces can be stitched back into a trace σ that fits SN. □

Going back to the running example of this paper (Fig. 2), Fig. 6 shows the SESE decomposition using a k of 15. Let us show how the fitness problems are now identified in a decomposed manner. For that, we will use the trace σ_2 = *abijlmnpqnpqs* from the log L_1 in Sect. 2.4. Given σ_2 and each one of the SESEs provided in Fig. 6, the only ones that reveal fitness anomalies are S'_3, B_4 and B_6 (for the other components we can find perfect alignments when projecting σ_2 to the activities of the component). The alignment for S'_3 is:

i	j	$\gg$	l
i	j	k	l

which reveals that the mandatory check of the medical history is missing in the log. Analogously, the alignment for B_4 is:

m	n	$\gg$	n
m	n	o	n

that identifies the explicit need for notifying again the client, an action missing in the log but required by the model. Finally, the alignment for B_6:

q	q	s
$\gg$	q	s

reveals another fitness problem for trace σ_2: the system has stored in the log an early registration of the notification which was not meant at that point in time, since later notifications were sent and according to the model, the registration is only expected to be done at the end of the case.

Using Prop. 4, we can infer that the fact that some components in the decomposition identify fitness problems implies that the whole model does not fit log L_1.

3.1. Decomposing with Invisible/Duplicates

So far, the approach presented in this paper was assuming that all the Petri net transitions were associated with a unique single activity, i.e., a transition could be unambiguously identified by its label. In this section we lift this assumption in order to consider invisible and duplicate transitions. An invisible transition is a transition without activity associated, e.g., transitions included for routing purposes. Duplicate transitions are transitions with the same activity associated. For example, consider the net of Fig.7, which is a slight variation of the running example of Fig. 2. This model contains an invisible transition (represented in black) used to *skip* the execution of *contract hospital*, i.e., now *contract hospital* is optional. Moreover, the new model does not distinguishes between *high insurance check* and *low insurance check*, but the same action *insurance check* is modeled in two different parts of the model, i.e., is a duplicate activity.

The definition of Petri net presented in the preliminaries is now adapted to include the possibility of invisible and duplicate transitions.

Definition 16 (Labeled Petri Net) *A labeled Petri net $PN = (P,T,A,l)$ is a Petri net (P,T,A) with labeling function $l \in T \nrightarrow \mathcal{U}_A$ where $\mathcal{U}_A$ is some universe of activity labels. If a transition $t \notin dom(l)$, it is called invisible. $T_v(PN) = dom(l)$ is the set of* visible transitions *in PN. $T_v^u(PN) = \{t \in T_v(PN) \mid \forall_{t' \in T_v(PN)}\ l(t) = l(t') \Rightarrow t = t'\}$ is the set of* unique *visible transitions in PN (i.e., there are no other transitions having the same visible label).*

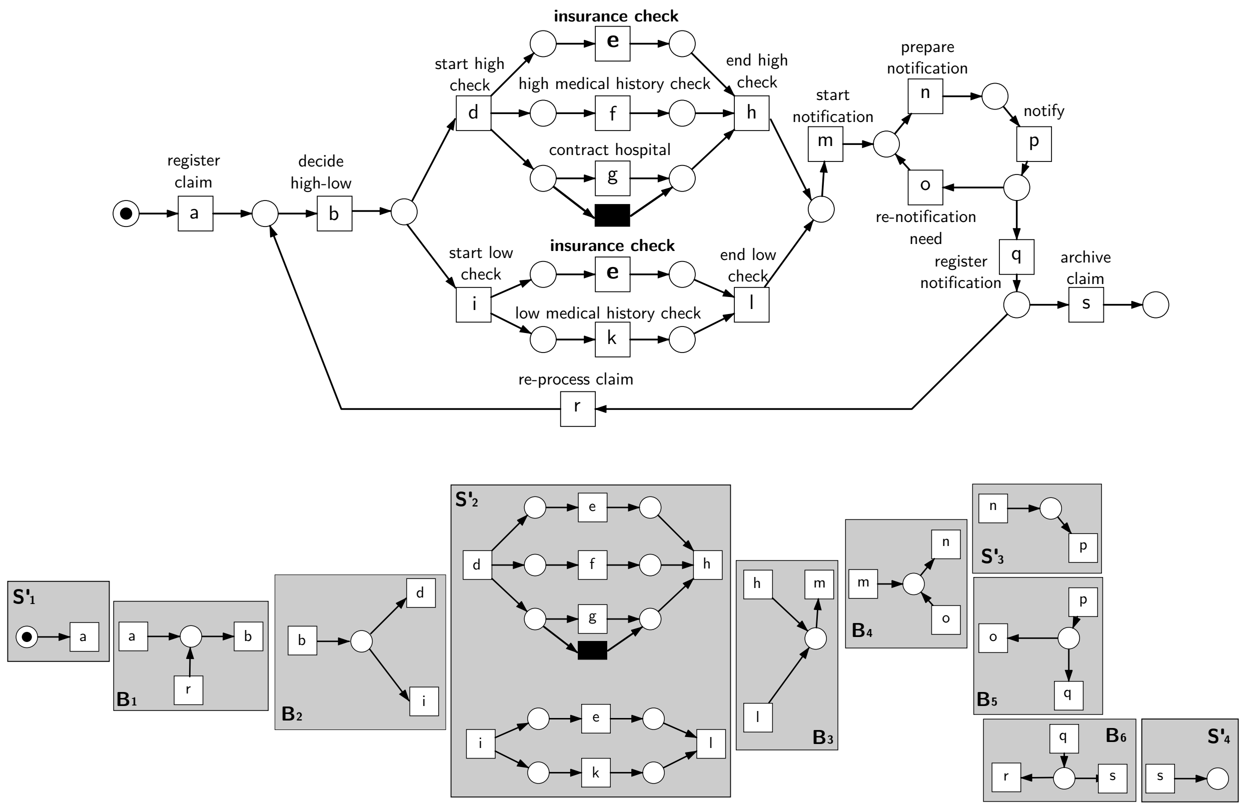


Figure 7: Variant of the running example of Fig. 2 including invisible and duplicates (top), and its corresponding decomposition (bottom).

As it has been illustrated previously in this paper, when a net is decomposed, the labels of the transitions are used to synchronize and preserve the fitness properties. However, sharing invisible and duplicate transitions among subnets generates ambiguity invalidating this synchronization. Thus, the definition of valid decomposition presented in Def.14 is refined to consider invisible and duplicates, i.e., only unique visible transitions can be shared among subnets.

Definition 17 (Valid Decomposition with Invisible and Duplicates[13]) *Let $SN = (PN, M_{ini}, M_{fin})$ be a system net where $PN = (P, T, A, l)$. $D = \{SN^1, SN^2, \dots SN^n\}$ is a* valid *decomposition of SN if and only if:*

- *$SN^i = (PN^i, M^i_{ini}, M^i_{fin})$ is a system net with $PN^i = (P^i, T^i, A^i, l^i)$ for all $1 \leq i \leq n$,*
- *$l^i = l\restriction_{T^i}$ for all $1 \leq i \leq n$,*
- *$P^i \cap P^j = \emptyset$ for $1 \leq i < j \leq n$,*
- *$T^i \cap T^j \subseteq T^u_v(SN)$ for $1 \leq i < j \leq n$, and*
- *$SN = \bigcup_{1 \leq i \leq n} SN^i$.*

Let $SN = (PN, M_{ini}, M_{fin})$ with $N = (P, T, A, l)$ be a system net with valid decomposition $D = \{SN^1, SN^2, \dots, SN^n\}$. We can observe the following properties:

- each place appears in precisely one of the subnets, i.e., for any $p \in P$: $|\{1 \leq i \leq n \mid p \in P^i\}| = 1$,
- each invisible transition appears in precisely one of the subnets, i.e., for any $t \in T \setminus T_v(SN)$: $|\{1 \leq i \leq n \mid t \in T^i\}| = 1$,
- visible transitions that do not have a unique label (i.e., there are multiple transitions with the same label) appear in precisely one of the subnets, i.e., for any $t \in T_v(SN) \setminus T^u_v(SN)$: $|\{1 \leq i \leq n \mid t \in T^i\}| = 1$,
- visible transitions having a unique label may appear in multiple subnets, i.e., for any $t \in T^u_v(SN)$: $|\{1 \leq i \leq n \mid t \in T^i\}| \geq 1$, and
- each edge appears in precisely one of the subnets, i.e., for any $(x, y) \in A$: $|\{1 \leq i \leq n \mid (x, y) \in A^i\}| = 1$.

In order to instantiate a decomposition complying with this new definition of valid decomposition, Algorithm1 needs to be refined (cf. Alg. 2).

Algorithm 2 checks if considering the children of a SESE s will violate the definition of valid decomposition in Def.17. The three conditions need to be satisfied:

- transitions shared (T) between any subset of SESEs

Algorithm 2 Refined k-decomposition algorithm

```
function k-DEC(RPST,k)
    $V = \{root(RPST)\}$
    $D = \emptyset$
    while $V \neq \emptyset$ do
        $v \leftarrow pop(V)$
        if $|v.arcs()| \leq k$ or not $Decomposable(v)$ then
            $D = D \cup \{v\}$
        else $V = V \cup \{children(v)\}$
    return $D$

function DECOMPOSABLE($s$)
    $\{s_1, \dots s_n\} \leftarrow children(s)$
    $T \leftarrow$ shared transitions in $\{s_1, \dots s_n\}$
    $P \leftarrow$ shared places in $\{s_1, \dots s_n\}$
    $T^P \leftarrow$ transitions connected with $P$

    if $T \cap T^u_v \neq T$ then return false
    else if $T^P \cap T^u_v \neq T$ then return false
    else if same label in different $\{s_1, \dots s_n\}$ then
        return false
    else return true
```

$\{s_1, \dots s_n\}$ must be unique visible transitions (T^u_v).

- places shared (P) between any subset of SESEs $\{s_1, \dots s_n\}$ will be *bridged* according to Def. 15. Therefore, transitions connected with the places shared (P) between any subset of $\{s_1, \dots s_n\}$ must be unique visible transitions (T^u_v), in order to avoid be duplicated boundary transitions after the bridging.
- Transitions with the same label must belong to the same v_i.

The main difference between the original k-decomposition algorithm presented previously and Alg. 2 is that the latter checks if considering the children of SESE v for the decomposition D will violate the valid decomposition definition (Def.17). This additional checking makes the algorithm quadratic in the number of edges, although our experiments show no noticeable increase in computation time. Notice that by definition, if the children $\{s_1, \dots s_n\}$ of v violate the definition, considering further descendants of v will also violate the definition. Therefore, when the algorithm checks that the SESE must not be decomposed, it includes it into the decomposition D. As a result, Algorithm 2 does not guarantee the k property, i.e., some components may have more than k arcs. For instance, consider the subnets resulting of a 15-decomposition and bridging shown in Fig.7. Unlike Fig.6, here when the algorithm tries to decompose the SESE S_2, it detects than this will result in splitting the duplicate e, and thus it must consider S_2, even if the number of arcs of S_2 is greater 15[3]. Notice that some worst case scenarios exist for Alg. 2: consider the example of Fig.8. In this case, the presence of invisible transitions in the model boundaries makes it impossible for the algorithm decompose more that the root S_1, and therefore, the resulting decomposition will be the overall net. The effect of those cases can be alleviated by pre-processing the model and the log before applying the decomposed conformance.

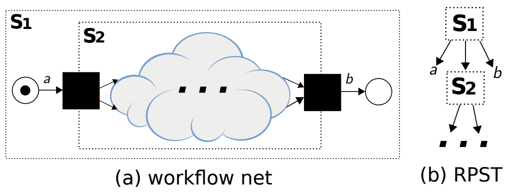


Figure 8: Example of worst case scenario for the k-decomposition with invisible/duplicates.

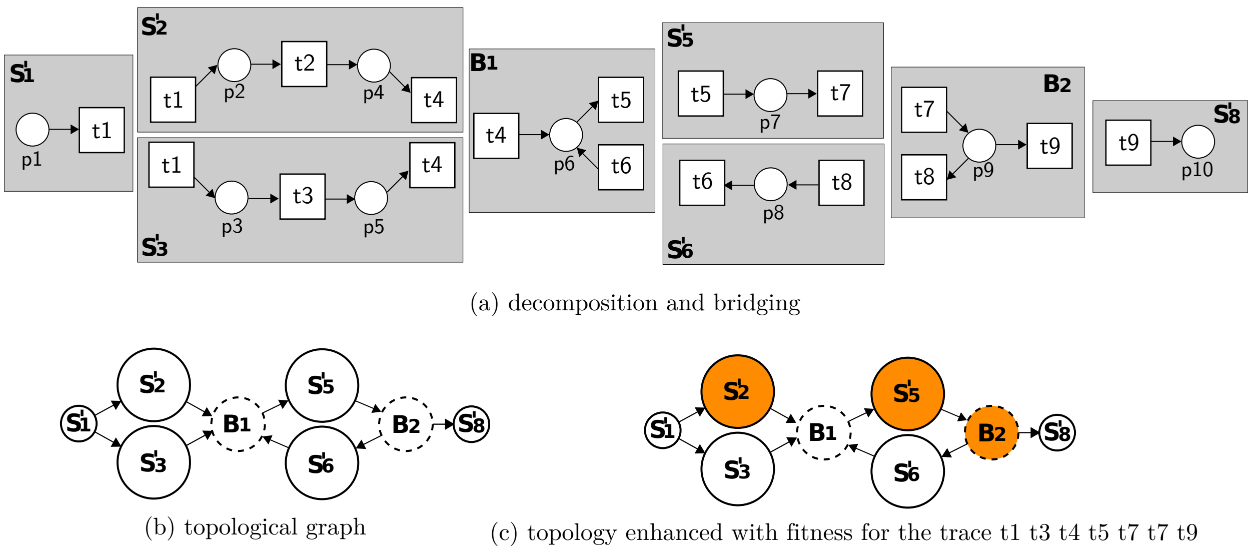
4. Topology of a Decomposition

A valid decomposition is a collection of subnets that may be related to each other through the sharing of transitions, i.e., two subnets are related if they share a transition. The *topology* of a valid decomposition is an undirected graph where the vertices denote subnets and the edges denote the sharing of transitions.

Definition 18 (Topology of a Decomposition) *Let* $D = \{SN^1, SN^2, \dots SN^n\}$ *be a valid decomposition, where* $SN^i = (PN^i, M^i_{ini}, M^i_{fin})$ *and* $PN^i = (P^i, T^i, A^i)$*. The* topology *of decomposition* D *is defined as the undirected graph* $T_D = (D, C)$ *such that two components are connected if they share any transition, i.e.,* $C = \{\{SN^i, SN^j\} | 1 \leq i < j \leq n \wedge T^i \cap T^j \neq \emptyset\}$.

In the general definition of topology over a valid decomposition the relations remain undirected, i.e., two subnets sharing the same transition are connected by an undirected edge. However, in the specific case of a valid decomposition derived from SESEs, this definition can be extended to include the concept of direction: the transition being the exit of the SESE is considered the

[3]Notice that, after the bridging process, a fragment may lose its SESE structure, e.g., the entry and exit places of S_2 are removed when it becomes S'_2 due to the bridges B_2 and B_3. In spite of this, the decomposition obtained still satisfies Def. 17. Moreover, although the entry or exit place has been removed explicitly from the graph, the fragment still represents its corresponding subprocesses.



(a) decomposition and bridging

(b) topological graph

(c) topology enhanced with fitness for the trace t1 t3 t4 t5 t7 t7 t9

Figure 9: Example of valid decomposition and its topology

source of the edge, while the entry is the target. Bridges can have multiple entry and exit nodes, but again we can derive the direction connections among bridges and SESEs.

Definition 19 (Topology of a SESE Decomposition) *Let $\mathbb{D} = \{S_1, \dots S_n\}$ and $\mathbb{D}' = \{S'_1, \dots S'_n, B_1, \dots B_k\}$ be a SESE decomposition before and after applying bridging. Let $\{p_1, \dots, p_k\}$ be the boundary places in $\mathbb{D}$. Let $D_{\mathbb{D}'} = \{SN^{S'_1}, \dots SN^{S'_n}, SN^{B_1} \dots SN^{B_k}\}$ represent the decomposition constructed from $\mathbb{D}'$. The* topology *of $D_{\mathbb{D}'}$ is defined as the directed graph $T_{D_{\mathbb{D}'}} = (D_{\mathbb{D}'}, C)$ such that $C = \{(SN^{S'_i}, SN^{S'_j}) | 1 \leq i, j \leq n \wedge (y, x) \in S_i \wedge (x, z) \in S_j\} \cup \{(SN^{S'_i}, SN^{B_j}) | 1 \leq i \leq n \wedge 1 \leq j \leq k \wedge (y, p_j) \in S_i\} \cup \{(SN^{B_j}, SN^{S'_i}) | 1 \leq i \leq n \wedge 1 \leq j \leq k \wedge (p_j, y) \in S_i\}$.*

Note that the topological graph has as vertices the nets in $\mathbb{D}'$, but some arcs of this graph (those regarding connection to bridges) are defined over the original SESE decomposition $\mathbb{D}$, e.g., $(y, p_j) \in S_i$ refers to an arc in the original SESE and is used to infer a directed connection from $SN^{S'_i}$ to SN^{B_j}.

One of the features of the topology is to aid in the visualization of a valid decomposition. For example, let us consider the valid decomposition in Fig. 9 (a slight modification of the model in Fig. 1). The decomposition is the result of applying a 4-decomposition over the model of Fig. 9a (i.e., SESEs with at most 4 edges: $S'_1, S'_2, S'_3, S'_5, S'_6, S'_8$) and followed by the bridging (resulting in two bridges, B_1 and B_2, corresponding with the two boundary places p_6 and p_9)[4]. The corresponding topology is shown in Fig. 9b.

Besides simply showing the connections among subnets, the topology can be enhanced with other information about the components and their characteristics. For instance, bridges can be denoted by circles having dotted borders and SESEs can be denoted by circles having solid borders. Moreover, the size of the nodes in the graph is directly related with the size of the corresponding subnets, i.e., a subnet with many arcs is depicted using a larger circle compared to subnets with fewer arcs. Given the final goal of this paper (i.e., conformance analysis), a particular interesting case is to enhance the topology with conformance information. For example, consider the trace $\sigma = t_1 t_3 t_4 t_5 t_7 t_7 t_9$. When we check fitness in the subnets of decomposition $D_{\mathbb{D}'} = \{SN^{S'_1}, \dots SN^{S'_8}, SN^{B_1}, SN^{B_2}\}$, we detect the following fitness anomalies: in $SN^{S'_2}$, t_4 is fired without firing t_2; in $SN^{S'_5}$, t_7 is executed twice, but this requires the firing of t_5 also twice; finally, in the bridge SN^{B_2}, t_7 is fired twice, but t_9 only once, leaving a token remaining in p_9. This information can be used to enhance the topology of Fig. 9b. As shown in Fig. 9c the vertices have problems can be depicted in gray (here S'_2, S'_5 and B_2).

Although the topology is an important aid for the process diagnosis by itself, it can also guide further analy-

[4]Note that the original trivial SESE S_4 that corresponds to the arc $(t4, p6)$ has disappeared once the bridging has been done, i.e., the arc is now in B_1. The same happens for the original trivial SESE S_7 corresponding to the arc $(p9, t9)$.

sis. For instance, the topological graph enhanced with conformance information can be used to *identify maximal process fragments with fitness problems*. This allows us to focus on the problematic parts of a model, discarding the parts without any fitness problems. Algorithm 3 describes a procedure that is based on detecting connected components (C_c) on the graph induced by the non-fitting vertices. First, the topological graph (T_D) is filtered, leaving only non-fitting vertices (V). Then, the weakly connected components (C_c) are detected: 1) a random node v_1 is chosen, 2) all nodes $\{v_1, \ldots v_n\}$ weakly connected (i.e., connected vertices without considering the direction of the edges) with v_1 are computed using a depth-fist search exploration and they constitute a new connected component, and finally 4) $\{v_1, \ldots v_n\}$ are removed from the graph and the exploration of connected components continues. For each connected component, we project the elements of the original net they refer to. Note that this algorithm prioritizes the connectivity among vertices resulting in weakly connected components. However, alternative versions of the algorithm yielding strongly connected components are possible. For instance, given the example of Fig. 9c, two connected components are found as shown in Fig. 10: one corresponding to $SN^{S'_2}$ and the other to the union of $SN^{S'_5}$ and SN^{B_2}.

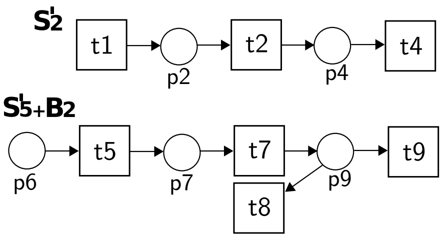


Figure 10: Examples of non-fitting weakly connected components.

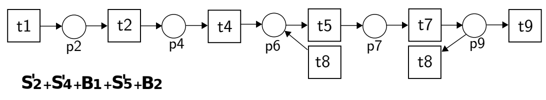


Figure 11: Example of a non-fitting subnet.

The topological graph enhanced with conformance information can also be used to create one complete subnet that includes all non-fitting subnets of the decomposition, i.e., a *connected* set of vertices V containing all the non-fitting vertices V_{nf}. Algorithm 4 illustrates the heuristic-based approach proposed, based on the greedy expansion of the largest non-fitting connected components, to compute the complete non-fitting subnet. Initially, V contains the non-fitting vertices V_{nf},

Algorithm 3 Non-Fitting Weakly Connected Components Algorithm

```
function NFWCC($T_D$,$V$)                ▷ $V$ is non-fitting vertices
    $C_c = \emptyset$
    remove from $T_D$:                    ▷ Graph induced by $V$
    -all arcs $c = \{x, y\}$ such that $x, y \notin V$
    -all vertices $z \notin V$

    while $T_D$ has vertices do
        $v_1 \leftarrow$ select random vertex on $T_D$
        $\{v_1, \ldots v_n\} \leftarrow$ get vertices weakly connected with $v_1$ using Depth-first search
        remove $\{v_1, \ldots v_n\}$ from $T_D$
        $C_c = C_c \cup \bigcup_1^n v_i$
    return $C_c$
```

and G denotes the graph induced by V. The goal of the algorithm is to have all the vertices in V connected, i.e. G be connected. If this is not the case, the algorithm detects the two largest connected components (c_1 and c_2) of G, and then computes the shortest path ($\{v_1 \ldots v_n\}$) between any vertex in c_1 and any vertex in c_2. Finally, $\{v_1 \ldots v_n\}$ are included to V, and it is checked again if the induced graph G is connected. Given the example of Fig. 9c, the net resulting (shown in Fig. 11) contains the union of the subnets $SN^{S'_2}$, $SN^{S'_4}$, SN^{B_1}, $SN^{S'_5}$ and SN^{B_2}.

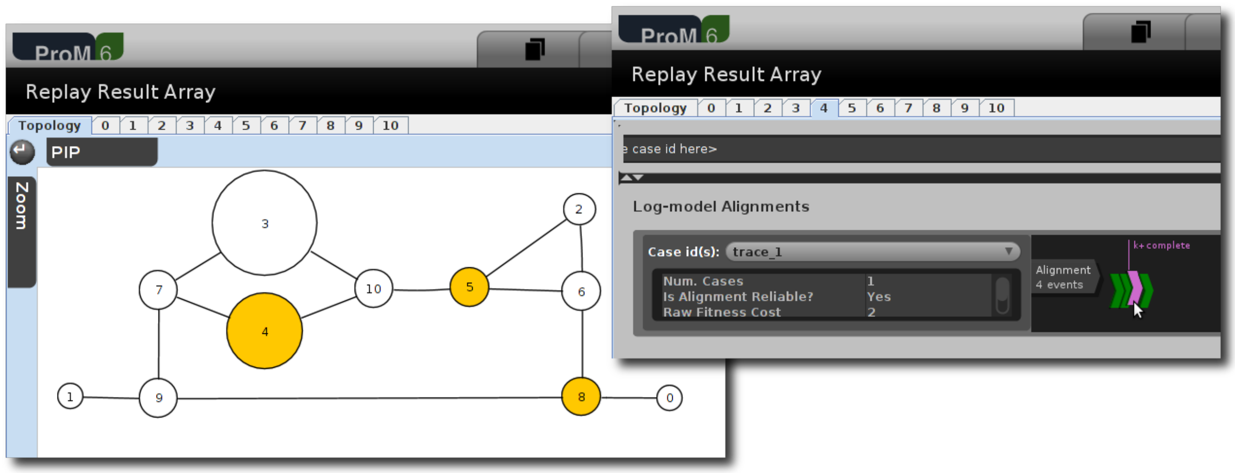
Algorithm 4 Non-Fitting Subnet Algorithm

```
function NFN($T_D$,$V_{nf}$)             ▷ $V_{nf}$ is non-fitting vertices
    $V \leftarrow V_{nf}$
    $G \leftarrow$ graph induced by $V$ on $T_D$
    while $G$ is not connected do
        $c_1 \leftarrow$ get the $1st$ largest conn. comp. of $G$
        $c_2 \leftarrow$ get the $2nd$ largest conn. comp. of $G$
        $\{v_1 \ldots v_n\} \leftarrow$ shortest_path_vertex($T_D, c_1, c_2$)
        $V = V \cup \{v_1 \ldots v_n\}$.
        $G \leftarrow$ graph induced by $V$ on $T_D$
    return Petri net induced by $V$
```

In Fig. 12, the topology for the running example is presented as a screenshot of the tool associated with this paper. In the figure, the alignment for the problematic component 4 (S'_3 in Fig. 6) is also shown.

5. Multi-level Analysis and its Applications

Thus far the analysis of the conformance was always performed using a complete decomposition of the



Case id(s): trace_1

Num. Cases	1
Is Alignment Reliable?	Yes
Raw Fitness Cost	2

Figure 12: Screenshot of the topology for the running example.

model. However, for detailed process diagnosis it is important to also be able to do a more focused analysis. This section presents three approaches to achieve this: (1) stand-alone checking, (2) multi-level analysis, and (3)filtering.

5.1. *Stand-alone Checking*

First we consider the problem of analyzing a selected subprocess in isolation. Clearly, assumptions on the subprocess and its context must be defined in order to perform such an isolated conformance check. The conformance results obtained are strongly correlated with the assumptions considered, and hence the analysis of the model properties and domain knowledge becomes an essential part, e.g., whether a place has a bound on the number of tokens, or the number of activations of the subprocess within a trace.

Let us show an application of the stand-alone checking for the typical case of well-structured process models, that can easily be modeled using the subclass of *safe workflow nets* (for formal details see [11]). Given a SESE S obtained from a decomposition, one can apply the following steps to conduct a local diagnosis of S:

1. *Workflowing the SESE:* In order to have a clear starting and ending point for the subprocess represented, re-define the net derived from S. In other words, given a SESE S, define the net derived from S in terms of a workflow net (cf. Def. 3), with a starting place (*start*) and a final place (*end*). By construction, a SESE has both an entry (i) and an exit (o) node. The *start* corresponds with i if i is a place. However, when i is a transition, we define *start* to be an artificial place and we connect it with i. Similarly for *end* and o.
2. *Initial and Final Marking:* Given the workflow-net from the previous step, determining a plausible initial marking becomes straightforward, i.e., due to the safeness assumption of safe workflow nets, we consider a single token in the *start* in order to *enable* the execution of the subprocess. Similarly for the final marking.
3. *SESE activations:* the number of potential activations of a SESE within a case must be determined. In case it is always one, the SESE is left as is. However, in case it can be executed more than once (e.g., the SESE is inside some loop in the model), the net in the previous step is *short-circuited*, using a silent transition between *end* and *start*. Finally, it can also happen that a SESE may be not executed in a trace. In this last case, a silent transition between *start* and *end* avoiding the SESE content will be used. Determining if a suprocess can be executed several times is a complex matter. In [11], it is proposed the use of Petri net structural theory (*minimal T-invariants* [20]) as a best effort strategy for estimating repetitive behavior.

5.2. *Multi-Level Analysis*

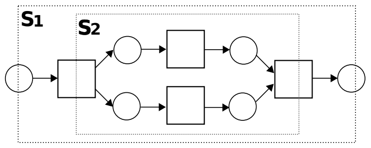
In this section we propose to combine the stand-alone checking just presented with the RPST to achieve a conformance analysis on a hierarchical manner. RPST nodes enriched with conformance information enable the analysis at different degrees of granularity and independence, similar to zooming in and out using online maps. Note that, by construction, the root of the

RPST is the overall net. Therefore, any hierarchical analysis that involves the conformance checking of all the RPST nodes will require checking conformance on the original net (plus the checks of the rest of nodes), i.e., the computation time for a exhaustive hierarchical analysis will always be, by definition, greater than checking conformance on the overall net. For complex and time-consuming cases, this problem can be alleviated by limiting the size of the nodes to check or by using less expensive replay-based conformance techniques like [3, 21]. The latter techniques use heuristics in order to deal with unfitting situations.

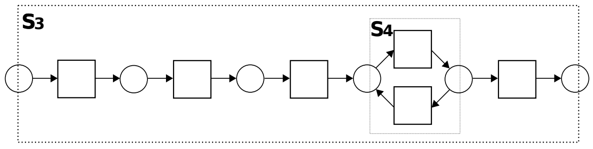
5.3. Filtering

The experiments presented in [11] suggest that there are three main differences between manual hierarchical decomposition and the one provided by the RPST-based decomposition: (1) analysts prefer to *discard small components*, (2) analysts prefer to *not consider similar components*, and (3) analysts prefer to *have a hierarchy with a limited number of levels*. Additionally, in this paper we point out a fourth difference: (4) analysts prefer to *base hierarchies on other (non-control-flow) perspectives*. In the remainder of this section we propose filtering techniques to allow for RPST-based decompositions closer to hierarchical decompositions preferred by analysts.

- *Small components:* Small components of the RPST can be removed by filtered using a minimal size threshold.
- *Similarity:* In order to reduce the redundancy of components and the unnecessary growth of the hierarchy, a *similarity metric* between parent-child components is defined, together with a threshold that determines the similarity frontier that will determine when two components are considered essentially the same. The proposed metric for estimating the similarity between a node S and its *single* child S' is based on two factors: size and simplicity. The *size* factor is related with the number of arcs of S not included on S'. The more arcs shared by both components, the more similar they are. For instance, considering the component S_1 of Fig. 13a, all its arcs are included in S_2 except two, i.e., S_2 is in essence S_1. Therefore, a detailed conformance diagnosis over S_1 may be sufficient for understanding both subprocesses. The *simplicity* factor refers to the simplicity of part of the parent S not included on the child S'. When such part defines a simple behavior (e.g., the strictly sequential behavior of S_3 not included in S_4, in Fig. 13b), the analysis and understanding of the parent may again be enough. On the other hand, when the behavior not included in S' contains complex control-flow constructs (e.g., mixtures of concurrency and choice) it may be more advisable to analyze both subprocesses. An example similarity metric is formalized as follows.



(a) similar size among SESEs



(b) high simplicity among SESEs

Figure 13: Example of cases with high similarity between nested SESEs.

Definition 20 (Similarity Metric) *Let $S_P = (V_P, F_P)$ be an RPST node, and let $S_C = (V_C, F_C)$ be its only child. Let* size *define the difference on size between them, i.e., size $= |F_C|/|F_P|$. Let $F_O = F_P \setminus F_C$ be the set of non-intersecting arcs. Let F^*_O be the arcs in F_O that have a source vertex with only one outgoing edge, and a target vertex with only one incoming edge, i.e., $F^*_O = \{(x,y) \in F_O | (x,v) \in F_O| = 1 \wedge |(w,y) \in F_O| = 1\}$. Let* simplicity *define the simplicity of the non-intersecting arcs, i.e., simplicity $= |F^*_O|/|F_O|$. The* similarity *between S_P and S_C is the harmonic mean between size and simplicity:*

$$similarity = 2 \cdot \frac{size \cdot simplicity}{size + simplicity}$$

Although the similarity evaluation is restricted to nodes with only one child, our experimental results show that the reduction achieved on the RPST may be significant (specially after applying a small nodes filtering).

- *Multi-perspective filtering:* The filtering presented until now is based on only structural net properties, not taking into account other perspectives (e.g., data, costs, roles, departments). However, there may be situations where we would like to focus the analysis only on those subprocesses satisfying certain domain conditions, e.g., an analyst may want to focus on the subprocesses involving tasks executed in a particular department. Therefore, we need to support filtering based on user-requirements and focus the analysis

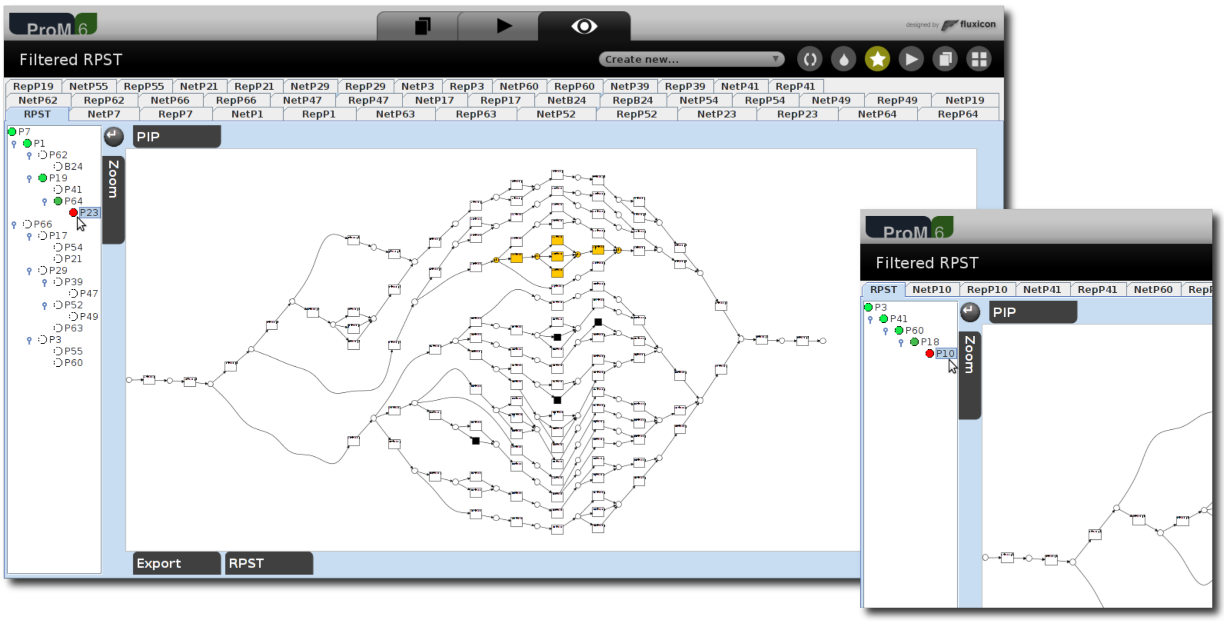


Figure 14: A screenshot illustrating the filtering by similarity and small components (left), or together with multi-perspective (right).

on the subprocesses involving activities relevant from the selected viewpoint. Such filtering is not limited to activities and may involve other perspectives (e.g., resources, actors, or costs), determining the activities they are connected with, and using them for filtering. This type of filtering was not considered in [11]. Figure 14 illustrates some of the aforementioned multi-level analysis with a log that includes information on resources. The figure is a screenshot of the tool implementing the techniques of this paper. The figure shows two windows, representing the result of applying two different filters to a model for this log. In the left-hand side of the main window, the filtered RPST is shown. The parameters used for the filtering are similarity 0.8 and considering small components to be less than 10. The RPST is enhanced with conformance information (white RPST nodes have no conformance problems, and a gradient between green and red on an RPST node identifies the severity of a conformance problem). By clicking in any of the RPST nodes, the corresponding SESE in the net is highlighted. The user can analyse the replay or the net used (cf. Sect. 5.1) on the tab section in the top of the figure for each one of the RPST nodes. Finally, the tool provides the means to focus on particular transitions of the model that involve some resources. In the case of the example, the RPST of the left window is further filtered by only considering the transitions under the supervision of the user *Bob*. This is shown in the small right window.

6. Implementation and Experiments

In this section we provide experimental results demonstrating that our decomposition approach provides significant performance gains and improved diagnostics. Moreover, we briefly describe the *Decomposed Conformance* package in ProM used to conduct these experiments.

Implementation

The techniques presented in this paper have been implemented within the ProM 6 tool[5], and have been included in the package *Decomposed Conformance*. The new implementation (unlike the one presented in [11, 12]) is fully compliant with the *Divide&Conquer* package – a new initiative for a common framework among all decomposed process mining techniques within ProM, such as [22, 23, 11, 12]. This modular framework is based on common objects and encourages and facilitates the reusability of plugins, both for discovery and conformance. Compared to the implementation in [11, 12] the architecture changed dramatically. For example, arrays of components are used, instead of the

[5]`http://www.promtools.org/prom6/nightly`

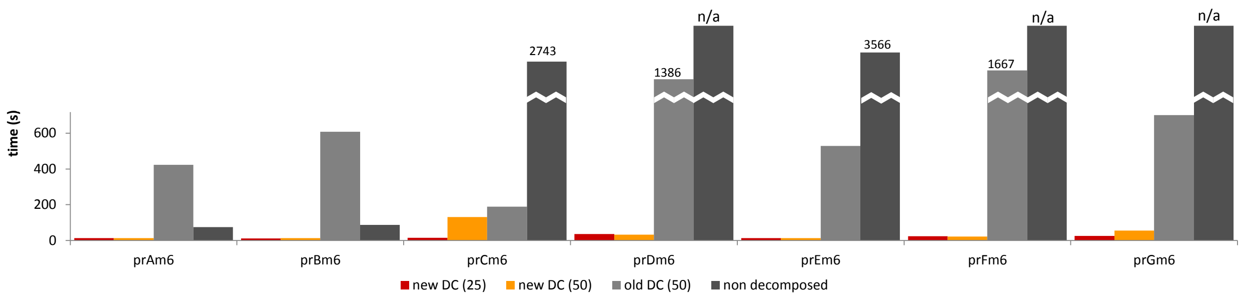


Figure 15: Comparison of computation time among different approaches: the new decomposed conformance checking technique (two variants: one which limits the maximum size of each component to $k = 25$ and the other to $k = 50$), the old decomposed conformance checking technique [12], and the approach without decomposition.

sequential creation and processing of the components. These changes in the implementation had a significant impact on the performance of the techniques. The *Decomposed Conformance* package in ProM supports all of the decomposition and filtering approaches and the advanced diagnostics described in this paper.

Performance Improvements

Figure 15 illustrates the performance improvement achieved by SESE-based decomposition and the new implementation in ProM. For this first analysis we use the *bpm2013* benchmark[6] that was also used in [12]. The benchmark contains large models with different levels of fitness (ranging from perfectly fitting as *prBm*6, to models with fitness of 0.57 – like *prCm*6), according to the fitness metric in [6]. Figure 15 compares four approaches using seven model-log combinations. The chart includes the results of the new Decomposed Conformance (DC), using a *k* to decompose of 25 and 50 respectively (cf. Alg. 1). In this experiment (and in the rest of section) we used a conformance based on alignments for checking the conformance of each component [6]. The comparison also includes the results of the previous implementation (with *k* of 50) [11, 12], and the non-decomposed results of [6] (using the same algorithm and parameters as the decomposed approach).

The chart illustrates perfectly the vast difference, in computation time, between the approach presented and the non-decomposed alternative. The non-decomposed approach remains competitive for the less complex and highly fitting models (e.g., *prAm*6 and *prBm*6). Because of the component creation overhead the non-decomposed approach may even be faster for simple and well-fitting models as noted in [12]. For example, for *prAm*6 and *prBm*6 the non-decomposed approach is faster than the previous implementation presented in [12]. This is no longer the case for the new decomposed implementation which is outperforming the earlier approaches. In those cases where the complexity and fitness is an issue, the difference could reach two orders of magnitude (e.g., from 15 to 3566 seconds in *prEm*6). More importantly, the approach proposed in this paper is able to tackle and provide conformance information for those cases (*prDm*6, *prFm*6 and *prGm*6) where [6] is not able to provide a result within a period of 12 hours. Notice though, that the goal of both approaches is slightly different: while [6] aims for a global conformance, the decomposed approach aims for an optimal conformance of each component. However, a decomposed approach makes it possible to locate in a smaller vicinity where the conformance problems are, get a better understanding of the cause, and eventually be able to provide and bound conformance properties in a global manner [13, 11, 12]. The comparison also shows the significant speedup yield by the new implementation with respect to the one in [12], due to the new architecture based on arrays of components.

Conformance Diagnosis

One of the main contributions presented in this paper is a decomposed strategy to aid on the diagnosis of conformance problems in large systems, pinpointing which subprocesses are producing them. In order to illustrate this contribution we provide the fitness results per component for the running example and the benchmark *bpm2013* (cf. Fig. 16 and Fig. 17).

We use a circumference to graphically depict the fitness evaluation of a decomposition by means of a colored gradient for each component. All components of the decomposition are placed in different positions of

[6]`http://dx.doi.org/10.4121/uuid:44c32783-15d0-4dbd-af8a-78b97be3de49`

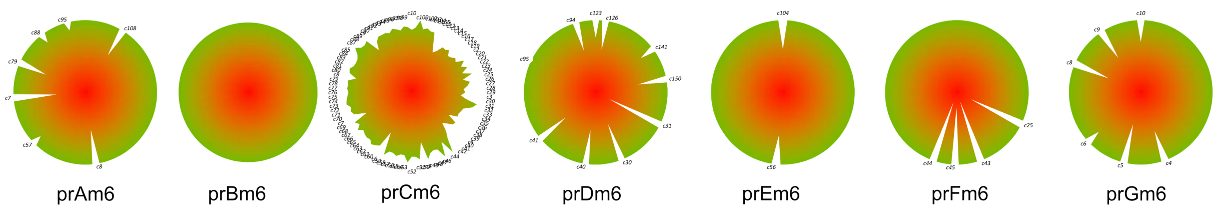


Figure 17: Fitness results per components for benchmark *bpm2013*.

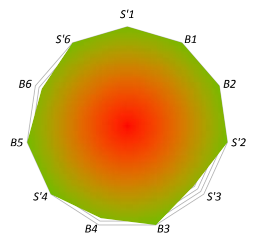


Figure 16: Fitness visualization for the running example.

the circumference. Let us use the running example of this paper to illustrate the graphical visualization used.

For each component, a line from the center of the circumference indicates its fitness. If the line reaches the perimeter, the fitness is 1.0 (components S'_1, B_1, B_2, S'_2, B_3, S'_4, B_5, S'_6), while the line for components with fitness anomalies does not reach the perimeter. To show intuitively the fitness, a color gradient is included in the circumference: the fitness ranges from red (fitness problems close to 0.0) down to green (perfect fitness of 1.0).

The fitness diagnosis of each one of the models of benchmark *bpm2013* can be understood more easily: for model *prAm*6, 7 components have fitness anomalies, with diverse severity.[7] On the other hand, all components in *prBm*6 are perfectly fitting. This contrasts with *prCm*6, where fitness problems are clearly spread over multiple components. The other of benchmark model-log combinations have fitness anomalies in just a few components. This supports the approach taken in this paper. The diagnostics help to focus on the problematic parts while at the same time provide performance gains.

Performance vs Comprehension

The previous experiment included two different sizes for the decomposition: 25 and 50. This second set of experiments is designed to determine the *optimal* decomposition size for both perspectives: performance impact and comprehension of the conformance anomalies detected. The benchmark used (*isbpm2013*), has been made publicly available within the *3TU.Datacentrum*, and can be referred and accessed through its DOI[8]. The *isbpm2013* benchmark includes several models, and a set of logs with different properties for each model. The models have been generated by *PLG* tool [24], and the logs are the result of the log generator based on the replayer from [21]. In this experiment we have checked conformance for different values of k. Note that, when the k is the total number of arcs of the model, the decomposed approach behaves as non-decomposed (i.e., there is only one component). Figure 18 reflects the computation time for two of the cases in the benchmark (*pr1908-m34-l3-noise* and *pr1151-m37-l3-noise*), which summarizes the global tendency for all the models in the benchmark. The mark representing the minimum time is slightly differentiated.

The main conclusion one can reach from the experiments is that, from a computational time point of view, the smaller the better. This is perfectly reflected in Fig. 18. For small values of k (e.g., $1 \dots 20$), the results show a slight overhead because many components need to be created. However the effect of this overhead is negligible in most of the cases. On the other hand, when the k-decomposition starts to allow larger components, the computation time abruptly increases. This disruption on the computation time is produced by the hierarchical nature of the RPST, e.g., a decomposition k instead of $k+1$ could lead the selection of n subprocesses of low complexity instead of the one subprocess that includes all n. The results and insights based on these new experiments differ from [12], where –due to inefficiencies of the previous implementation– the overhead caused by the processing of components was significantly higher, making k of 200 faster than 50 in some cases.

If components are excessively small (e.g., $1 \dots 10$), the semantic information they provide is rather trivial

[7]When no fitness anomalies exist, we do not explicitly label components in the visualization.

[8]`http://dx.doi.org/10.4121/uuid:b8c59ccb-6e14-4fab-976d-dd76707bcb8a`

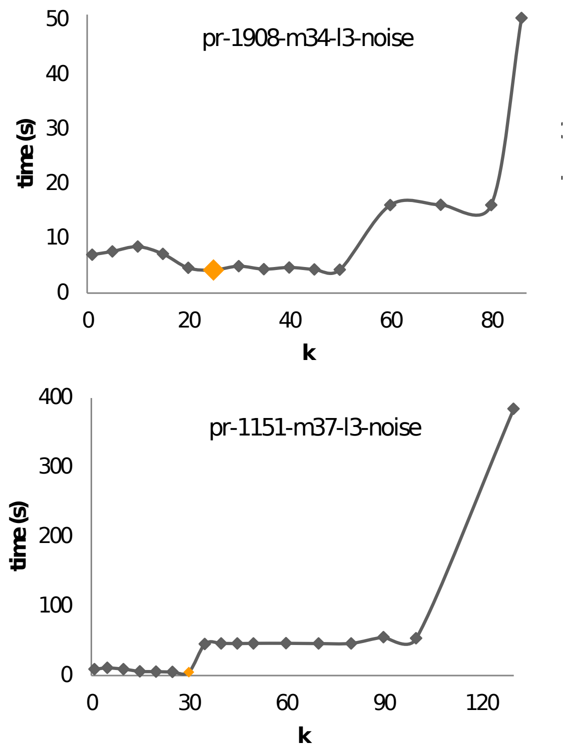


Figure 18: Comparison of computation time among different *k* values.

or insufficient. From a diagnostic point of view, the subprocesses to be analyzed should be large enough to be able to make meaningful conclusions. Our empirical experiments show that, decomposition with *k* between 20 and 40 could represent a good trade-off between computation time and diagnostic quality.

Trace length and grouping

A third set of experiments was conduced to study the effect of the trace lengths on the proposed approach. We aim to compare decomposed and non-decomposed conformance checking for different traces lengths. All logs and models of this experiments are included in the *isbpm2013* benchmark. For each model used in this experiment, four logs were generated, each one with a different average length of the traces on it (e.g., *pr1908-m18-l1* has an average trace length of 18, while *pr1908-m41-l4* has average length of 41). Each one of these four logs has been generated from simulating the same model and using the same parameters (except the length of the traces), and all them are completely fitting. Additionally, we have created another four logs for each model, with the same characteristics, but containing noise (and hence being non-fitting). Figure 19 shows the results for two models: *pr-1908* and *pr-1151*, being the results similar for the rest of models-logs in the benchmark. For each model, the chart contains the computation times of each alternative: decomposed (using *k* of 25) with noisy logs and fitting logs, and the results for the same noisy and fitting logs using the original non-decomposed approach.

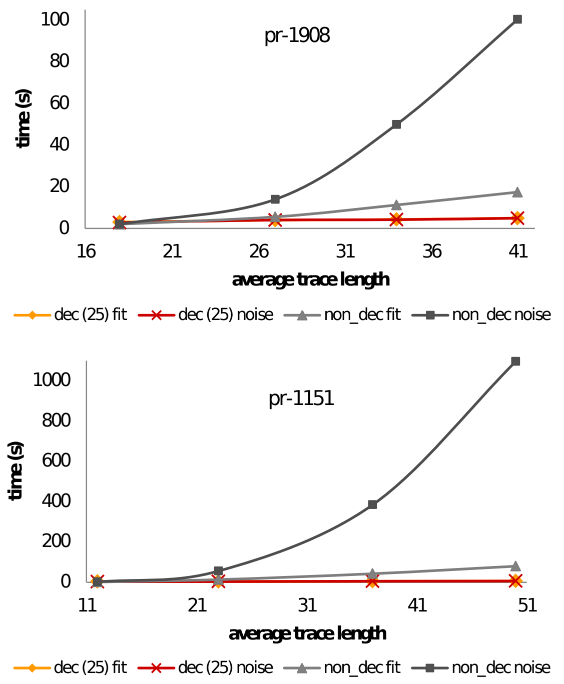


Figure 19: Comparison of computation time among different trace length.

The first conclusion that arises from the experiments refers to the processes with noise – the most plausible assumption in a real world scenario. Figure 19 shows that, when the log has short traces, both decomposed and non-decomposed alignment checking perform good. However, once the length of the traces grows (or simply traces of large models or with lot of concurrency), it has a severe effect on the non-decomposed performance. This was to be expected, i.e., the more activities in a trace, the more difficult it is to compute the alignment. On the other hand, the decomposed approach performs both fast and with a near-to constant growth (and eventually constant at some point). This is justified by the effect of the decomposition on the computation time (as has been shown in Fig. 15), but also due to the *grouping* (as explained below).

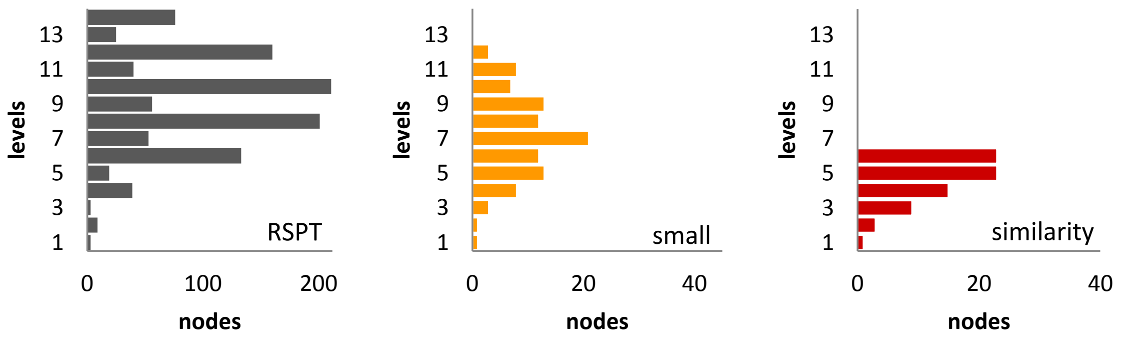


Figure 20: Results of filtering by small (< 10) and merging by similarity (> 0.8) over the model *prCm*6.

The current implementation of the align-based conformance checking includes the grouping optimization: when the algorithm analyzes a trace, it first checks if it has already computed an alignment for an identical trace. In this is the case, it re-uses the previously computed alignment, thus reducing the time significantly. The effect of this optimization for the non-decomposed scenario depends on the case at hand, and it is strongly related with the size and the behavior of the model. However, in the decomposed scenario, the chances to encounter this situation increase: the smaller is the component (e.g., $k = 25$), the fewer activities it contains, and therefore, the more likely it is to find a trace already seen before (once the original trace has been projected onto the component). The effects of the grouping are perfectly reflected by the fitting cases of Fig. 19: the decomposed approach performs faster than the non-decomposed alternative even in a fitting scenario. This is remarkable because alignments can be created easily in this case.

Topology and Filtering

The last set of experiments is designed to illustrate the effects of some of the techniques proposed for process diagnosis. In particular, the *Non-fitting Subnet Algorithm* (cf. Alg. 4), and the techniques of filtering the RPST based on small components and similarity (cf. Sect. 5.3). Table 1 shows the application of the `NFN` algorithm over the benchmark *bpm2013*, with components of size at most 50. For each model (containing P places and T transitions) the table provided the size of the minimal net containing all the non-fitting components, i.e., the number of places and transitions ($|P|$ and $|T|$), and the number of vertices $|V|$ used to create the net. The table illustrates the benefits of the proposed algorithm to detect and isolate the fitness mismatches. In case the fitness problems are spread all over the whole model, the resulting net is almost the original net (e.g., *prCm*6). However, when the fitness problems are local, the net that encloses all problem spots may be orders of magnitude smaller than the original net, thus easing the diagnosis.

Table 1: Results of `NFN` algorithm.

Dataset			`NFN`		
	P	T	$\|V\|$	$\|P\|$	$\|T\|$
prAm6	363	347	14	15	14
prCm6	317	317	113	315	317
prDm6	529	429	31	55	52
prEm6	277	275	31	29	40
prFm6	362	299	7	27	25
prGm6	357	335	5	34	29

The final experiment performed illustrates the effects of the simplification techniques over the RPST. Figure 20 reflect the results for one of the models (*prCm*6). The charts show the number of nodes of the original RPST, after filtering small components (< 10) and then merging by similarity (> 0.8). The number of nodes are distributed by levels of depth in the RPST tree, i.e., the distance with the root represented as the level 1. The chart clearly reflects the difference between the number of components on the original RPST and the one after removing the small components, i.e., most of the RPST nodes are small. After removing small nodes the depth of the RPST only decreases two levels (from 14 to 12). On the other hand, when merging on similarity is applied over the filtered RPST, the number of nodes is not reduced so drastically, but the number of levels of the tree is (from 13 to 6), providing a hierarchical decomposition with less redundancy and more aligned with the human perception [11].

7. Related Work

For an introduction to process mining we refer to [1]. For an overview of best practices and challenges, we refer to the Process Mining Manifesto [25].

The seminal paper [3] established the basics of conformance checking, based on the use of model replay to determine the fitness of a log trace. In the last years, however, the use of alignment techniques has increased [6]. In spite of its high complexity, alignment techniques provide a robust analysis of fitness anomalies, especially in the presence of non-determinism in the model or noise in the logs. Replay techniques have been revisited recently in the scope of a general framework for conformance checking based on the use of negative information [21].

Decomposing large graphs into smaller fragments is a topic widely studied in the literature. There exist plenty of techniques with different goals for that task, e.g., minimizing the connectivity among fragments [26], or obtaining fragments with a single entry and a single exit [16, 10]. Regarding decomposed conformance checking, there have been recent contributions related to the one presented in this paper. In [22] it is shown that so-called "passages" can be used to decompose both process discovery and conformance checking problems. The aforementioned work is then generalized in [13], where the minimal requirements for decomposing conformance in general are described. This paper uses the results presented in [13] for the particular case of SESEs. Experimental results suggest that a SESE-based decomposition is faster than a decomposition based on passages.

Also related is the work on conformance checking of proclets [27]. Proclets can be used to define so-called artifact centric processes, i.e., processes that are not monolithic but that are composed of smaller interacting processes (called proclets). In [27] it is shown that conformance checking can be done per proclet by projecting the event log onto a single proclet while considering interface transitions in the surrounding proclets.

8. Conclusions and Future Work

This paper presented a novel decomposition technique for handling large conformance instances and enabling the detailed and focused diagnosis of conformance checking. By focusing the decomposition on finding subprocesses with a clear interface to the rest of the process we achieve two goals: (1) conformance checking can be done much faster and allows us to analyze models/logs previously intractable and (2) improved diagnostics. We were able to exploit existing results for Single-Entry Single-Exit (SESE) decompositions and apply these to the process mining domain.

The decomposition approach and related diagnostic capabilities have been implemented in ProM. Extensive experimental results demonstrate the advantages of our approach and efficiency of the implementation. The use of the decomposed paradigm reduces the computation time of the original stand-alone conformance approach due to a reduction of the state-space to explore, and the effect of the grouping. Moreover, innovative visualizations of the misalignments further support the diagnosis of the conformance problems. The technique can deal with models of hundreds of nodes, representing a disruptive step into enabling conformance checking for industrial scenarios.

There are several research directions that can be tackled based on the results presented in this paper. First, we aim to extend divide and conquer techniques with a decomposed fitness metric that would serve as an estimate for the real fitness value of the whole model. We would also like to include other conformance metrics, e.g., precision. Note that this is far from trivial for metrics that correspond to global properties. We also aim to extend the technique presented in this paper in order to deal with other perspectives (e.g., data) available in the event log. It appears to be possible to apply the approach to process models having data conditions or resource allocation rules. Future work includes also the study of new decomposition techniques that minimize the bridging process, in order to preserve the strict SESE structure of the fragments as much as possible. Finally, we would like to use the insights provided in this paper to develop SESE-based discovery techniques.

Acknowledgments

The authors would like to thank Dr. Artem Polyvyanyy for his comments, and Seppe vanden Broucke for his help with the experiments. This work has been partially supported by the Ministerio de Educación (AP2009-4959) and by the project TIN-2007-66523. This work was also supported by the Basic Research Program of the National Research University Higher School of Economics (HSE) in Moscow.

[1] W. M. P. van der Aalst, Process Mining: Discovery, Conformance and Enhancement of Business Processes, Springer, 2011.

[2] S. Rogers, Big data is scaling BI and analytics-data growth is about to accelerate exponentially, Information and Management - Brookfield 21 (5) (2011) 14.

[3] A. Rozinat, W. M. P. van der Aalst, Conformance checking of processes based on monitoring real behavior, Inf. Syst. 33 (1) (2008) 64–95.

[4] T. Murata, Petri nets: Properties, analysis and applications., Proceedings of the IEEE 77 (4) (1989) 541–580. doi:10.1109/5.24143.

[5] W. M. P. van der Aalst, A general divide and conquer approach for process mining, in: M. Ganzha, L. Maciaszek, M. Paprzycki (Eds.), Federated Conference on Computer Science and Information Systems (FedCSIS 2013), IEEE Computer Society, 2013, pp. 1–10.

[6] A. Adriansyah, B. F. van Dongen, W. M. P. van der Aalst, Conformance checking using cost-based fitness analysis, in: EDOC, IEEE Computer Society, 2011, pp. 55–64.

[7] A. Adriansyah, J. Munoz-Gama, J. Carmona, B. F. van Dongen, W. M. P. van der Aalst, Alignment based precision checking, in: M. L. Rosa, P. Soffer (Eds.), Business Process Management Workshops, Vol. 132 of Lecture Notes in Business Information Processing, Springer, 2012, pp. 137–149.

[8] A. Adriansyah, J. Munoz-Gama, J. Carmona, B. F. van Dongen, W. M. P. van der Aalst, Measuring precision of modeled behavior, Information Systems and e-Business Management (2014) 1.

[9] IEEE Task Force on Process Mining – Case Studies , `http://www.win.tue.nl/ieeetfpm/doku.php?id=shared:process_mining_case_studies`.

[10] A. Polyvyanyy, J. Vanhatalo, H. Völzer, Simplified computation and generalization of the refined process structure tree, in: M. Bravetti, T. Bultan (Eds.), WS-FM, Vol. 6551 of Lecture Notes in Computer Science, Springer, 2010, pp. 25–41.

[11] J. Munoz-Gama, J. Carmona, W. M. P. van der Aalst, Hierarchical conformance checking of process models based on event logs, in: Colom and Desel [28], pp. 291–310.

[12] J. Munoz-Gama, J. Carmona, W. M. P. van der Aalst, Conformance checking in the large: Partitioning and topology, in: F. Daniel, J. Wang, B. Weber (Eds.), BPM, Vol. 8094 of Lecture Notes in Computer Science, Springer, 2013, pp. 130–145.

[13] W. M. P. van der Aalst, Decomposing Petri nets for process mining: A generic approach, Distributed and Parallel Databases 31 (4) (2013) 471–507.

[14] J. Munoz-Gama, J. Carmona, A fresh look at precision in process conformance, in: R. Hull, J. Mendling, S. Tai (Eds.), BPM, Vol. 6336 of Lecture Notes in Computer Science, Springer, 2010, pp. 211–226.

[15] W. M. P. van der Aalst, A. Adriansyah, B. van Dongen, Replaying History on Process Models for Conformance Checking and Performance Analysis, WIREs Data Mining and Knowledge Discovery 2 (2) (2012) 182–192.

[16] J. E. Hopcroft, R. E. Tarjan, Dividing a graph into triconnected components, SIAM J. Comput. 2 (3) (1973) 135–158.

[17] A. Polyvyanyy, Structuring process models, Ph.D. thesis, University of Potsdam (2012).

[18] J. Vanhatalo, H. Völzer, J. Koehler, The refined process structure tree, Data Knowl. Eng. 68 (9) (2009) 793–818.

[19] R. P. J. C. Bose, W. M. P. van der Aalst, I. Zliobaite, M. Pechenizkiy, Handling concept drift in process mining, in: H. Mouratidis, C. Rolland (Eds.), CAiSE, Vol. 6741 of Lecture Notes in Computer Science, Springer, 2011, pp. 391–405.

[20] M. Silva, E. Teruel, J. M. Colom, Linear algebraic and linear programming techniques for the analysis of place or transition net systems, in: W. Reisig, G. Rozenberg (Eds.), Petri Nets, Vol. 1491 of Lecture Notes in Computer Science, Springer, 1996, pp. 309–373.

[21] S. K. L. M. vanden Broucke, J. D. Weerdt, J. V. B. Baesens, Determining process model precision and generalization with weighted artificial negative events, IEEE Transactions on Knowledge and Data Engineering (accepted).

[22] W. M. P. van der Aalst, Decomposing process mining problems using passages, in: S. Haddad, L. Pomello (Eds.), Petri Nets, Vol. 7347 of Lecture Notes in Computer Science, Springer, 2012, pp. 72–91.

[23] S. J. J. Leemans, D. Fahland, W. M. P. van der Aalst, Discovering block-structured process models from event logs - a constructive approach, in: Colom and Desel [28], pp. 311–329.

[24] A. Burattin, A. Sperduti, Plg: A framework for the generation of business process models and their execution logs, in: M. zur Muehlen, J. Su (Eds.), Business Process Management Workshops, Vol. 66 of Lecture Notes in Business Information Processing, Springer, 2010, pp. 214–219.

[25] IEEE Task Force on Process Mining, Process Mining Manifesto, in: F. Daniel, K. Barkaoui, S. Dustdar (Eds.), Business Process Management Workshops, Vol. 99 of Lecture Notes in Business Information Processing, Springer, 2012, pp. 169–194.

[26] G. Karypis, V. Kumar, A Fast and Highly Quality Multilevel Scheme for Partitioning Irregular Graphs, SIAM Journal on Scientific Computing 20 (1) (1999) 359–392.

[27] D. Fahland, M. de Leoni, B. F. van Dongen, W. M. P. van der Aalst, Conformance checking of interacting processes with overlapping instances, in: S. Rinderle-Ma, F. Toumani, K. Wolf (Eds.), BPM, Vol. 6896 of Lecture Notes in Computer Science, Springer, 2011, pp. 345–361.

[28] J. M. Colom, J. Desel (Eds.), Application and Theory of Petri Nets and Concurrency - 34th International Conference, PETRI NETS 2013, Milan, Italy, June 24-28, 2013. Proceedings, Vol. 7927 of Lecture Notes in Computer Science, Springer, 2013.

Appendix A. Benchmarks

Table A.2: *isbpm2013* and *bpm2013* Benchmarks – Logs

log	cases	activities	length	fitting
pr-1244-A59_l1	2000	59	17	yes
pr-1244-A59_l1_noise	2000	59	17	no
pr-1244-A59_l2	2000	59	29	yes
pr-1244-A59_l2_noise	2000	59	29	no
pr-1244-A59_l3	2000	59	41	yes
pr-1244-A59_l3_noise	2000	59	41	no
pr-1244-A59_l4	2000	59	55	yes
pr-1244-A59_l4_noise	2000	59	55	no
pr-1151-A48_l1	2000	48	12	yes
pr-1151-A48_l1_noise	2000	48	12	no
pr-1151-A48_l2	2000	48	23	yes
pr-1151-A48_l2_noise	2000	48	23	no
pr-1151-A48_l3	2000	48	37	yes
pr-1151-A48_l3_noise	2000	48	37	no
pr-1151-A48_l4	2000	48	50	yes
pr-1151-A48_l4_noise	2000	48	50	no
pr-1908-A32_l1	2000	32	18	yes
pr-1908-A32_l1_noise	2000	32	18	no
pr-1908-A32_l2	2000	32	27	yes
pr-1908-A32_l2_noise	2000	32	27	no
pr-1908-A32_l3	2000	32	34	yes
pr-1908-A32_l3_noise	2000	32	34	no
pr-1908-A32_l4	2000	32	41	yes
pr-1908-A32_l4_noise	2000	32	41	no
pr-1912-A57_l1	2000	57	15	yes
pr-1912-A57_l1_noise	2000	57	15	no
pr-1912-A57_l2	2000	57	26	yes
pr-1912-A57_l2_noise	2000	57	26	no
pr-1912-A57_l3	2000	57	39	yes
pr-1912-A57_l3_noise	2000	57	39	no
pr-1912-A57_l4	2000	57	52	yes
pr-1912-A57_l4_noise	2000	57	52	no

log	cases	tasks	length	fitting
prAm6	1200	317	32	no
prBm6	1200	317	41	yes
prCm6	500	317	43	no
prDm6	1200	429	249	no
prEm6	1200	275	99	no
prFm6	1200	299	241	no
prGm6	1200	335	143	no

Table A.3: *isbpm2013* and *bpm2013* Benchmarks – Models

model	*P*	*T*	*A*	Inv
pr-1244-A59	71	68	164	9
pr-1151-A48	55	56	130	8
pr-1908-A32	37	36	86	4
pr-1912-A57	64	57	142	5

model	*P*	*T*	*A*
prAm6	363	347	842
prBm6	317	317	748
prCm6	317	317	748
prDm6	529	429	1136
prEm6	277	275	648
prFm6	362	299	768
prGm6	357	335	822