Conformance Checking in the Large: Partitioning and Topology

Jorge Munoz-Gama[1], Josep Carmona[1], and Wil M.P. van der Aalst[2]

[1] Universitat Politecnica de Catalunya, Barcelona, Spain
[2] Eindhoven University of Technology, Eindhoven, The Netherlands
{jmunoz,jcarmona}@lsi.upc.edu, w.m.p.v.d.aalst@tue.nl

Abstract. The torrents of event data generated by today's systems are an important enabler for process mining. However, at the same time, the size and variability of the resulting event logs are challenging for today's process mining techniques. This paper focuses on "conformance checking in the large" and presents a novel decomposition technique that partitions larger processes into sets of subprocesses that can be analyzed more easily. The resulting topological representation of the partitioning can be used to localize conformance problems. Moreover, we provide techniques to refine the decomposition such that similar process fragments are not considered twice during conformance analysis. All the techniques have been implemented in ProM, and experimental results are provided.

Keywords: Process Mining, Conformance Checking, Process Diagnosis.

1 Introduction

The interest in process mining is increasing because of the widespread availability of event data and the desire to improve performance and compliance of operational processes. Process mining relates modeled behavior and observed behavior [1,2]. This novel discipline tackles three challenges relating event data (i.e., log files) and process models: the *discovery* of a process model from an event log, checking the *conformance* of a process model and a log, and the *enhancement* of a process model with the information extracted from a log. Process mining research resulted in a variety of algorithms that demonstrated to be of great value for undertaking small or medium-sized problem instances. However, real-life experiences show that most of the existing algorithms have difficulties dealing with industrial-sized problems (cf. Section 6).

This paper proposes a decomposition technique for determining the conformance of a Petri net with respect to a log (i.e., how good is the model describing the behavior of the log). Instead of trying to asses the conformance of the whole event log and the complete Petri net, we check conformance for selected subprocesses (subnets of the initial Petri net and corresponding sublogs). Subprocesses are identified as fragments 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 [3].

F. Daniel, J. Wang, and B. Weber (Eds.): BPM 2013, LNCS 8094, pp. 130–145, 2013.

In [4], we presented a conformance checking approach using the so-called *Refined Process Structure Tree* (RPST). The RPST [3] allows for the construction of hierarchy of SESEs. This paper extends the approach presented in [4] as follows: First of all, we present a new strategy to compute fitness (i.e., the dimension of conformance that focuses on analyze if the traces in the log are valid sequences of the model) by selecting a partitioning of the RPST. Then this partitioning is extended with a new set of fragments corresponding to the interface between place-bordered SESEs. With this extension, it is guaranteed that the fitness of the whole Petri net can be computed directly from the fitness of the fragments forming the partition. Experiments show a considerable reduction (orders of magnitude) in the fitness checking, and moreover the techniques allow for identifying those subnets that have fitness problems, allowing the process owner to focus on the problematic parts of a large model.

The RPST-based 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 and independently of the rest of the model. Finally, the approach is refined to avoid considering the same problem multiple times. For example, it makes no sense to consider small or highly similar process fragments.

Related Work. Cook et al. [5] were among the first to quantify the relationship between event logs and process models. They compared event streams of the model with event streams generated from the event log. Several authors proposing process discovery algorithms also provide a quality metric (often related to fitness). For example, in [6] the authors define a fitness function for searching for the optimal model using a genetic approach.

The first comprehensive approach to conformance analysis was proposed in [7]. Two different types of metrics are proposed: (a) *fitness metrics*, i.e., the extent to which the log traces can be associated with valid execution paths specified by the process model, and (b) *appropriateness metrics*, i.e., the degree of accuracy in which the process model describes the observed behavior in the log, combined with the degree of clarity in which it is represented. One of the drawbacks of the approach in [7] and most other approaches that "play the token game", is that fitness is typically overestimated. When a model and log do not fit well together, replay will overload the process model with superfluous tokens. As a result, the model will allow for too much behavior. Approaches such as the one in [7] also have problems when the model has "invisible activities" (silent steps that are not recorded in the event log) or "duplicate activities" (multiple transitions bearing the same label). To deal with such phenomena state-space exploration and heuristics are needed to replay the event log. In fact, most conformance techniques give up after the first non-fitting event or simply "guess" the corresponding path in the model. Therefore, Adriansyah et al. formulated conformance checking problems as an optimization problem [8,9].

Lion's share of attention in conformance checking has been devoted to checking fitness. However, in recent papers researchers started to explore the other

quality dimensions [1,8,10]. For example, Munoz-Gama et al. quantified the precision dimension [11,12].

In [13] various process mining decomposition approaches are discussed. In [14] the notion of passages is used to decompose a process model and/or event log into smaller parts that can be checked or discovered locally. This approach is generalized in [15] where it is shown that fitness-related problems can be decomposed as long as the different process fragments only share transitions having unique labels. This idea is used in this paper. However, unlike [14,15] we use an RPST-based decomposition that also allows for place-boundaries. Moreover, the refined RPST-based decomposition and the topological structure enable additional diagnostics not considered before.

Outline. Section 2 provides some preliminaries, before describing our partitioning approach (Section 3). Section 4 presents the notion of a topological graph to show conformance diagnostics. In Section 5 we discuss a refinement to avoid inspecting small or highly similar process fragments. Experimental results are presented in Section 6. Section 7 concludes the paper.

2 Preliminaries

To explain our conformance checking approach we introduce some basic notions. We use Petri nets (workflow nets to be precise) to model processes.

Definition 1 (Petri Net, Workflow Net). *A* Petri net*[16] is a tuple* $PN = (P, T, A)$ *having a finite set of places* P *and a finite set of transitions* T *where* $P \cap T = \emptyset$*, and a flow relation* $A \subseteq (P \times T) \cup (T \times P)$.[1] *The* preset *and* postset *of a node are defined as* ${}^\bullet x = \{y | (y, x) \in A\}$ *and* $x^\bullet = \{y | (x, y) \in A\}$*, respectively. The state of a Petri net is defined by its* marking*, i.e., a multiset over* P*. A* workflow net *(WF-net)* $WN = (P, T, A, i, o)$ *is a particular type of Petri net where the net has one source place* i *and one sink place* o*, and all the other nodes are in a path between them.*

Definition 2 (Workflow Graph). *Given a Petri net* $PN = (P, T, A)$*, we define its* workflow graph *simply as the structural graph* $G = (V, E)$ *with no distinctions between places and transitions, i.e.,* $V = P \cup T$ *and* $E = A$*.*

Definition 3 (System Net, Full Firing Sequences). *A* system net *is a tuple* $SN = (PN, m_i, m_o)$ *where* m_i *and* m_o *define the initial and final marking of the net, respectively.* $(PN, m_1)[\sigma\rangle(PN, m_2)$ *denotes that a sequence of transitions* $\sigma \in T^*$ *is enabled in marking* m_1 *and executing* σ *in* m_1 *results in marking* m_2*.* $\{\sigma \mid (PN, m_i)[\sigma\rangle(PN, m_o)\}$ *are all full firing sequences of* SN*.*

An *event log* is a multiset of traces. Each *trace* is a sequence of activities (in this paper corresponding to the set of transitions T). Multiple cases may follow the same trace.

[1] Although the approach is valid also for weighted Petri nets, for the sake of clarity in this paper we restrict to the case with no weights on the arcs.

Definition 4 (Event Log). *An* event log $L \in \mathbb{B}(T^*)$ *is a multiset of traces.*

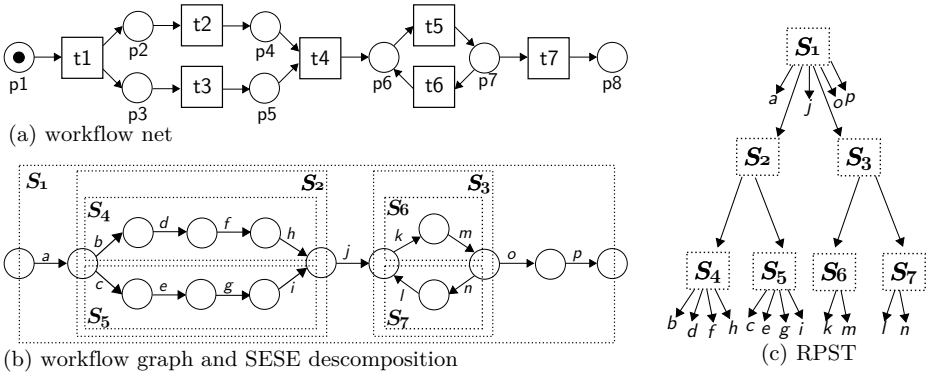
Fitness can be defined in various ways [1,8,10]. In this paper, we just classify traces into fitting or non-fitting. Fitting traces correspond to full firing sequences.

Definition 5 (Fitting Trace). *A trace* $\sigma \in T^*$ fits $SN = (PN, m_i, m_o)$ *if* $(PN, [m_i])[\sigma\rangle(PN, [m_o])$*, i.e., σ corresponds to a full firing sequence of SN. An event log* $L \in \mathbb{B}(T^*)$ fits SN *if* $(PN, [m_i])[\sigma\rangle(PN, [m_o])$ *for all* $\sigma \in L$.

To decompose conformance checking problems, we identify so-called *SESE components*. In the remainder, the following context is assumed: Let G be the workflow graph of a given WF-net WN, and let $G_S = (V_S, S)$ be a connected subgraph of G formed by a set of edges S and the vertexes $V_S = \Pi(S)$ induced by S.[2]

Definition 6 (Interior, Boundary, Entry and Exit nodes [3]). *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.*

For example, given the model in Fig. 1a and its corresponding workflow graph in Fig. 1b, let consider the subgraph S_4 containing the arcs b, d, f, h and the vertexes induced by them. The nodes corresponding with t_1 and t_4 are boundary nodes, while p_2, t_2 and p_4 are interior nodes. Moreover, the node t_1 is an entry node, while t_4 is an exit.



(a) workflow net

(b) workflow graph and SESE descomposition

(c) RPST

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

Definition 7 (SESE, Trivial SESE and Canonical SESE [3]). *$S \subseteq E$ 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*

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

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 [17], the authors proposed the algorithm for constructing the *Refined Process Structure Tree (RPST)*, i.e., an hierarchical structure containing all the canonical SESEs of a model. In [3], the computation of the RPST is considerably simplified and generalized by introducing a pre-processing step that reduces the implementation effort considerably.

Definition 8 (RPST-based Decomposition [3]). *Let G be the workflow graph of the WF-net WN.*[3] *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 $\mathbb{S}$.*

Figure 1c shows the RPST for the given example. 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 of the workflow graph or Petri net induced by those edges ($PN_S = (P \cap \Pi(S), T \cap \Pi(S), A \cap S)$). For example, the SESE S_4 of Fig. 1b containing the edges b, d, f and h, refers also to the Petri net composed by the transitions t_1, t_2 and t_4, the places p_2 and p_4, and the arcs between them in the WF-net of Fig. 1a.

3 Partitioning Conformance Diagnosis

In this section, we propose a divide-and-conquer approach for conformance checking, preserving the SESE decomposition's underlying semantics. Moreover, we show that a trace is fitting the overall model if and only if it is fitting the individual fragments. The proposed approach is based on selecting a set of RPST nodes that *partition* the set of arcs of the process model. The maximum size of the components to be analyzed can be limited in order to deal with computation time restrictions or to control the complexity of individual components. Formally:

Definition 9 (k-partitioning over a SESE decomposition). *Given the SESE decomposition $\mathbb{S}$ of a WF-net WN, we define $\mathbb{P} = \{S_1, \ldots, S_n\} \subseteq \mathbb{S}$:*

[3] 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 [3].

Algorithm 1. k-partitioning algorithm

```
procedure k-PART(RPST,k)
    V = {root(RPST)}
    ℙ = ∅
    while V ≠ ∅ do
        v ← pop(V)
        if |v.arcs()| ≤ k then ℙ = ℙ ∪ {v}
        else V = V ∪ {children(v)}
```

a partitioning *of SESEs such that each arc in WN is contained in exactly one S_i. A* k-partitioning *of* $\mathbb{S}$ *is a set of SESEs* $\mathbb{P} = \{S_1, \ldots, S_n\} \subseteq \mathbb{S}$ *where each* S_i *contains at most* k *arcs.*

Proposition 1 (k-partitioning existence). *Given a SESE decomposition* $\mathbb{S}$ *over the WF-net* $WN = (P, T, A, i, o)$*, and given any* k *such that* $1 \leq k \leq |A|$*, there always exists a* k-partitioning *of* $\mathbb{S}$*.*

Proof. By definition, any edge is a SESE (and they appear as leaves of the RPST). Therefore, a trivial partitioning with all parts being trivial SESEs is always possible. □

Algorithm 1 shows how to compute a k-partitioning. The algorithm has linear complexity (with respect to the size of the RPST) and termination is guaranteed by the fact that fragments size is reduced with every iteration.

Given a partitioning, we use it to decompose conformance checking. Remember that SESEs only interface the rest of the net through the single entry and single exit nodes, which may be shared by different SESEs. The rest of nodes of a SESE (i.e., the interior nodes) have no connection with other SESEs. For the boundary nodes, we distinguish two cases: *transition bounded* and *place bounded*.

As proven in [15], transition bounded net fragments can be checked in isolation. For a partitioning into SESEs where all entry and single nodes are transitions the following holds: a trace perfectly fitting the whole WF-net will also fit all individual SESEs (projected trace on corresponding transitions) and vice versa. For example, consider the WF-net in Fig. 2a, and the partitioning shown in Fig. 2b. The existence of d in both S_1 and S_2 ensures that both subnets move synchronously when replaying a trace. For instance, trace *abcddefg* (non-fitting in the original net due the double d) is fitting on S_2 (on S_2, the preset of d is empty), but not in S_1. On the other hand, trace *abcefg* (also non-fitting in the original net) is fitting in neither S_1 nor S_2.

However, the case of *place bounded* SESEs (i.e., entry and/or exit nodes correspond to places) is completely different. Places, unlike transitions, have no reflection in the log, and therefore, cannot be used to synchronize the individual SESEs during replay. Consider, for example, the net in Fig. 3a, and the partitioning shown in Fig. 3b. There is a strong dependency between the execution of S_1 and the initial marking considered for S_2. For example, consider a marking of one token on p and the trace *abcdef*. Such trace fits the original model, but it

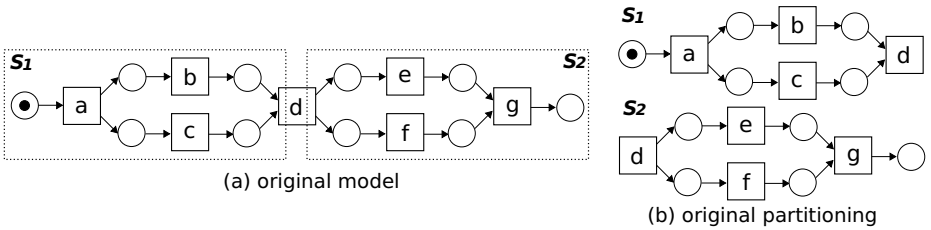


Fig. 2. Example of partitioning with transition boundary

does not fit S_2 (i.e., it requires two tokens on p). On the other hand, considering an initial marking of S_2 with two tokens on p, the trace *abdecf* fits S_1 and S_2 but does not fit the original net.

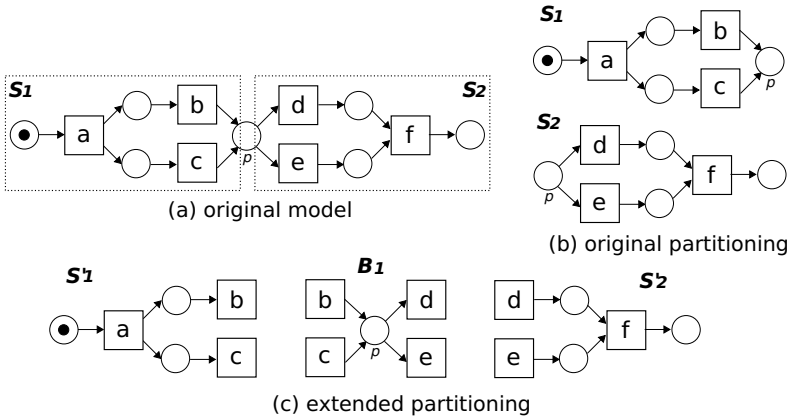


Fig. 3. Example of partitioning with place boundary

Thus, in case of place-boundaries, we extend the isolated fitness calculation by considering a new element that we call *bridge*. A bridge simply contains the pre and post sets of the boundary place. Bridges replicate the behavior on the boundary places thus synchronizing all components connected to such place. For example, given the place boundary of Fig. 3a, besides the two SESE components S'_1 and S'_2, a third component B_1 is constructed explicitly, containing the place p, its preset, and its postset (cf. Fig. 3c). Although bridges do not satisfy the SESE definition, their structure is very specific (i.e., nets with only one place). Given that the bridge makes the synchronization explicit on the boundary place, SESEs having this boundary place no longer need it, and therefore, it is removed from all the SESEs (cf. S'_1 and S'_2 on Fig. 3c). Note that the modified SESEs do not longer satisfy the SESE definition, but have a set of input and output transitions. Remarkably, the removal of boundary places in the original SESEs and the introduction of bridges ensures transition bounded fragments, and therefore, the results of [15] can be applied directly. We now formally define the so-called extended partitioning:

Definition 10 (Extended partitioning over a SESE decomposition). *Let* $\mathbb{P} = \{S_1, \dots S_n\}$ *be a partitioning of the WF-net* $WN = (P, T, A, i, o)$*. Let* $I_{\mathbb{P}} = \{i_1, \dots, i_n\}$ *and* $O_{\mathbb{P}} = \{o_1, \dots, o_n\}$ *be the set of all entry and exit nodes of the SESEs in* $\mathbb{P}$*.* $B = \{p_1, \dots, p_k\} = ((I_{\mathbb{P}} \cup O_{\mathbb{P}}) \cap P) \setminus \{i, o\} = (I_{\mathbb{P}} \cap O_{\mathbb{P}}) \cap P$ *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* extended partitioning $\mathbb{P}' = \{S'_1, \dots S'_n, B_1 \dots B_k\}$ *of* $\mathbb{P}$ *is constructed as follows:*

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

Note that, a bridge may not satisfy the *k-size* property. However, its size is limited because it contains a single place. Lemma 1 shows that, given any extended partitioning, fitness is preserved among its components, i.e., a trace fits the whole WF-net if and only if it fits all the parts of the extended partitioning.

Lemma 1 (Decomposed Fitness Checking). *Let* L *be a log and* $SN = (WN, m_i, m_o)$ *be a system net where* $WN = (P, T, A, i, o)$ *is a WF-net. Let* $\mathbb{P}'$ *be any extended partitioning over* WN*. A trace* $\sigma \in L$ fits SN *(i.e.,* $(WN, [m_i])[\sigma\rangle(WN, [m_o])$*) if and only if it fits all the parts, i.e., for all* $S \in \mathbb{P}'$*,* $PN_S = (P_S, T_S, A_S) = (P \cap \Pi(S), T \cap \Pi(S), A \cap S)$*:* $(PN_S, [m_{i\downarrow P_S}])[\sigma_{\downarrow T_S}\rangle (PN_S, [m_{o\downarrow P_S}])$*).*[4]

Proof. Special case of the more general Theorem 2 in [15]. If the overall trace σ fits SN, then each of the projected traces $\sigma_{\downarrow T_x}$ fits the corresponding SESE. If this is not the case, then at least there exist one projected trace $\sigma_{\downarrow T_x}$ that does not fit. If the projected traces $\sigma_{\downarrow T_x}$ fit the corresponding SESEs, then these traces can be stitched back into a trace σ that fits SN.

Although the use of a partitioning makes it possible to decompose a complex problem as conformance checking into smaller subproblems, there are applications (e.g., process diagnosis) where a more fined-grained analysis is required. In other words, we need to be able to navigate zooming in and out of the model, to get a better understanding (see Chapter 13 in [1]). With this idea in mind, the theory proposed on this section can be combined with the properties of the RPST to obtain a hierarchy of fitness results based on SESEs. Therefore, given an RPST, an extension based on bridges is performed over each level of the tree, and the fitness is checked for the complete level. Unlike other techniques (like in [4]), this analysis guarantees the fitness for the complete level. Note that, the RPST contains the whole net as its root. Therefore, in those cases where the complete system cannot be checked due its complexity, this is also not possible with this technique. However, a greedy procedure can be developed, that starts processing the higher levels of the RPST hierarchy (root is at level 0), and goes up until it reaches a level non-computable due to complexity or time reasons. Algorithm 2 describes the resulting conformance checking technique.

[4] $\sigma_{\downarrow T}$ is the projection of sequence σ onto transitions T and $m_{\downarrow P}$ is the projection of marking m onto the multiset of places P.

Algorithm 2. Extended RPST Conformance algorithm

procedure ConfExtRPST(RPST,log)
  $level \leftarrow heigth(RPST)$
  while $level \geq 0$ **and** $level$ computable **do**
    $\{S_1 \dots S_n\} \leftarrow$ Find partitioning containing the SESEs in $level$ of the RPST
    $\{S'_1 \dots S'_n, B_1 \dots B_k\} \leftarrow$ Extend partitioning $\{S_1 \dots S_n\}$ with bridges
    check fitness for the pairs $(S'_1, log_{\downarrow T_1}), \dots, (B_k, log_{\downarrow T_k})$
    $level \leftarrow level - 1$

4 Topological Graph of a Partitioning

In this section we present the *topological graph* of a partitioning, and some techniques that can use it to improve the diagnosis. Given an extended partitioning, the topological graph is the directed graph that represents the connections via boundary nodes between the parts. Formally:

Definition 11 (Topological Graph of a Partitioning). *Let $\mathbb{P} = \{S_1, \dots S_n\}$ be a partitioning of the WF-net $WN = (P, T, A, i, o)$, with boundary places $\{p_1, \dots, p_k\}$. Given an extended partitioning $\mathbb{P}' = \{S'_1, \dots S'_n, B_1, \dots B_k\}$ (cf. Def. 10), we define its topological graph $\mathbb{T} = (\mathbb{P}', C)$ as the graph whose vertexes are the parts of $\mathbb{P}'$, and the set of edges is $C = \{(S'_i, S'_j) | 1 \leq i, j \leq n \ \wedge \ (y, x) \in S_i \ \wedge \ (x, z) \in S_j\} \cup \{(S'_i, B_j) | 1 \leq i \leq n \ \wedge \ 1 \leq j \leq k \ \wedge \ (y, p_j) \in S_i\} \cup \{(B_j, S'_i) | 1 \leq i \leq n \ \wedge \ 1 \leq j \leq k \ \wedge \ (p_j, y) \in S_i\}$.*

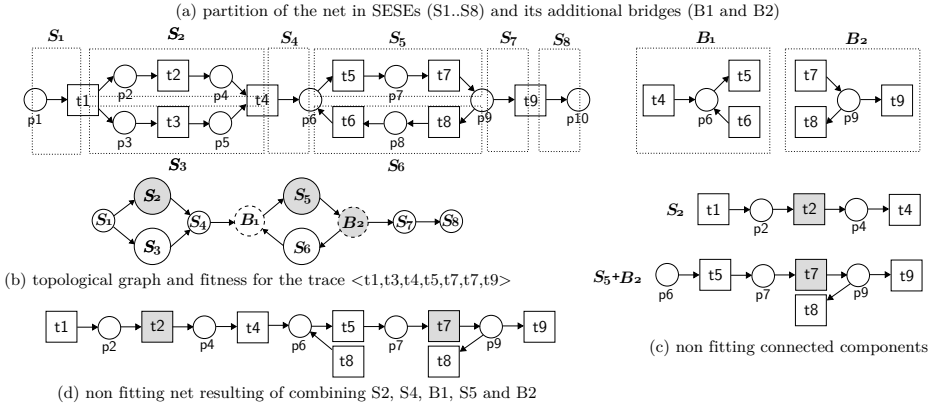


Fig. 4. Example of partitioning, topological graph, and its corresponding non-fitting connected components, and non-fitting net

Note that the topological graph has as vertexes the transition-bordered parts of the extended partitioning, but some arcs of this graph (those regarding connection to bridges) are defined over the parts of the original partitioning $\mathbb{P}$, since in the extended partitioning boundary places have been removed. One of the functions of the topological graph is to aid in the visualization of the extended

partitioning resulting from applying the techniques developed in Sec. 3. For example, let us consider an extended partitioning that arises from a 4-partitioning of the WF-net in Fig. 4a (a slight modification of the model in Fig. 1). The resulting extended partitioning is composed by the SESEs $S'_1 \dots S'_8$ and the two bridges B_1 and B_2 corresponding with the two boundary places p_6 and p_9. The corresponding topological graph is shown in Fig. 4b. Besides simply showing the connections through boundary nodes, the topological graph can be enhanced with other information. For instance, in this example, bridges are represented with dotted borders, while SESEs with solid borders. Moreover, the size of the nodes in the graph is directly related with the size of the corresponding parts, i.e., larger parts will have more importance in the representation and will appear larger than smaller parts. Finally, the graph can be enhanced with the conformance analysis results. In this example we have considered the fitness dimension of the model with respect to the log composed by only one trace $t_1t_3t_4t_5t_7t_7t_7$. Considering this trace, three parts contain fitness anomalies (filled in gray): in S'_2, t_4 is fired without firing t_2; in S'_5, t_7 is executed twice, but this requires the firing of t_5 also twice; finally, in the bridge B_2, t_7 is fired twice, but t_9 only once, leaving a token remaining in p_9.

Although the topological graph is an important aid for the process diagnosis by itself, it can also guide subsequent analysis. In the remainder of this section we present and motivate some ideas.

The topological graph extended 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 those parts of the model perfectly fitting. Algorithm 3 describes a procedure that is based on detecting connected components on the graph induced by the non-fitting vertexes. First, the topological graph is filtered, leaving only non-fitting vertexes, and the edges connecting them. Then, for each set of weakly connected components (i.e., connected vertexes without considering the direction of the edges), we project the elements of the original net they refer to. Note that this algorithm prioritizes the connectivity among vertexes resulting in weakly connected components. However, alternative versions of the algorithm yielding strongly connected components are possible. For instance, given the example of Fig. 4b, two connected components are found as shown in Fig. 4c: S_2 and $S_5 + B_2$.

The topological graph extended with conformance information can also be used to create one complete subnet that includes all non-fitting parts of the extended partitioning. We use a heuristic based on the greedy expansion of the largest non-fitting connected component (based on Algorithm 3), to get connected with the second largest component, until all the non-fitting behavior is connected, trying to include as few fitting nodes as possible. A schema of the procedure is shown in Algorithm 4. Given the example of Fig. 4b, the net resulting (shown in Fig. 4d) contains the elements of S_2, S_4, B_1, S_5 and B_2. In Sec. 6 we provide experimental results on large models for the two techniques proposed in this section.

Algorithm 3. Non-Fitting Weakly Connected Components Algorithm

function NFWCC($\mathbb{T}$,V) ▷ Let V be the non-fitting vertexes
  $C_c = \emptyset$
  remove from $\mathbb{T}$ all arcs $c = \{x, y\}$ such that $x, y \notin V$ ▷ Graph induced by V
  remove from $\mathbb{T}$ all vertexes $z \notin V$
  while $\mathbb{T}$ has vertexes **do** ▷ Find Weakly Connected Components
    $v_1 \leftarrow$ select random vertex on $\mathbb{T}$
    $\{v_1, \dots v_n\} \leftarrow$ get vertexes weakly connected with v_1 using Depth-first search
    remove $\{v_1, \dots v_n\}$ from $\mathbb{T}$
    $C_c = C_c \cup \{(\bigcup_1^n places(v_i), \bigcup_1^n trans(v_i), \bigcup_1^n arcs(v_i))\}$
  return C_c

Algorithm 4. Non-Fitting Subnet Algorithm

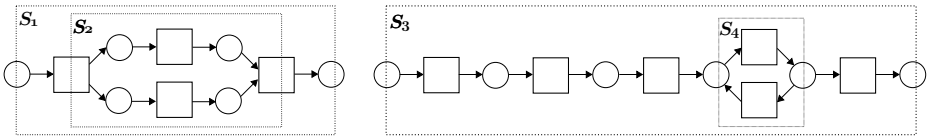
function NFN($\mathbb{T}$,V) ▷ Let V be the non-fitting vertexes
  while graph G induced by V on $\mathbb{T}$ is not connected **do**
    $c_1 \leftarrow$ get the largest connected component of G
    $c_2 \leftarrow$ get the second largest connected component of G
    $\{v_1 \dots v_n\} \leftarrow$ shortest_path_vertexes($\mathbb{T}, c_1, c_2$)
    $V = V \cup \{v_1 \dots v_n\}$.
  return Petri net induced by V

5 RPST Simplifications

Although the decomposition of a model based on SESEs and their RPST is intuitive and fine-grained, it remains different from the conceptual decomposition typically on the mind of the process analysts. In [4], the results of an experiment performed over 7 subjects identify three main differences between their manual decomposition and the one provided by the RPST: 1) predisposition of the analysts to *discard small components*, 2) to *not consider twice similar components*, and 3) to *not make grow the depth of the hierarchy unnecessarily*. In this section we formalize the two last items into a similarity metric between parent-child SESEs, enabling discarding child components when the similarity with the parent is above some threshold. Also we tackle 1) by defining a threshold on the minimal size of a SESE to consider. Note that in the case of 1), Lemma 1 may not be applicable (since a partitioning of the net may not be possible) and therefore these RPST simplifications can only be applied without any guarantee.

In particular we present a metric (cf. Def.12) for estimating the *similarity* between a node S and its single child S' based on two factors: *size* and *simplicity*. The size factor is straightforwardly 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. 5a, 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. 5b), the analysis and understanding of the parent may again be enough. On the other hand, when the behavior not included in S' contains complex constructions (e.g., mixtures of concurrency and choice) it may be more advisable to analyze both subprocesses.



(a) similar size among SESEs (b) high simplicity among SESEs

Fig. 5. Example of cases with high similarity between nested SESEs

Definition 12 (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 merging is restricted to single-child nodes, our experimental results show that the reduction achieved on the tree may be considerable. Both simplification techniques (small components and similarity merging) have been implemented and tested. The next section shows the effects of their application on large models.

6 Experimental Results

All techniques presented in this paper have been implemented within ProM framework and are accessible through the *JorgeMunozGama* package.[5] To test performance we created various benchmarks generated by `PLG` tool [18][6] In this section we first highlight the empirical differences with related conformance checking approaches described in the literature and the partitioning-based proposed in this paper. Second, we provide some results on the application of the topological graph algorithms. Finally, we illustrate the effects of the simplification methods proposed on large models.

Table 1 shows the ability to handle conformance problems of industrial size using our approach. The experiment is composed of several large models (having

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

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

P places and T transitions), and their corresponding logs. For each benchmark, the table contains the fitness value (f) and the time required for analyzing their conformance using the approach proposed in [9,11] (t). Dashes denote the lack of results after 10 hours of computation. The rest of the table contains the results of applying the same conformance technique over a 50, 100 or 200 SESE-based extended partitioning, respectively. For each k-partitioning the table provides the number of parts (S SESEs and B bridges), the number of non-trivial small parts (> 5) containing more than 5 arcs (threshold extracted from the study in [4]), and the total time required for the fitness analysis (t). In addition, the table shows the number of parts with a fitness value lower than one (i.e., non-fitting nf) and the percentage of arcs they represent within the whole model. Remarkably, the time required to compute the RPST and the k-partitioning is negligible (i.e., never more than few seconds).

Table 1. Comparison between k-partitioning and [9,11]

			[9,11]		$k = 50$				$k = 100$				$k = 200$			
	P	T	f	t	S/B	>5	nf	t	S/B	>5	nf	t	S/B	>5	nf	t
prAm6	347	363	0.92	75	129/57	29	7(3%)	423	62/27	14	1(9%)	323	27/12	7	1(10%)	180
prBm6	317	317	1	88	93/38	22	0(0%)	608	66/29	14	0(0%)	318	36/16	8	0(0%)	114
prCm6	317	317	0.57	2743	93/38	22	58(92%)	189	66/29	14	41(94%)	185	36/16	8	22(96%)	502
prDm6	529	429	-	-	105/34	33	5(8%)	1386	60/23	18	4(14%)	986	33/15	9	4(23%)	1284
prEm6	277	275	0.97	3566	82/35	20	2(5%)	529	35/13	11	2(5%)	343	15/7	5	2(6%)	211
prFm6	362	299	-	-	108/43	28	2(6%)	1667	57/23	15	2(21%)	863	21/9	5	1(23%)	562
prGm6	357	335	-	-	94/37	25	2(8%)	867	67/31	15	2(8%)	850	51/25	11	2(8%)	474

Table 1 shows that partitioning yields significant speedups in case event logs are not perfectly fitting the model [9,11]. In cases with a perfect fitting (e.g., prBm6) the time required for the proposed approach is higher, due the overhead caused by creating and storing in memory the generated parts. However, in real cases where the log is poorly fitting the model, the time needed for conformance checking is reduced in one order of magnitude using k-partitioning. More important, the proposed approach is able to tackle and provide conformance information for those cases where [9,11] is not able to provide a result (e.g., prDm6, prFm6 and prGm6).

Table 1 also shows the capability of the proposed approach to detect and isolate the subprocesses causing the fitness problems. In particular, the approach is able to identify those cases were all the fitness problems are located in only a few parts of the process, e.g., in prEm6, all the fitness problems of a net with 277 transitions can be restricted to 2 subprocesses that represent only the 5% of the model. Note that, although the number of parts generated by the approach can be considered high, most of them are trivial parts with less than 5 arcs. Importantly, the number of large parts remains low, and its maximum size can be controlled by the parameter k of Algorithm 1.

The second experiment aims at illustrating the role of the topological graph. Figure 6 shows a graphical example on how the techniques of Sec. 4 can be used for diagnosis: given a large model having conformance problems (denoted in red), Algorithms `NFWCC` and `NFN` can be used to identify one subprocess with conformance problems or a connected subnet including all the subprocesses with

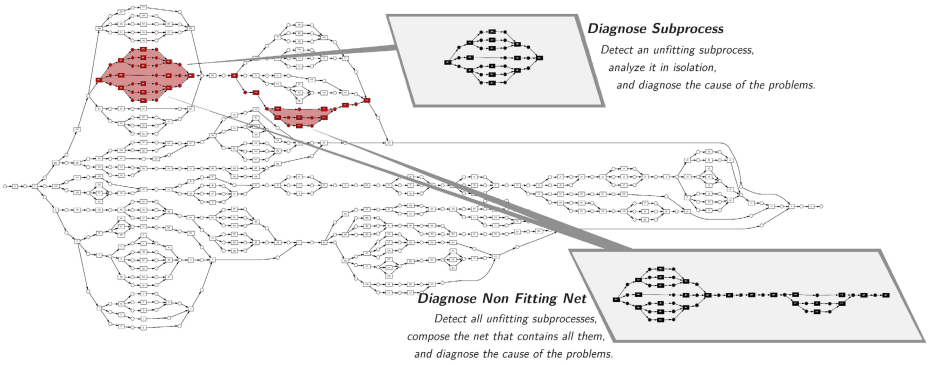


Fig. 6. The 50-partition as a diagnosis tool for the *prFm6* benchmark

Table 2. Results of `NFWCC` and `NFN` algorithms

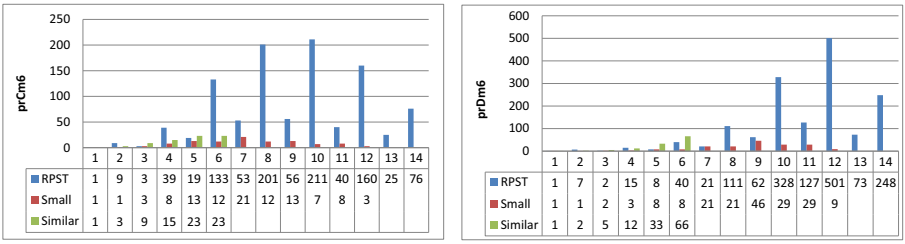
			`NFWCC`				`NFN`		
	P	T	$\|C_C\|$	$\|V\|$	$\|P\|$	$\|T\|$	$\|V\|$	$\|P\|$	$\|T\|$
prAm6	317	317	7	1	2.1	3	14	15	14
prCm6	317	317	38	1.5	8.2	9.5	113	315	317
prDm6	529	429	5	1	9.4	9.4	31	55	52
prEm6	277	275	2	1	1	2	31	29	40
prFm6	362	299	2	1	13	11	7	27	25
prGm6	357	335	2	1	16.5	14.5	5	34	29

conformance problems, respectively. Table 2 reports on the performance of these two algorithms using the examples of the previous experiment.[7] For the experiments, we have considered the topological graph resulting from the 50-partitioning for the different log-model combinations. For the `NFWCC` algorithm, the table contains the number of non-fitting weakly connected components ($|C_C|$), the average size (places $|\bar{P}|$ and transitions $|\bar{T}|$) and average number of vertexes ($|\bar{V}|$) whose connected components are composed of. For the `NFN` algorithm, the table provides the size of the derived non-fitting net ($|P|$ and $|T|$), and the number of topology vertexes it includes. The table illustrates the benefits of the proposed algorithms 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., prCm6). 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.

The final experiment illustrates the effects of the simplification over the RPST decomposition on the number of components and hierarchy levels. Figure 7 shows the simplification of two models used in the previous experiments of this section. For each model, the figure provides the number of components (Y-axis) at each level of the RPST tree (being 1 the root of the RPST, and 14 the deepest level). The figure contains the number components for the original RPST, after removing the

[7] Only non-fitting models of Table 1 are considered in Table 2.

small components (less than 10 arcs), and after merging *similar* nested nodes (i.e., similarity degree over 0.8). Both charts reflect 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 13). On the other hand, the effect on the depth after merging similar nodes is high. In both cases, the number of levels of the tree is reduced significantly (from 12 to 6), providing a decomposition with less redundancy and more aligned with human perception [4].



(a) prCm6 (b) prDm6

Fig. 7. Effect of the simplification techniques

7 Conclusions and Future Work

The practical relevance of process mining increases as more event data becomes available. More and more events are being recorded and already today's event logs provide massive amounts of process related data. However, as event logs and processes become larger, many computational challenges emerge.

In this paper, we presented an approach that makes use of the well-known SESE and RPST analysis to decompose the problem of conformance checking and process diagnosis. The approach makes it possible to discover conformance problems more efficiently both in terms of computation and diagnostics. Although our experimental results support these claims, more real-life case studies need to be conducted. For example, we would like to empirically show that the diagnostics based on the topological graph are indeed more insightful because the analyst can focus on the problem spots. Moreover, the inclusion of other conformance dimensions into the approach, together with the possibility of tackling invisible and duplicate transitions, is another direction for future work.

Acknowledgments. The authors would like to thank Dr. Artem Polyvyanyy for his comments and help. This work has been partially supported by the Ministerio de Educación (AP2009-4959) and by the projects TIN-2011-22484 and TIN-2007-66523.

References

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

2. IEEE Task Force on Process Mining: Process Mining Manifesto. In: Daniel, F., Barkaoui, K., Dustdar, S. (eds.) Business Process Management Workshops. LNBIP, vol. 99, pp. 169–194. Springer (2012)
3. Polyvyanyy, A., Vanhatalo, J., Völzer, H.: Simplified computation and generalization of the refined process structure tree. In: Bravetti, M. (ed.) WS-FM 2010. LNCS, vol. 6551, pp. 25–41. Springer, Heidelberg (2011)
4. Munoz-Gama, J., Carmona, J., van der Aalst, W.M.P.: Hierarchical Conformance Checking of Process Models Based on Event Logs. In: Applications and Theory of Petri Nets (2013), TR: `http://www.lsi.upc.edu/~techreps/files/R13-5.zip`
5. Cook, J., Wolf, A.: Software Process Validation: Quantitatively Measuring the Correspondence of a Process to a Model. ACM Transactions on Software Engineering and Methodology 8(2), 147–176 (1999)
6. de Medeiros, A.K.A., Weijters, A.J.M.M., van der Aalst, W.M.P.: Genetic Process Mining: An Experimental Evaluation. Data Mining and Knowledge Discovery 14(2), 245–304 (2007)
7. Rozinat, A., van der Aalst, W.M.P.: Conformance checking of processes based on monitoring real behavior. Inf. Syst. 33(1), 64–95 (2008)
8. van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.: Replaying History on Process Models for Conformance Checking and Performance Analysis. WIREs Data Mining and Knowledge Discovery 2(2), 182–192 (2012)
9. Adriansyah, A., van Dongen, B.F., van der Aalst, W.M.P.: Conformance checking using cost-based fitness analysis. In: EDOC, pp. 55–64. IEEE Computer Society (2011)
10. Weerdt, J.D., Backer, M.D., Vanthienen, J., Baesens, B.: A Multi-Dimensional Quality Assessment of State-of-the-Art Process Discovery Algorithms Using Real-Life Event Logs. Information Systems 37(7), 654–676 (2012)
11. Adriansyah, A., Munoz-Gama, J., Carmona, J., van Dongen, B.F., van der Aalst, W.M.P.: Alignment Based Precision Checking. In: La Rosa, M., Soffer, P. (eds.) BPM Workshops 2012. LNBIP, vol. 132, pp. 137–149. Springer, Heidelberg (2013)
12. Munoz-Gama, J., Carmona, J.: Enhancing Precision in Process Conformance: Stability, Confidence and Severity. In: Chawla, N., King, I., Sperduti, A. (eds.) IEEE Symposium on Computational Intelligence and Data Mining (CIDM 2011), Paris, France, pp. 184–191. IEEE (April 2011)
13. van der Aalst, W.M.P.: Distributed Process Discovery and Conformance Checking. In: de Lara, J., Zisman, A. (eds.) FASE 2012. LNCS, vol. 7212, pp. 1–25. Springer, Heidelberg (2012)
14. van der Aalst, W.M.P.: Decomposing Process Mining Problems Using Passages. In: Haddad, S., Pomello, L. (eds.) PETRI NETS 2012. LNCS, vol. 7347, pp. 72–91. Springer, Heidelberg (2012)
15. van der Aalst, W.M.P.: Decomposing Petri Nets for Process Mining: A Generic Approach. BPMCenter.org BPM-12-20 (accepted for Distributed and Parallel Databases) (2012)
16. Murata, T.: Petri nets: Properties, analysis and applications. Proceedings of the IEEE **77**(4) () 77(4), 541–580 (1989)
17. Vanhatalo, J., Völzer, H., Koehler, J.: The refined process structure tree. Data Knowl. Eng. 68(9), 793–818 (2009)
18. Burattin, A., Sperduti, A.: Plg: A framework for the generation of business process models and their execution logs. In: Muehlen, M.z., Su, J. (eds.) BPM 2010 Workshops. LNBIP, vol. 66, pp. 214–219. Springer, Heidelberg (2011)