

# Flexible and Hierarchical Decomposition of Workflow Nets for Process Analysis

Tsung-Hao Huang<sup>1</sup>[0000-0002-3011-9999], Lukas M. Jansen<sup>2</sup>,  
Marco Pegoraro<sup>1</sup>[0000-0002-8997-7517], Gyunam Park<sup>3</sup>[0000-0001-9394-6513], and  
Wil M. P. van der Aalst<sup>1</sup>[0000-0002-0955-6940]

<sup>1</sup> Process and Data Science (PADS), RWTH Aachen University, Aachen, Germany

`{tsunghao.huang, pegoraro, wvdaalst}@pads.rwth-aachen.de`

<sup>2</sup> RWTH Aachen University, Aachen, Germany

`lukas.maximilian.jansen@rwth-aachen.de`

<sup>3</sup> Eindhoven University of Technology

`g.park@tue.nl`

**Abstract.** Decomposition is widely used in process mining to enhance scalability, facilitate focused analysis, and mitigate the cognitive complexity of large-scale models. While both process discovery and conformance checking benefit from decomposition, existing approaches—especially those based on single-entry single-exit (SESE) fragments—impose structural constraints that limit diagnostic value. This paper proposes a complementary decomposition technique that partitions a WF-net into isolated subnets, i.e., internally coherent fragments with clear interfaces. The decomposition is hierarchical, independent of SESE assumptions, and compatible with valid system net decompositions, enabling divide-and-conquer conformance checking. Experiments show that our approach improves alignment runtime and yields fitness approximations that are closer to the true values than existing decomposition methods.

**Keywords:** Process mining · Conformance checking · Decomposition · Process diagnosis

## 1 Introduction

Decomposition is a common strategy in process mining to improve scalability and enable focused analysis on large or structurally complex models. By splitting a model into smaller fragments, tasks such as discovery and conformance checking can be performed more efficiently and with reduced overhead [3]. This is particularly valuable for real-world processes with many activities, intricate routing, and large event logs.

*Problem* Current decomposition techniques based on the Refined Process Structure Tree (RPST) and SESE blocks help localize analysis and lower computational cost [18]. While every WF-net can be transformed into an RPST, SESE-based decomposition imposes strong structural boundaries. WF-nets with

multi-entry or multi-exit constructs cannot be meaningfully grouped, resulting in overly fine-grained or trivial fragments. As illustrated in Fig. 1a, edges 1–7 and 16–22 cannot be combined into larger SESE components and therefore remain as isolated fragments in the RPST (Fig. 1c). This leads to fragmented decompositions and additional transformations for visualization [18].

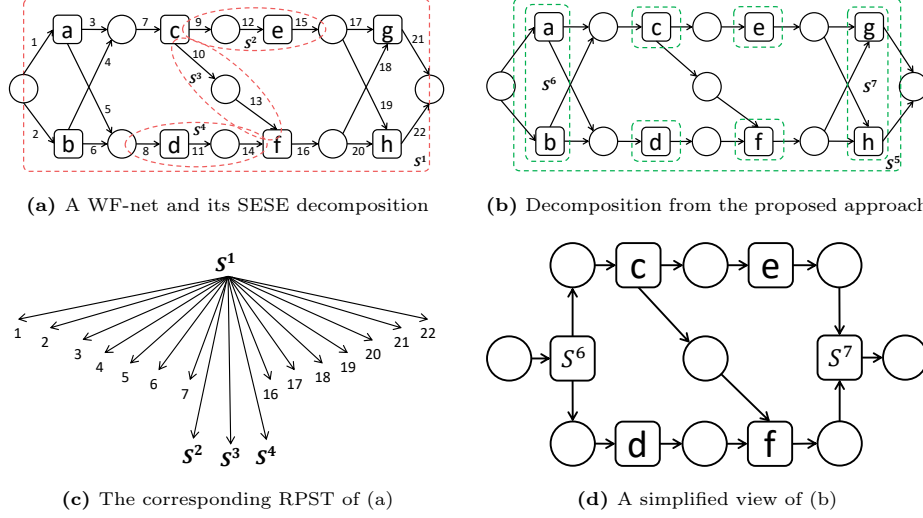


**Fig. 1:** Comparison of two different decomposition methods.

*Solution & Contribution* We introduce a decomposition method based on *isolated subnets*—transition-bordered, structurally coherent fragments that capture subprocesses of the WF-net. As shown in Fig. 1b, the approach yields meaningful control-flow units even for nets where SESE methods are less effective. The decomposition is naturally hierarchical: subnets can be collapsed for abstraction (Fig. 1d) or expanded for detailed analysis. The resulting fragments retain key behavioral properties and integrate with valid system net decomposition [3], enabling correct decomposed conformance checking.

*Results* We show that the produced subnets preserve soundness and free-choiceness and support valid decomposed conformance checking. Experiments on a diverse set of models demonstrate that, despite decomposition overhead, our method achieves better runtime and produces fitness estimates closer to the global optimum than existing decomposition strategies.

*Outline* Section 2 reviews related work. Section 3 introduces preliminaries. Section 4 details the method. Section 5 reports experiments. Section 6 concludes the paper.

## 2 Related Work

For an overview of process mining, we refer to [4]. In this paper, we focus on decomposition approaches to enable scalable process mining, covering both process discovery and conformance checking. Process discovery aims to automatically derive a process model from an event log, while conformance checking quantifies the alignment between observed (event logs) and modeled behavior. It identifies deviations, assesses fitness, precision, and generalization, and provides diagnostics to refine models or correct executions.

Given the growing scale and complexity of real-world processes, decomposition is increasingly necessary to make process mining computationally feasible. [2] and [3] propose general decomposition techniques for process mining, particularly targeting process discovery and conformance checking. In [3], it is shown that conformance checking can be decomposed, provided that the resulting fragments share only uniquely labeled transitions. The general decomposition strategy primarily aims to improve scalability. However, previous work argues that it is not well suited to provide additional diagnostic insights due to fragmented components of maximal decomposition [18]. Moreover, decomposed conformance checking is known to provide an over-optimistic approximation of the fitness metric [3]. Later, the recomposition [15] approach was proposed to overcome the problem. Additionally, it is used to isolate subprocesses and repair them to improve the overall model quality [21].

To support both improved diagnostics and faster conformance checking, the Refined Process Structure Tree (RPST) [19] has been used in various studies [18,16] to construct a hierarchy of single entry single exit fragments (SESE). Rather than evaluating the entire process model against the full event log, this method checks alignments between specific subprocesses—defined by SESE fragments—and their corresponding sublogs. These SESE fragments serve as fragments with well-defined interfaces, enabling localized conformance analysis and process comparison [6]. However, as the name implies, this approach implicitly assumes that the process model is made up of single-entry single-exit blocks. While RPST can decompose any WF-net in theory, the resulting fragments may be less informative when the input WF-net includes arbitrary structures such as multi-entry or multi-exit blocks. In contrast, our approach imposes no such structural constraints on the input net; it automatically decomposes the WF-net into subnets with self-contained control flow. Additionally, the resulting decomposition is naturally hierarchical, enabling more detailed diagnostic analysis.

Decomposition techniques for multiple-entry-multiple-exit (MEME) fragments have been proposed in works such as [22,11]. However, the resulting decompositions do not easily translate into valid decompositions as defined in [3], limiting their applicability for decomposed conformance checking. Later, we demonstrated that our proposed decomposition can be extended to satisfy the criteria of a valid decomposition [3], thereby enabling the use of decomposed conformance checking.

### 3 Preliminaries

This section presents essential concepts used throughout the paper, starting with fundamental notations, followed by definitions related to event logs and Petri net-based concepts.

For any given set  $A$ ,  $\mathcal{B}(A)$  represents the set of all multisets over  $A$ . If  $M \in \mathcal{B}(A)$ , then  $M(a)$  denotes the number of occurrences of  $a \in A$  in the multiset  $M$ . For example, if we consider the set  $A = \{w, x, y, z\}$  and the multiset  $M = [w^2, x^4, y^3, z^5]$ , we have  $M(y) = 3$  because  $y$  appears 3 times in  $M$ . The notation  $\sigma \in A^*$  is used to represent a sequence over some set  $A$ . For a sequence  $\sigma = \langle a_1, a_2, a_3, \dots, a_n \rangle$ , the length of the sequence is denoted by  $|\sigma| = n$ . For each  $1 \leq i \leq |\sigma|$ , the element  $\sigma(i) = a_i \in A$  refers to the  $i$ -th element of the sequence. Given two sequences  $\sigma$  and  $\sigma'$ , their concatenation is written as  $\sigma \cdot \sigma'$ .

Next, we introduce the projection function. Given a set  $A$  and a subset  $X \subseteq A$ , a projection function  $\downarrow_X$  takes an object (such as a set, multiset, or sequence) over  $A$  and returns a corresponding object over  $X$  by filtering out elements that do not belong to  $X$ .

For a set  $A$ , the projection onto  $X$  is  $A \downarrow_X = \{a \in A \mid a \in X\}$ , i.e., only elements from  $X$  are retained. For a multiset  $M$  over  $A$ , the projection is  $M \downarrow_X(a) = M(a)$  if  $a \in X$  and 0 otherwise, preserving multiplicities of elements in  $X$ . For a sequence  $\sigma \in A^*$ , projection is defined recursively:  $\langle \rangle \downarrow_X = \langle \rangle$ , and for  $\langle a \rangle \cdot \sigma$ ,  $(\langle a \rangle \cdot \sigma) \downarrow_X = \langle a \rangle \cdot (\sigma \downarrow_X)$  if  $a \in X$ , and  $\sigma \downarrow_X$  otherwise, ensuring that elements of  $X$  keep their relative order while others are removed.

**Definition 1 (Activities, Traces, and Logs).** *Let  $\mathcal{A}$  be the universe of activities. A trace  $\sigma \in \mathcal{A}^*$  is a sequence of activities. A log  $L \in \mathcal{B}(\mathcal{A}^*)$  is a multiset of traces.*

**Definition 2 (Petri Net).** *A Petri net is a tuple  $N = (P, T, F)$ , where  $P$  is the set of places,  $T$  is the set of transitions,  $P \cap T = \emptyset$ ,  $F \subseteq (P \times T) \cup (T \times P)$  is the set of arcs. A labeled Petri net  $N = (P, T, F, l)$  is a Petri net with a labeling function  $l \in T \rightarrow \mathcal{A}$ , where  $\mathcal{A}$  is some universe of activity labels. If a transition is not in the domain of  $l$ , it is called a silent transition.*

For any  $x \in P \cup T$ ,  $\overset{N}{\bullet}x = \{y \mid (y, x) \in F\}$  denotes the set of input nodes (preset) of  $x$  and  $x\overset{N}{\bullet} = \{y \mid (x, y) \in F\}$  denotes the set of output nodes (postset) of  $x$ . The superscript  $N$  is dropped if it is clear from the context.

**Definition 3 (Marking, Marked Net).** *Let  $N = (P, T, F)$  be a Petri net. A marking  $M \in \mathcal{B}(P)$  is a multiset of places.  $(N, M)$  denotes a marked net.*

A place  $p \in P$  is marked at a marked net  $(N, M)$  if  $M(p) > 0$ . A marked place is visually noted by one or more black dots (depending on the cardinality) called tokens. A transition  $t \in T$  is enabled in  $M$ , denoted as  $(N, M)[t]$ , if every input place of  $t$  is marked in  $M$ , i.e.,  $\forall p \in \bullet t, M(p) > 0$ .  $(N, M)[t](N, M')$  denotes that  $t$  is enabled and the resulting marking  $M'$  of firing  $t$  is  $M' = (M \setminus \bullet t) \cup t\bullet$ .

Let  $\sigma \in T^*$  be a sequence of transitions.  $(N, M)[\sigma](N, M')$  denotes that there exists a set of marking  $M_1, M_2, \dots, M_{n+1}$  such that  $M_1 = M, M_{n+1} = M'$ , and  $(N, M)[\sigma(i)](N, M')$  for  $1 \leq i \leq n$ . A marking  $M'$  is reachable from  $M$  if  $\exists \sigma \in T^* (N, M)[\sigma](N, M')$ . We denote the set of all reachable markings as  $R(N, M) = \{M' \in \mathcal{B}(P) \mid \exists \sigma \in T^* (N, M)[\sigma](N, M')\}$ .

**Definition 4 (Subnet).** Let  $N = (P, T, F, l)$  and  $N' = (P', T', F', l')$  be Petri nets.  $N'$  is a subnet of  $N$  if  $P' \subseteq P$ ,  $T' \subseteq T$ ,  $F' = F \cap ((P' \times T') \cup (T' \times P'))$ , and  $l' = l|_{T'}$ .

**Definition 5 (Free-choice Net).** Let  $N = (P, T, F)$  be a Petri net.  $N$  is a free-choice net if for any  $t, t' \in T$ :  $\bullet t = \bullet t'$  or  $\bullet t \cap \bullet t' = \emptyset$ .

Free-choice nets have separate constructs for choices and synchronizations, as any two transitions either have the same preset or do not share any places in their presets.

In business processes, we are often interested in a specific type of Petri net, namely the Workflow net (WF-net). WF-nets have well-defined structures and properties, such as soundness, to avoid apparent anomalies.

**Definition 6 (Workflow Net (WF-net)).** Let  $N = (P, T, F)$  be a Petri net.  $N$  is a workflow net if it has a dedicated source place  $i \in P$ :  $\bullet i = \emptyset$  and a dedicated sink place  $o \in P$ :  $o \bullet = \emptyset$ . Moreover, every node  $x \in P \cup T$  is on some path between  $i$  and  $o$ .

A workflow net  $N = (P, T, F)$  with source  $i$  and sink  $o$  is **sound** if it is safe (no place holds more than one token), always able to reach  $o$  from any marking, and has no dead transitions. For details, see [1].

**Definition 7 (System Net).** A system net is a tuple  $SN = (N, M_{init}, M_{final})$  where  $N$  is a Petri net.  $M_{init}$  and  $M_{final}$  denote the initial and final marking.

A WF-net  $N$  can be directly connected to a system net  $SN = (N, [i], [o])$  with its designated source and sink places. A log  $L$  is perfectly fitting  $SN$  if and only if every trace of  $L$  perfectly fits  $SN$ . In practice, logs rarely align perfectly with models, making conformance checking essential. Instead of a binary check, it quantifies behavioral deviations. Alignment-based techniques,<sup>4</sup> the gold standard, map log traces to model executions using synchronous (matching) and asynchronous (log- or model-only) moves.<sup>5</sup> Optimal alignments minimize user-defined deviation costs and enable conformance metrics like fitness.

<sup>4</sup> Our approach does not rely on alignments but uses them to illustrate benefits.

<sup>5</sup> A precise definition of conformance checking and alignment approach is out of scope. For details, see [5,8].

## 4 Approach

We first introduce the concept of an isolated subnet. Then, we detail how to use the concept of an isolated subnet to decompose a WF-net using ILP. Lastly, we connect the proposed decomposition to the concept of valid decomposition, as defined in [3], to decompose conformance checking.

**Definition 8 (Isolated Subnet [13,14]).** Let  $N = (P, T, F, l)$  be a WF-net. Let  $i$  be the source place, i.e.,  $i \in P : \bullet i = \emptyset$  and  $o$  be the sink place, i.e.,  $o \in P : o \bullet = \emptyset$ .  $N_s = (P_s, T_s, F_s, l_s)$  is an isolated subnet of  $N$  if

1.  $N_s$  is a subnet of  $N$
2. there exist  $P_{in} \subseteq P \setminus P_s, P_{out} \subseteq P \setminus P_s, T_{start} \subseteq T_s, T_{end} \subseteq T_s$  such that
  - (a)  $\forall t \in T_{start} (\bullet t = P_{in})$ , the preset of every transition in  $T_{start}$  equals  $P_{in}$ .
  - (b)  $\forall t \in T_{end} (t \bullet = P_{out})$ , the postset of every transition in  $T_{end}$  equals to  $P_{out}$ .
  - (c)  $\forall p \in P_s (\bullet p \cup p \bullet \subseteq T_s)$ , all places in  $P_s$  only connect to transition in  $T_s$ .
  - (d)  $\forall t \in T_s \setminus T_{start} (\bullet t \subseteq P_s)$ , except for  $T_{start}$ , the input places of every transition in  $T_s$  are in  $P_s$ .
  - (e)  $\forall t \in T_s \setminus T_{end} (t \bullet \subseteq P_s)$ , except for  $T_{end}$ , the output places of every transition in  $T_s$  are in  $P_s$ .
  - (f)  $\{i, o\} \cap P_s = \emptyset$ , the source and sink places are not in the subnet.

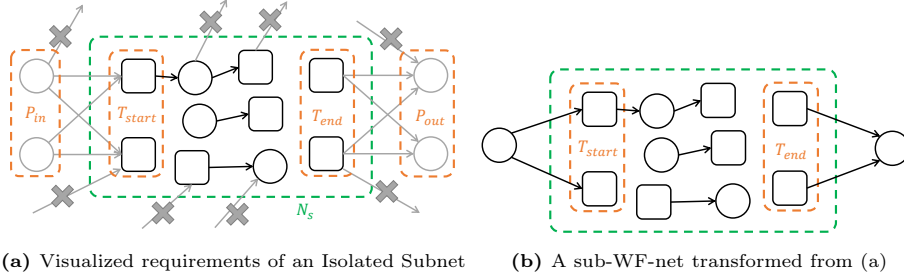


**Fig. 2:** An Isolated Subnet and its corresponding sub-WF-net

As in Def. 8, the requirements for an isolated subnet are visualized in Fig. 2a. In this figure,  $N_s$  satisfies all requirements, with the only incoming connections passing through the start transitions  $T_{start}$  and the only outgoing connections through the end transitions  $T_{end}$ . All other nodes within  $N_s$  remain internally connected, without external incoming/outgoing arcs. Note that any transition in a WF-net is an isolated subnet by definition, as well as the whole net (excluding the source and the sink places  $i$  and  $o$ ). We refer to these two types of isolated subnets as *minimal* and *maximal trivial subnets* respectively.

As shown in Fig. 2b, isolated subnets can be transformed into WF-nets. The WF-nets can be analyzed in isolation to enable focus analysis. A convenient aspect of sub-WF-nets derived from isolated subnets is that they inherit both soundness and free-choiceness from the original WF-net. Soundness helps rule out anomalies, while free-choice nets offer efficient analysis backed by strong theory [9]. Recent advances also suggest the potential for more efficient conformance checking algorithms for sound free-choice WF-nets [20].

#### 4.1 Decomposition

The proposed decomposition aims to partition a WF-net into isolated, non-overlapping subnets that collectively cover all transitions of the original net. Formally, we define it as follows:

**Definition 9 (Decomposition).** Let  $N = \{P, T, F, l\}$  be a labeled WF-net and  $D = \{S^1, S^2, \dots, S^n\}$  be a set of isolated subnets of  $N$ .  $D$  is a decomposition of  $N$  if:  $l^i = l|_{T^i}$ ; the subnets in  $D$  are pairwise disjoint, meaning  $(T^i \cap T^j = \emptyset) \wedge (P^i \cap P^j = \emptyset)$  for  $1 \leq i < j \leq n$ ; and the union of all the transitions in the isolated subnets equals the set of transitions in  $N$ , i.e.,  $\cup_{1 \leq i \leq n} T^i = T$ .

A decomposition is called trivial if  $|D| = 1$  or  $|D| = |T|$ . Fig. 3 shows the running example of this paper, where Fig. 3a and Fig. 3a are the input and output, respectively. In Fig. 3a,  $\{S^1, S^6, S^7, S^8, S^9, S^{10}, S^{11}, S^{12}, S^{14}, S^{15}, S^{16}\}$  is trivial, as well as  $\{S^0\}$ .

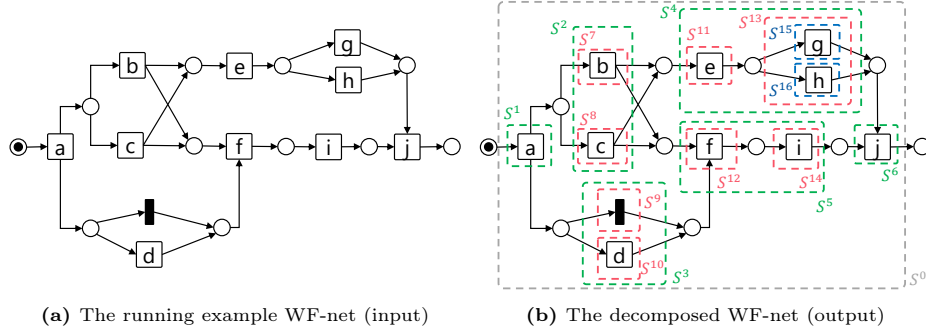


**Fig. 3:** The running example.

This definition is intentionally broad to allow flexibility in different applications, which may result in multiple valid decompositions. For the use case in this paper, we are interested in decompositions that produce subnets suitable for diagnostic purposes. Specifically, we seek non-trivial decompositions where the resulting subnets are as large as possible. These subnets may be further decomposed, enabling a hierarchical structure. In practice, the assignment can be computed using ILP.

#### 4.2 ILP Formulation for Isolated Subnet Assignment

Given a WF-net  $N = (P, T, F)$  with source and sink places  $i, o \in P$  ( $\bullet i = \emptyset$ ,  $o \bullet = \emptyset$ ) and a set  $T_{init} \subseteq T$ , the goal is to extract a maximal isolated subnet  $N_s = (P_s, T_s, F_s)$  containing all transitions in  $T_{init}$  without including all of  $T$ .

**Decision Variables:**

$$x_t, y_p, s_t, e_t, u_p, v_p \in \{0, 1\}$$

where  $x_t$  and  $y_p$  indicate inclusion of transitions and places;  $s_t, e_t$  mark start/end transitions; and  $u_p, v_p$  mark input/output places.

**Objective:**

$$\max \sum_{t \in T} x_t \quad (1)$$

**Constraints<sup>6</sup>:**

$$x_t = 1 \quad \forall t \in T_{init} \quad (2)$$

$$x_t = 0 \quad \forall t \in T_{excluded} \quad (3)$$

$$y_p \leq \sum_{t: (p,t) \in F \vee (t,p) \in F} x_t \quad \forall p \in P \quad (4)$$

$$s_t \leq x_t, \quad x_t \leq s_t + y_p \quad \forall t \in T, \forall p \in \bullet t \quad (5)$$

$$e_t \leq x_t, \quad x_t \leq e_t + y_p \quad \forall t \in T, \forall p \in t \bullet \quad (6)$$

$$y_i = 0, \quad y_o = 0 \quad (7)$$

$$u_p \geq s_t - y_p, \quad u_p \leq 1 - y_p \quad \forall t \in T, \forall p \in \bullet t \quad (8)$$

$$v_p \geq e_t - y_p, \quad v_p \leq 1 - y_p \quad \forall t \in T, \forall p \in t \bullet \quad (9)$$

$$1 \geq y_p + u_p + v_p \quad \forall p \in P \quad (10)$$

$$\sum_{t \in T} x_t < |T| \quad (11)$$

$$y_p \leq x_t \quad \forall p \in P, \forall t \in \bullet p \cup p \bullet \quad (12)$$

$$u_p \leq I_{pt}, \quad u_p \leq s_t, \quad u_p \geq s_t - (1 - I_{pt}) \quad \forall t, p \quad (13)$$

$$v_p \leq O_{tp}, \quad v_p \leq e_t, \quad v_p \geq e_t - (1 - O_{tp}) \quad \forall t, p \quad (14)$$

*Explanation of the Formulation* The ILP maps Definition 8 to linear constraints as follows:

- **Subnet Logic:** Eqs. (2)–(3) handle transition membership. Eq. (12) enforces Condition 2c by requiring all transitions incident to an internal place ( $y_p = 1$ ) to be included ( $x_t = 1$ ).
- **Boundary Identification:** Eqs. (5)–(6) identify  $T_{start}$  and  $T_{end}$  by detecting included transitions linked to external places (Conditions 2d and 2e).
- **Interface Consistency:** Eqs. (8)–(14) use indicator constants  $I_{pt}$  and  $O_{tp}$  to ensure boundary transitions share identical presets/postsets, satisfying Conditions 2a and 2b.
- **Structural Constraints:** Eq. (7) enforces Condition 2f, and Eq. (10) ensures exclusive place roles. Objective (1) maximizes transitions while Eq. (11) prevents the maximal trivial isolated subnet.

To obtain a decomposition as defined in Definition 9, we iteratively apply the ILP solver [17] using this formulation to assign each transition to an isolated subnet.

To better illustrate the idea and provide an overview, Fig.4 presents an overview of the decomposition, specifically visualizing the first two iterations.

<sup>6</sup> The presented constraints are non-exhaustive to avoid trivial edge cases; the tool implements the full specification.

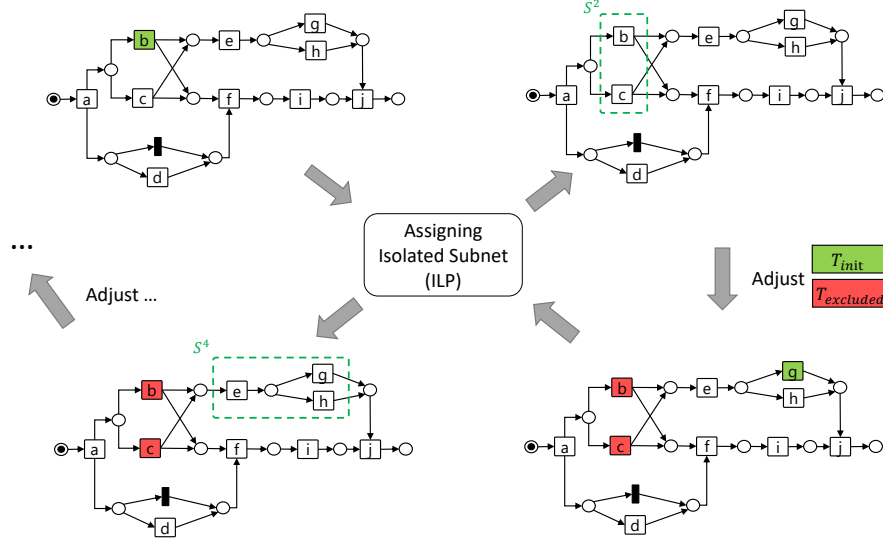


**Fig. 4:** An overview of the proposed decomposition with the running example.

At each iteration, we begin by assigning  $T_{init}$  and  $T_{excluded}$ . The set  $T_{init}$  is initialized with the transition that has the highest connectivity, i.e., the highest sum of incoming and outgoing arcs<sup>7</sup>. In the running example, the transition labeled  $b$  is selected. Transitions that were already assigned in previous iterations are excluded and placed in  $T_{excluded}$ . For the first iteration,  $T_{excluded}$  is empty. Next, the ILP solver described in Sect. 4.2 identifies the maximal non-trivial isolated subnet for transition  $b$ , which is  $S^2$ , including transitions labeled  $b$  and  $c$ . These two transitions are then placed in  $T_{excluded}$  for the next iteration.

Since each isolated subnet can be transformed into a WF-net, the decomposition can be applied recursively to build a hierarchy of subnets until each transition can only be assigned to the minimal trivial subnet (i.e., the transition itself). Fig. 3b shows the resulting decomposition of the running example.

**Iterative Approach, Complexity, and Determinism** We employ an iterative greedy strategy rather than a global optimization because finding an optimal global partition can be considered as a variation of the NP-hard Balanced Graph Partitioning problem. A single global ILP is computationally intractable for large-scale industrial WF-nets; however, our local maximization approach provides a scalable and practical compromise. To guarantee reproducibility, transitions are selected as seeds ( $T_{init}$ ) via a deterministic ordering (e.g., lexicographical or by connectivity), and previously assigned transitions are added to  $T_{excluded}$  to ensure a disjoint partition of  $T$ .

<sup>7</sup> If multiple transitions share the same connectivity, one is selected at random.

The proposed decomposition procedure aims to bridge two objectives: structural coherence for process diagnosis and formal validity for decomposed conformance checking. While isolated subnets (Def. 8) are designed to be naturally self-contained sub-WF-nets, the theory of valid decompositions [3] requires specific conditions to ensure the soundness of decomposed conformance checking results. The procedure introduced in the following subsection ensures the resulting fragments satisfy such theoretical requirements.

### 4.3 Connection to Valid Decomposition

Generally speaking, a valid decomposition [3] is a partition of a system net where subnets only share uniquely labeled transitions<sup>8</sup>. The union of the subnets constitutes the system net. Formally, we define the valid decomposition as follows.

**Definition 10 (Valid Decomposition [3]).** *Let  $SN = (N, M_{init}, M_{final})$  be a system net where  $N = (P, T, F, l)$  is a Petri net.  $D = \{SN^1, SN^2, \dots, SN^n\}$  is a valid decomposition of  $SN$  if and only if for all  $1 \leq i \leq n$ :  $SN^i = (N^i, M_{init}^i, M_{final}^i)$  is a system subnet with  $N^i = (P^i, T^i, F^i, l^i)$  where  $l^i = l|_{T^i}$  and  $SN = \bigcup_{1 \leq i \leq n} SN^i$ ; and for all  $1 \leq i < j \leq n$ : the subnets are essentially disjoint, meaning  $(P^i \cap P^j = \emptyset) \wedge (F^i \cap F^j = \emptyset) \wedge (T^i \cap T^j \subseteq T_v^u(N))$ .*

In [3], it is proven that a valid decomposition has a useful property: a log fits a system net if and only if it fits all subnets. This property implies that conformance checking can be divided into several smaller problems, i.e., a divide-and-conquer approach. Moreover, it is guaranteed that the aggregated cost of subnets' alignments never exceeds the optimal overall alignment's cost [3]. This makes it a useful optimistic estimate: it provides an upper bound for the overall fitness and a lower bound for the total costs. To take advantage of this, we have to extend the proposed decomposition in the previous section.

Comparing Def. 9 and Def. 10, we see that the pieces missing for the proposed decomposition to be valid are the places connecting the subnets and the corresponding arcs. To achieve this, we first introduce a helper function.

**Definition 11 (Place-induced Subnet).** *Let  $N = (P, T, F, l)$  be a labeled Petri net and let  $p \in P$  be a place. The place-induced subnet of  $p$  in  $N$ , denoted by  $\mathbf{p}\text{-Subnet}_N(p)$ , is the labeled Petri net  $N' = (P', T', F', l')$  defined as follows:  $P' = \{p\}$  (containing only the place  $p$ );  $T' = \bullet p \cup p \bullet$  (containing all transitions directly connected to  $p$ );  $F' = \{(t, p) \in F \mid t \in \bullet p\} \cup \{(p, t) \in F \mid t \in p \bullet\}$  (containing all arcs linking  $p$  to its adjacent transitions); and  $l' = l|_{T'}$  (the restriction of the labeling function to these transitions). This function can be extended to a set of places  $P' \subseteq P$  by applying it element-wise, i.e.,  $\mathbf{p}\text{-Subnet}_N(P') = \{\mathbf{p}\text{-Subnet}_N(p) \mid p \in P'\}$ .*

<sup>8</sup> Note this requirement can be fulfilled naturally in our approach by forcing the duplicated labels to be in the  $T_{init}$  as defined in the ILP constraints.

In other words, the  $\mathbf{p}\text{-Subnet}_N(p)$  consists of the place  $p$ , all transitions directly connected to  $p$ , the arcs between them, and the restriction of the labeling function to those transitions.

**Definition 12 (Extended Decomposition).** Let  $N = (P, T, F, l)$  be a labeled WF-net, and let  $D = \{S^1, S^2, \dots, S^n\}$  be a decomposition of  $N$  as defined in Def. 9. Let  $P^c = P \setminus \left(\bigcup_{1 \leq i \leq n} P^i\right)$  be the set of places that are not in any subnets in  $D$ . The extended decomposition of  $N$  is defined as  $D \cup \mathbf{p}\text{-Subnet}_N(P^c)$ .

Using the example in Fig. 3b, the extended decomposition would be  $\{S^1, S^2, \dots, S^6\} \cup \{C^1, C^2, \dots, C^9\}$  in Fig. 5.

**Definition 13 (Induced Valid Decomposition).** Let  $SN = (N, M_{init}, M_{final})$  be a system net with  $N = (P, T, F, l)$  a labeled WF-net. Let  $\mathcal{D}_N^+$  denote the extended decomposition of  $N$ . The induced valid decomposition of  $SN$ , denoted by  $\mathcal{V}_N$ , is obtained as follows:

- For any subnets that share a silent transition  $t \in (T^i \cap T^j) \cap (T \setminus \text{dom}(l))$ , merge them into a single subnet  $S^{ij} = (P^i \cup P^j, T^i \cup T^j, F^i \cup F^j, l^i \cup l^j)$ .
- Remove any subnet that is strictly contained in another subnet, i.e., for  $N^i = (P^i, T^i, F^i, l^i)$  and  $N^j = (P^j, T^j, F^j, l^j)$  in the merged set, if  $P^i \subset P^j$ ,  $T^i \subset T^j$ , and  $F^i \subset F^j$ , then discard  $N^i$ .
- For each remaining subnet  $N^i = (P^i, T^i, F^i, l^i)$ , construct a system subnet  $SN^i = (N^i, M_{init}^i, M_{final}^i)$ , where  $l^i = l|_{T^i}$ ,  $M_{init}^i = [p \in P^i \mid \bullet p = \emptyset]$ , and  $M_{final}^i = [p \in P^i \mid p \bullet = \emptyset]$ .
- Let  $\mathcal{V}_N := \{SN^i \mid 1 \leq i \leq m\}$  be the resulting set of system subnets.

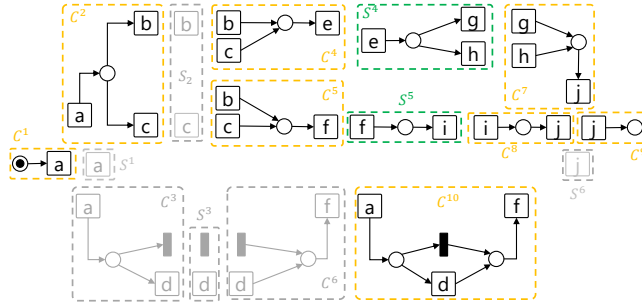


**Fig. 5:** A valid decomposition of the WF-net in Fig. 3b

The induced valid decomposition of the net in Fig. 3b is shown in Fig. 5. The grayout subnets are either a subset of another subnet or share silent transitions, and are thus removed. With the connection to valid decomposition, the proposed approach can utilize the benefits of decomposed conformance checking.

## 5 Experimental Evaluation

We conduct several experiments to evaluate the effectiveness of the proposed approach<sup>9</sup> when applying to decomposed conformance checking. We begin by introducing the dataset and then discuss the setups and results.

**Dataset** To generate WF-nets with arbitrary structures, we apply the synthesis rules from [9,12,14] to create six models (labeled [A/E] [100/200/300]) with 100, 200, and 300 transitions, and simulate them to obtain event logs. To better reflect real-world scenarios, we introduce three types of noise: *insertion*, *swapping*, and *deletion*. *Insertion* adds a random activity from the alphabet  $\mathcal{A}$  at a random position; *swapping* exchanges two activities; *deletion* removes one. Noise is applied randomly per trace—some remain noise-free, and others may contain one or more noise types multiple times. We also include model-log pairs (`pr[A-G]m6`, `pr-1151`, `pr-1908`) from [18], mostly generated using PLG [7], with 36–430 transitions. Lastly, the BPI challenge 2015 (BPIC15\_ [1–5]) [10], one of the few publicly available real-world data sets at a larger scale, is also included. Full statistics are available in the experiment repository.

**Scalability** A key benefit of decomposed conformance checking is improved scalability. We compare alignment runtimes for four approaches: general decomposition [3], SESE decomposition [18], our proposed method, and non-decomposed conformance checking [5], as shown in Figure 6. For decomposed methods, runtime is split into *Decomposing Time* (computing a valid decomposition) and *Aligning Time* (performing alignment on the decomposition).

*Decomposing Time* General decomposition yields the fastest decomposition time, as expected from its scalability-focused design—though at the cost of overly optimistic fitness, as discussed later. The decomposing time of our approach depends on the ILP solver and grows with model size (e.g., around 2s for A100, 9s for A200 and 27s for A300, showing a more than 4× increase when doubling transitions initially and even more after that).

*Aligning Time* All decomposed approaches outperform non-decomposed alignment across all synthetic datasets. However, no single decomposed method consistently outperforms the others. Alignment is faster when the traces fit the model well, as seen in `prBm6` (perfect fit) and `prAm6` (77% fitting traces). The decomposed algorithms benefit from the reduced complexity, as shown by the much lower computation time. In addition to model size and log noise, the average trace length is a major factor: logs that exceed the time limit often contain particularly long traces. One thing worth noticing is the performance<sup>10</sup> of the

<sup>9</sup> The implementation and experimental artifacts are available at <https://git.rwth-aachen.de/tsunghao.huang/flexibledecomposition4pm>

<sup>10</sup> Note that the bars are cut off at  $10^5$ .

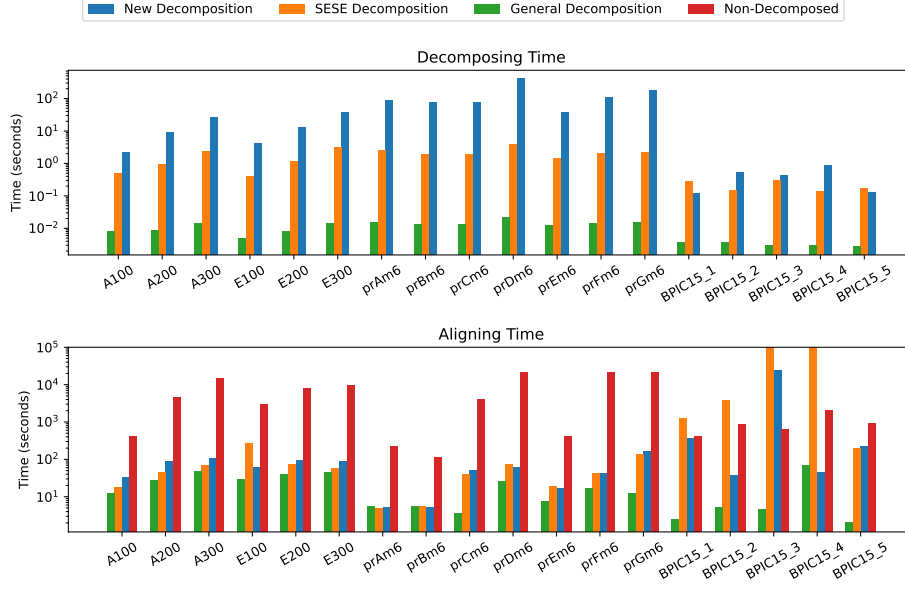


**Fig. 6:** Runtime comparison of three decomposition approaches and the non-decomposed alignment.

SESE decomposition on most of the real-world event logs BPIC15\_[1-4]. For these models, the presence of silent transitions makes the alignments of the decomposed approach larger, and therefore, they take longer to compute. The SESE decomposition is particularly impacted as silent transitions are on every fragment of the SESE components (implying they need to be merged according to Definition 10). In contrast, the decomposed components of the proposed approach are much smaller. In general, while our decomposition scales less effectively than others, it still delivers notable runtime gains over the non-decomposed baseline, highlighting a direction for future improvement.

**Impact of Trace Length & Noise** This experiment examines how trace length and log noise affect computation time, comparing the proposed decomposition with the non-decomposed variant. To assess trace length impact, we use logs with varying average lengths. For each, a perfectly fitting (noise-free) version is also included to evaluate the effect of noise. Results are shown in Figure 7.

The proposed decomposition consistently achieves the fastest runtimes, regardless of trace length or noise. In noise-free cases, the non-decomposed approach performs only slightly worse. However, with noise, its performance degrades significantly, whereas the decomposed method shows only a minor slowdown. This contrast becomes more significant as the average trace length increases.

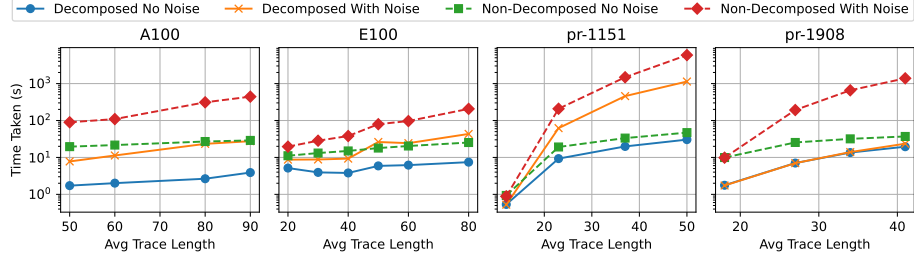


**Fig. 7:** Impact of trace length and noise on the runtime.

**Impact of Subnet Size** Since conformance checking time also depends on subnet size, we examine its impact on performance. The ILP solver can restrict subnet size, ensuring only isolated subnets within the limit are identified. Results for four model-log pairs are shown in Figure 8. Smaller subnet sizes generally lead to faster computation, as individual alignment tasks are simpler due to reduced net and trace complexity. The trade-off is more frequent ILP invocations, as fewer transitions are covered per iteration. However, in this experiment, this did not increase overall runtime. Note that the speed gain comes at the cost of lower fitness accuracy, which tends to be over-approximated. This trade-off is examined further in the next experiment.

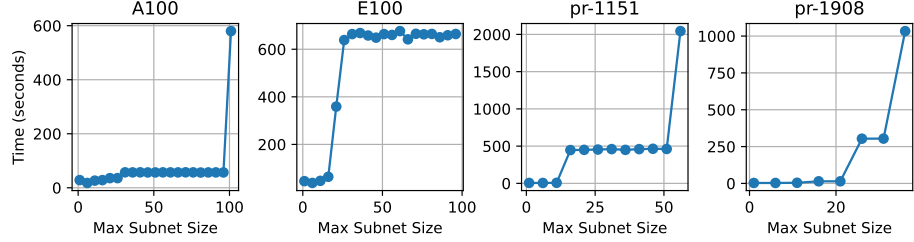


**Fig. 8:** Comparing the impact of the maximum subnet size on the runtime of the decomposed fitness-checking algorithm.

**Approximation of Fitness** Decomposed conformance checking guarantees that a log fits the system net iff it fits all subnets. However, if a trace is non-fitting, it can lead to overestimated fitness due to missing global constraints within subnets [3]. This experiment investigates the fitness discrepancy between decomposed methods and the "true fitness" from non-decomposed alignment.

Figure 9 compares overall fitness deviations, demonstrating that our approach achieves values closer to true fitness than the SESE decomposition. We believe this is due to the ability of our approach to group nodes that are fragmented for SESE decomposition, as discussed in Fig. 1.

Figure 10 shows that fewer subnets generally yield fitness values closer to true fitness, similarly for both SESE and our proposed decomposition, as expected

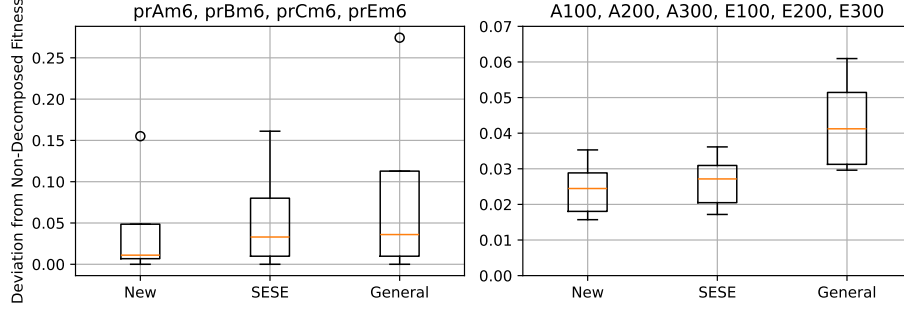


**Fig. 9:** Comparing the overall deviation of fitness.

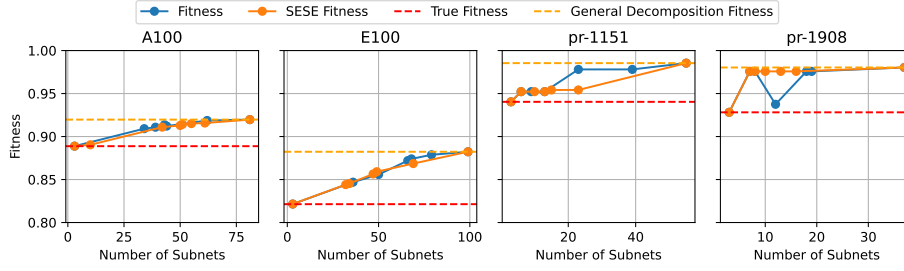


**Fig. 10:** Comparing the impact of the number of subnets on the fitness. The red line represents the fitness value acquired by non-decomposed alignment.

from [3]. Since the general decomposition fixes the number of subnets internally, its fitness appears as a dashed line.

## 6 Conclusion

We presented a decomposition approach based on isolated subnets that enables flexible, hierarchical partitioning of WF-nets with clear interfaces. Applied to conformance checking, it supports targeted diagnostics on a broader range of models and remains compatible with valid decompositions. The large-scale experiments showed improved fitness approximations compared to existing decomposed alignments, alongside better runtime than non-decomposed alignments.

**Limitations and Outlook** The method introduces higher computational overhead than SESE techniques due to repeated ILP solving, prioritizing structural precision over raw efficiency. Furthermore, the approach requires validation on more real-world logs, as large-scale public data ( $\approx 300$  activities) is currently unavailable to our best knowledge. Future work will focus on reducing ILP overhead by approximating, pre-processing, and developing user-adaptive decomposition features. Despite these limitations, our technique offers a practical foundation for analyzing complex models beyond SESE constraints.

## References

1. van der Aalst, W.M.P.: The application of Petri nets to workflow management. *J. Circuits Syst. Comput.* **8**(1), 21–66 (1998)
2. van der Aalst, W.M.P.: Decomposing process mining problems using passages. In: *Petri Nets. Lecture Notes in Computer Science*, vol. 7347, pp. 72–91. Springer (2012)
3. van der Aalst, W.M.P.: Decomposing Petri nets for process mining: A generic approach. *Distributed Parallel Databases* **31**(4), 471–507 (2013)
4. van der Aalst, W.M.P., Carmona, J. (eds.): *Process Mining Handbook, Lecture Notes in Business Information Processing*, vol. 448. Springer (2022)
5. Adriansyah, A.: *Aligning Observed and Modeled Behavior*. Ph.D. thesis, Technische Universiteit Eindhoven (2014)
6. Brockhoff, T., Gose, M.N., Uysal, M.S., van der Aalst, W.M.P.: Process comparison using petri net decomposition. In: *Petri Nets. Lecture Notes in Computer Science*, vol. 14628, pp. 83–105. Springer (2024)
7. Burattin, A., Sperduti, A.: PLG: A framework for the generation of business process models and their execution logs. In: *Business Process Management Workshops. Lecture Notes in Business Information Processing*, vol. 66, pp. 214–219. Springer (2010)
8. Carmona, J., van Dongen, B.F., Weidlich, M.: Conformance checking: Foundations, milestones and challenges. In: *Process Mining Handbook, Lecture Notes in Business Information Processing*, vol. 448, pp. 155–190. Springer (2022)
9. Desel, J., Esparza, J.: *Free Choice Petri Nets*. No. 40, Cambridge university press (1995)
10. van Dongen, B.B.: Bpi challenge 2015 (2015). <https://doi.org/10.4121/UUID:31A308EF-C844-48DA-948C-305D167A0EC1>
11. Hauser, R., Friess, M., Küster, J.M., Vanhatalo, J.: An incremental approach to the analysis and transformation of workflows using region trees. *IEEE Trans. Syst. Man Cybern. Part C* **38**(3), 347–359 (2008)
12. Huang, T., van der Aalst, W.M.P.: Discovering sound free-choice workflow nets with non-block structures. In: *EDOC. Lecture Notes in Computer Science*, vol. 13585, pp. 200–216. Springer (2022)
13. Huang, T., Schneider, E., Pegoraro, M., van der Aalst, W.M.P.: Fast & sound: Accelerating synthesis-rules-based process discovery. In: *BP-MDS/EMMSAD@CAiSE. Lecture Notes in Business Information Processing*, vol. 511, pp. 259–274. Springer (2024)
14. Huang, T., Schneider, E., Pegoraro, M., van der Aalst, W.M.P.: *Flexible and sound: Synthesis Miner. Software and systems modeling* (2025)
15. Lee, W.L.J., Verbeek, H.M.W., Munoz-Gama, J., van der Aalst, W.M.P., Sepúlveda, M.: Recomposing conformance: Closing the circle on decomposed alignment-based conformance checking in process mining. *Inf. Sci.* **466**, 55–91 (2018)
16. de Leoni, M., Munoz-Gama, J., Carmona, J., van der Aalst, W.M.P.: Decomposing alignment-based conformance checking of data-aware process models. In: *OTM Conferences. Lecture Notes in Computer Science*, vol. 8841, pp. 3–20. Springer (2014)
17. Maher, S.J., Miltenberger, M., Pedroso, J.P., Rehfeldt, D., Schwarz, R., Serrano, F.: Pyscipopt: Mathematical programming in python with the SCIP optimization suite. In: *ICMS. Lecture Notes in Computer Science*, vol. 9725, pp. 301–307. Springer (2016)

18. Munoz-Gama, J., Carmona, J., van der Aalst, W.M.P.: Single-entry single-exit decomposed conformance checking. *Inf. Syst.* **46**, 102–122 (2014)
19. Polyvyanyy, A., Vanhatalo, J., Völzer, H.: Simplified computation and generalization of the refined process structure tree. In: WS-FM. *Lecture Notes in Computer Science*, vol. 6551, pp. 25–41. Springer (2010)
20. Schwanen, C.T., Pakusa, W., van der Aalst, W.M.P.: Complexity of alignments on sound free-choice workflow nets. In: *Petri Nets. Lecture Notes in Computer Science*, vol. 15714. Springer (2025)
21. Tikhonov, S.E., Mitsyuk, A.A.: A method to improve workflow net decomposition for process model repair. In: AIST. *Lecture Notes in Computer Science*, vol. 11832, pp. 411–423. Springer (2019)
22. Zerguini, L.: A novel hierarchical method for decomposition and design of workflow models. *Trans. SDPS* **8**(2), 65–74 (2004)