Collaborative Model Optimization in Federated Stochastic Process Discovery

Hootan Zhian[1], Rajkumar Buyya[1], and Artem Polyvyanyy[1]

The University of Melbourne, Australia
hzhian@student.unimelb.edu.au
{rbuyya;artem.polyvyanyy}@unimelb.edu.au

Abstract. Process discovery focuses on constructing process models that accurately represent system behavior from event logs generated by these systems. Traditional methods rely on centralized event logs, which pose significant challenges in distributed environments where organizations are reluctant to share raw data due to privacy concerns. Federated process discovery has emerged as a promising approach to enabling collaborative model generation without compromising data privacy. However, existing federated approaches often produce overly complex process models, limiting their practical utility. This paper introduces an approach that enables organizations to collaboratively construct, via iterative refinements, a global process model from local models discovered from their own data. The iterations are guided by feedback from organizations on the quality of the current global model. By formulating process model refinement as a decision-tree pruning problem, our approach systematically reduces unnecessary model complexity while ensuring high model quality. Experiments with industrial logs show that our approach often yields smaller, more accurate models than a baseline federated method, comparable in quality to models discovered from centralized logs, and has runtime comparable to that of the centralized discovery approach.

Keywords: Stochastic process mining, federated process discovery, cross-silos process discovery

1 Introduction

With the rapid digitalization of organizational operations, information systems increasingly record data in the form of event logs, which capture detailed information about operational activities. This trend has elevated the importance of process mining as a means of extracting knowledge from event logs to discover, monitor, and improve processes. In particular, process discovery aims to automatically construct descriptive process models from such logs [2, 19]. These models are often represented as Directly-Follows Graphs (DFGs) [3]. A DFG is a directed graph whose nodes describe actions and whose edges capture "can-follow" ordering relations between actions. Numeric weights associated with nodes and edges indicate the number of observations of the corresponding actions and subsequent executions of the corresponding actions in the event log, respectively. Numeric weights associated with nodes and edges indicate, respectively, the number of observed occurrences of the corresponding actions and the number of times one action is directly followed by another in the event log.

To evaluate the quality of a discovered model with respect to an event log, several evaluation criteria have been proposed [10]. Recall, also known as fitness, reflects how well the model reproduces the behavior recorded in the log. Precision evaluates the extent to which the model avoids allowing unsupported behavior beyond what is observed in the log. Generalization captures the ability of the model to avoid overfitting by allowing plausible yet unseen behavior. Finally, simplicity measures the structural complexity of the model, favoring compact representations without unnecessary elements.

A plethora of process discovery algorithms have been proposed to construct models with good quality characteristics. Recent studies have explored optimization-based approaches, including metaheuristic techniques for automated process discovery based on single-objective and multi-objective search [8, 9]. Yan et al. [25] propose an approach that optimizes imprecise sub-processes using frequency-based filtering. In addition, multi-objective optimization strategies have been applied to discover Pareto-optimal models that balance quality dimensions such as model size and entropic relevance [5, 29]. Entropic relevance is a stochastic conformance measure that compares the likelihood of observing traces in event logs and stochastic process models, capturing how closely a model reflects the observed probability distribution of behavior [20]. Moreover, it offers a balanced assessment of model quality by jointly reflecting precision and recall [6]. However, these approaches generally assume centralized access to event logs. Cross-organizational settings pose significant challenges for process discovery because organizations are often reluctant to share event logs [4, 16].

Federated process discovery (FPD) has been proposed to overcome the limitations of centralized discovery, including privacy and security concerns [13, 16, 18, 21], data transfer costs [16, 24], and scalability and efficiency challenges [1, 11, 24, 27]. Zhian et al. [28] introduce the *FedGASPD* approach, in which organizations, or *clients*, share models discovered locally from their own data, referred to as *local models*, rather than raw event logs. A central server then aggregates these local models to derive a *global model* that describes the overall process executed across the clients. This strategy increases the level of abstraction of the shared information, thereby improving privacy and reducing data transfer requirements. *FedGASPD* produces models of quality comparable to those obtained from centralized event logs. Because discovery is performed locally on smaller event logs and the aggregation procedure is efficient, *FedGASPD* can often construct models faster than centralized discovery. However, the models discovered by *FedGASPD* may still become unnecessarily complex, as aggregating complete local models can preserve infrequent or client-specific behavior. This limitation highlights the need for further improving the quality of federated discovery results.

This paper presents *OptFedGASPD*, in which clients collectively optimize the global process model constructed by *FedGASPD* using feedback on the global model quality with respect to clients' event logs. The approach systematically prunes the aggregated global model, reducing unnecessary structural complexity while ensuring model quality for each participating organization. Its key innovation lies in enabling collaborative optimization of the global model under the coordination of a central server by casting the pruning task as a decision-tree optimization problem. Experimental results on industrial event logs show that *OptFedGASPD* often produces smaller and more accurate models than *FedGASPD*. Although pruning introduces additional computation, the overall run-

time remains comparable to that of *FedGASPD* and often still outperforms discovery from centralized event logs using *GASPD*.

Specifically, the evaluation gives positive answers to these two research questions.

RQ1 Can clients, under central coordination, systematically optimize the quality and size of the global process model constructed by *FedGASPD*?

RQ2 Is *OptFedGASPD* comparable to *FedGASPD* in terms of runtime performance and often faster than *GASPD*?

The remainder of the paper is structured as follows. Section 2 surveys related work on federated process discovery. Section 3 introduces basic concepts used in the subsequent discussions. Section 4 presents *OptFedGASPD*. Section 5 discusses the evaluation of the proposed method using an open-source implementation over publicly available industrial event logs. Finally, Section 6 concludes the paper by summarizing the contributions and outlining directions for future research.

2 Related Work

Decomposition and divide-and-conquer strategies have long been explored as alternatives to centralized process discovery methods. These approaches aim to address scalability and complexity challenges by breaking down event logs into smaller, more manageable components. Carmona et al. [11] propose methods to derive partial behavioral views from event logs, either by leveraging region theory to construct and combine local model components or by partitioning logs into tightly related event groups and discovering models for each partition. These models can then be assembled into a global process model or analyzed independently. Similarly, van der Aalst [1] introduces a generic framework for divide-and-conquer process discovery, which partitions actions, projects logs onto these partitions, and composes the resulting models. This framework employs strategies such as single-entry-single-exit (SESE) fragments to facilitate partitioning. Yan et al. [26] rely on this idea to propose RPSTHD, a multi-level decomposition approach that combines the Refined Process Structure Tree (RPST), heuristic mining, and Integer Linear Programming (ILP) to efficiently analyze large event logs by breaking them into smaller parts. Andersen et al. [7] propose EdgeMiner as a distributed process mining approach that avoids centralizing event logs by performing computation directly at data sources (e.g., sensor nodes). Instead of sharing raw event data, each node locally observes events, determines predecessor–successor relations, and constructs a partial footprint matrix (FM) of these relations. To generate a global view, a central entity collects and merges these partial FMs, forming a complete FM equivalent to one derived from a centralized event log. This global FM is then used as input to process discovery algorithms to extract the overall process model.

The concept of federated process discovery was introduced by van der Aalst [4], which aims to enable organizations to collaboratively perform process discovery without sharing raw event data. Rafiei et al. [22] introduce a privacy-aware federated conformance-checking method designed to assess the correctness of cross-organizational process models while safeguarding sensitive information. Two main strategies for sharing information during federated process discovery are sharing filtered event data and

sharing abstract representations of event data. Limited works on merging abstract data representations into a global model. Rennert et al. [23] develop a method for discovering DFGs from federated event logs without requiring a trusted third party or direct data sharing. Their approach uses fully homomorphic encryption to securely compute a global DFG, ensuring that sensitive information remains protected. Rafiei and van der Aalst [21] propose a privacy-preserving federated process discovery by introducing an abstraction-based method that uses directly-follows relations to represent event log abstractions. This method employs a handover mechanism to capture trace transitions between organizations, enabling collaboration while limiting exposure to sensitive data. Nucciarelli et al. [18] introduce a federated adaptation of the Alpha discovery algorithm. The focus is on enabling privacy-preserving process discovery in healthcare. Each client computes an FM, similar to the approach by Andersen et al. [7]. A master node then combines the FMs from the clients in a way that preserves the ordering relations and ensures consistency in the aggregated FM. The aggregated FM is subsequently used to construct the final process model, which is captured as a Petri net. In other efforts, federated process discovery approaches have also been proposed to reduce communication overhead and improve scalability. For example, Khan et al. [16] design a protocol for federated process discovery using dependency graphs and a trusted central server. Rojo et al. [24] explore federated process discovery in distributed environments, such as smartphone networks, by either discovering models locally or aggregating raw traces centrally. While local discovery reduces communication costs, the quality of the resulting global model often suffers from aggregating intermediate local models.

Zhian et al. [28] introduce *FedGASPD*, a federated *stochastic process discovery* approach designed to address privacy and scalability challenges of centralized discovery. The models constructed by *FedGASPD* describe traces and the likelihoods of observing them. Unlike other approaches that focus on assembling local event logs or intermediate abstractions, *FedGASPD* merges local process models on the server node, reducing the need for raw data sharing and intermediate model aggregation. The approach builds on the *GASPD* algorithm, which applies grammatical inference techniques to extract stochastic language models from event logs [5].

3 Preliminaries

GASPD uses the *ALERGIA* grammatical inference algorithm to learn accurate stochastic representation of the traces in the input event log [5]. The language models constructed by *ALERGIA* are Deterministic Frequency Finite Automata (DFFAs) [12].

Definition 3.1 (Deterministic Frequency Finite Automata)
A *Deterministic Frequency Finite Automaton* (DFFA) is a tuple $(Q, \Lambda, q_0, \mathbb{I}, \mathbb{F}, \delta)$, where:
- Q is a finite nonempty set of *states*;
- Λ is a finite set of *actions*, called the *alphabet*;
- $\mathbb{I} : Q \to \mathbb{N}$ is the *initial state frequency function*, with one *initial state* $q_0 \in Q$ for which it holds that $\mathbb{I}(q_0) > 0$ and for all $q \in Q$, $q \neq q_0$, it holds that $\mathbb{I}(q) = 0$;
- $\mathbb{F} : Q \to \mathbb{N}$ is the *final state frequency function*; and
- $\delta : Q \times \Lambda \times Q \to \mathbb{N}$ is the *transition frequency function*, such that $\forall q \in Q \, \forall a \in \Lambda \, \forall q' \in Q \, \forall q'' \in Q : ((\delta(q, a, q') > 0 \wedge \delta(q, a, q'') > 0) \Rightarrow (q' = q''))$. ⌟

The notation $\delta(q, a, q') = n$ indicates that there is a transition from state q to state q' labeled a occurring n times. Figure 1 shows a DFFA with states $\{p_0, \ldots, p_3\}$, initial state p_0, $\mathbb{I}(p_0) = 20$, $\mathbb{F} = \{(p_0, 0), (p_1, 10), (p_2, 5), (p_3, 5)\}$, and $\delta(p_0, a, p_1) = 10$, $\delta(p_0, b, p_2) = 10$, $\delta(p_1, b, p_2) = 2$, $\delta(p_2, a, p_1) = 2$,and $\delta(p_2, b, p_3) = 5$.

Algorithm 1 summarizes the *FedGASPD* procedure. The algorithm constructs local models for each event log in L (lines 1–3) and then merges them into a global DFFA (line 4). The resulting DFFA can be converted into a Stochastic Directed Action Graph (SDAG), a DFG-like model that supports duplicate action nodes. Multiple local DFFAs are discovered independently (in parallel) at each client using the *GASPD* algorithm. Then, the best models are selected for each client using the *SELECTDFFA* procedure and sent to the server, which constructs the list of best local models (line 3); note that "$\circ$" is the list concatenation operation. Finally, the best models from clients are merged on the server into the resulting DFFA using the *MergeDFFAs* function on line 4.

Algorithm 1: *FedGASPD* [28]

Input: Initial population size p, number of generations m, number of parents k to generate offspring, and a non-empty list of event logs $L = \langle L_1, \ldots, L_n \rangle$
Output: A DFFA discovered from L

1 $M \leftarrow \langle\rangle$;
2 **parallelfor** $i \in [1 \,..\, n]$ **do**
3 $\quad M \leftarrow M \circ \langle SELECTDFFA\,(GASPD(p, m, k, L_i))\rangle$;
4 **return** $MergeDFFAs(M)$;

To implement merging, *FedGASPD* relies on the unfolding of DFFAs. Figure 3 shows a prefix of the unfolding of the DFFA in Figure 2. The construction starts from the initial state of the input automaton and proceeds iteratively. At each step, a state in the current prefix is selected and extended using one of its outgoing transitions in the input automaton. Each such extension introduces a fresh state representing the target of the chosen transition, ensuring that the unfolding has a tree structure. To guide the construction, only states with unexplored outgoing transitions are selected for expansion. If the target state of a transition has already appeared d (depth) of times among the ancestors of the current node in the prefix, the expansion is stopped; otherwise, if the input DFFA contains a cycle, the unfolding procedure would continue indefinitely. The automaton in Figure 3 is constructed for the depth of two ($d = 2$).

The frequencies of states and transitions in the prefix automaton are computed by multiplying the corresponding frequencies in the input automaton by the probability of reaching the respective occurrence during execution. This probability is estimated from the state and transition frequencies of the input automaton.

Although *FedGASPD* effectively addresses privacy and scalability concerns, the resulting models may still exhibit unnecessary complexity. The following example illustrates this issue. Consider two event logs: $L_1 = [\langle a\rangle^8, \langle b\rangle^3, \langle a, b\rangle^2, \langle b, a\rangle^2, \langle b, b\rangle^5]$ and $L_2 = [\langle a\rangle^4, \langle a, a\rangle^4, \langle a, a, b\rangle^2, \langle b\rangle^{10}]$. Figure 1 shows A_1, the DFFA discovered from event log L_1, while Figure 2 shows A_2, the DFFA discovered from event log L_2.

The merging process described in *FedGASPD* [28] combines these local models into an aggregated model, A^{agg}, shown in Figure 4, by embedding a prefix unfolding of A_2, shown in Figure 3, into A_1. After the merging process, the resulting model A^{agg} contains states that are not visited by traces from one or both event logs.

For instance, the states $\{q_3, q'_1, q'_3, q''_1\}$ are not visited with respect to L_1, while the states $\{q'_3, q''_1, p_3\}$ are not visited with respect to L_2. Moreover, q_3 represents behavior that appears only in L_2, p_3 represents behavior that appears only in L_1, and q'_3 and q''_1 represent behavior that appears in neither L_1 nor L_2. These states not only increase the structural complexity of the aggregated model but also reduce its overall quality [17]. To address this issue, our proposed approach enables clients to collaboratively prune the aggregated model, thereby obtaining a more accurate and compact representation.

Figure 5 shows an SDAG equivalent to Figure 4. In *FedGASPD*, DFFAs are used as intermediate stochastic representations during local model discovery and model aggregation, because they provide a convenient formalism for capturing trace frequencies and supporting operations such as unfolding and merging. However, the final process models constructed by *FedGASPD* are Stochastic Directed Action Graphs (SDAGs). An SDAG extends the DFG proposed in [5] by allowing duplicate actions, thereby enabling a more faithful representation of stochastic process behavior. The procedure for translating a DFFA into an SDAG is described in the original *GASPD* paper [5]. Thus, while DFFAs serve as the formal basis for inference and aggregation, SDAGs provide the final process representation used for interpretation and analysis.

4 Approach

FedGASPD aggregates local models into a global process model without requiring event log sharing. However, it often produces overly complex aggregated models. This complexity arises from directly merging diverse local models, which introduces unvisited states that do not meaningfully contribute to the overall behavior of the aggregated model. To address this challenge, we propose *OptFedGASPD*, an optimized federated stochastic process discovery approach that systematically refines the aggregated model by pruning unnecessary states. Unlike *FedGASPD*, which terminates once the aggregated model is constructed, *OptFedGASPD* introduces a collaborative pruning phase in which clients, under the orchestration of a central server, jointly optimize the aggregated model. The pruning decisions are based solely on model-level information and model quality measurements, thereby revealing no additional information about the event data.

Next, Section 4.1 describes the pruning procedure, while Section 4.2 embeds the pruning into the overall *OptFedGASPD* discovery routine.

4.1 DFFA Pruning

The DFFA pruning problem is closely related to classical decision-tree pruning under accuracy constraints, where the goal is to remove subtrees while ensuring the resulting model does not exceed a prescribed classification error threshold. The computational complexity of related optimal pruning problems has been rigorously studied. Hyafil and Rivest [15] showed that constructing optimal binary decision trees is NP-complete,

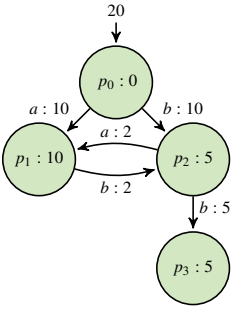


Fig. 1: DFFA A_1

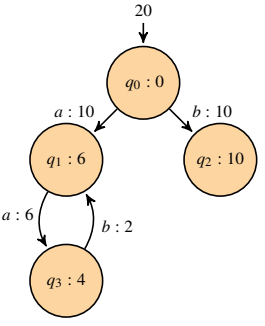


Fig. 2: DFFA A_2

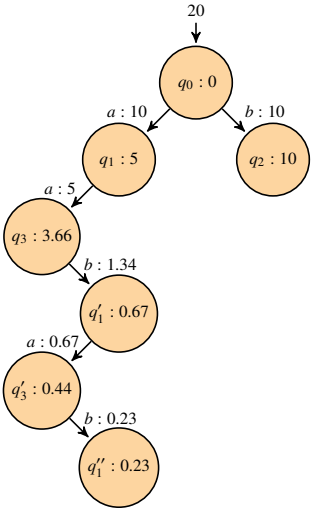


Fig. 3: DFFA A_2^u

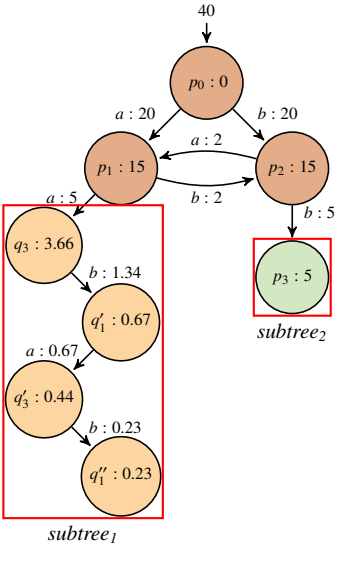


Fig. 4: DFFA A^{agg}

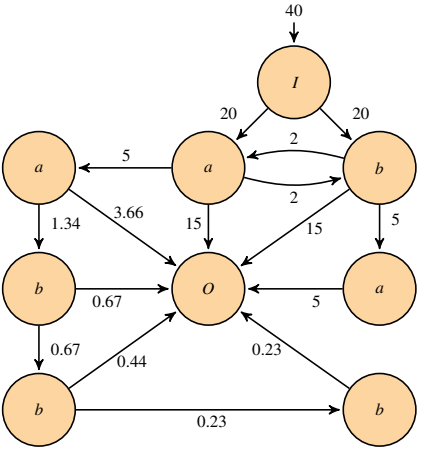


Fig. 5: SDAG A^{agg}

while more recent work has established that optimal decision-tree pruning under accuracy constraints is NP-hard [14]. Our problem has a similar combinatorial structure, requiring the selection of which parts of a tree-shaped or unfolded model to retain while satisfying quality constraints. However, DFFA pruning further involves global quality requirements across multiple clients.

Our DFFA pruning approach focuses on states in subtrees within the model. To identify the subtree states of a DFFA, we view the automaton as a directed graph whose nodes are the states and whose edges represent transitions with positive frequency. We then look for subgraphs that have the shape of a directed rooted tree. In such a subgraph, one state is selected as the root, every other state has exactly one parent within the subgraph, and all states in the subgraph are reachable from the root.

We only consider tree-shaped subgraphs that are attached to the rest of the DFFA through their root. The root must have one incoming edge connecting it to a state outside the tree-shaped subgraph. At the same time, no other state in the tree-shaped subgraph may have an incoming or outgoing edge to a state outside the subgraph. Thus, the root is the only interface between the tree-shaped part and the rest of the automaton.

Finally, we retain only maximal such subgraphs. That is, a tree-shaped subgraph is kept only if it cannot be extended with additional states while preserving the same properties. In this context, maximality is understood with respect to the attached branch itself: the root is the boundary state through which the tree-shaped branch is connected to the rest of the DFFA, while the predecessor of the root remains outside the subtree.

In the DFFA shown in Fig. 4, two maximal attached tree-shaped subgraphs are identified. The first one, denoted by $subtree_1$, consists of the states q_3, q'_1, q'_3, and q''_1. These states form a directed chain rooted at q_3. The root q_3 is connected to the rest of the DFFA through the transition from p_1 to q_3, while none of the other states in the subtree has any connection to states outside the subtree. Thus, q_3 is the only interface between $subtree_1$ and the rest of the model. The second maximal attached tree-shaped subgraph, denoted by $subtree_2$, consists only of the state p_3. This is a singleton tree whose root is p_3. The state p_3 is connected to the rest of the DFFA through the transition from p_2 to p_3. The states p_0, p_1, and p_2 are not included in these tree-shaped subgraphs. They form the surrounding non-tree part of the DFFA, in particular because p_1 and p_2 are connected by transitions in both directions.

We denote by Q_T the set of all states that belong to at least one maximal tree-shaped subgraph identified as described above. For the example automaton in Fig. 4, it holds that $Q_T = \{q_3, q'_1, q'_3, q''_1, p_3\}$. Let $\mathrm{Anc}(q)$ denote the set of states $p \in Q_T$ such that p and q belong to the same maximal tree-shaped subgraph and q is reachable from p by a non-empty directed path within that subgraph. In the example DFFA in Fig. 4, it holds that $\mathrm{Anc}(q_3) = \emptyset$, $\mathrm{Anc}(q'_1) = \{q_3\}$, $\mathrm{Anc}(q'_3) = \{q_3, q'_1\}$, $\mathrm{Anc}(q''_1) = \{q_3, q'_1, q'_3\}$, and $\mathrm{Anc}(p_3) = \emptyset$. Although q_3 is reachable from p_1 and p_3 is reachable from p_2 in the full DFFA, we have $p_1 \notin \mathrm{Anc}(q_3)$ and $p_2 \notin \mathrm{Anc}(p_3)$.

The objective is to maximize the total number of pruned states in the DFFA model. The decision variable x_q, cf. Constraint C1, indicates whether state q is removed during pruning; a value of 1 means that q is removed, while a value of 0 means that q is preserved after pruning. The decision variable h_q, see Constraint C2, indicates whether the pruning effect of state q is subsumed by the pruning of one of its ancestors. A value of 1 means that an ancestor of q is pruned, and therefore the effect of pruning q is ignored. A value of 0 means that no ancestor of q is pruned. Constraint C3 states that if an ancestor state q' is selected for pruning, then all states in the subtree below q', including q, must also be selected for pruning. Constraint C4 states that if any ancestor of state q, denoted by q', is selected for pruning, then the effect on entropic relevance

associated with pruning q is ignored. Constraint C5 ensures that the topmost selected state in a pruned subtree contributes its effect on the entropic relevance of the pruned model. Finally, Constraint C6 imposes quality constraints for clients, ensuring that the degradation caused by pruning remains below the threshold τ for each client. Here, ER_c denotes the entropic relevance of the model discovered by *FedGASPD* with respect to the event log of client c, before any states are pruned. $ERC_{c,q}$ denotes the change in the entropic relevance of the model with respect to the event log of client c when the subtree rooted at state q is pruned. The parameter τ is the maximum acceptable relative degradation in entropic relevance for the pruned model.

$$x_q \in \{0,1\} \qquad q \in Q_T \tag{C1}$$

$$h_q \in \{0,1\} \qquad q \in Q_T \tag{C2}$$

$$x_q \geq x_{q'} \qquad q \in Q_T, q' \in \mathrm{Anc}(q) \tag{C3}$$

$$h_q \geq x_{q'} \qquad q \in Q_T, q' \in \mathrm{Anc}(q) \tag{C4}$$

$$h_q \leq \sum_{q' \in \mathrm{Anc}(q)} x_{q'} \qquad q \in Q_T \tag{C5}$$

$$\tau \geq \frac{\sum_{q \in Q_T} ERC_{c,q} \cdot (x_q - h_q)}{ER_c} \qquad c \in C \tag{C6}$$

The objective function is defined as follows. The optimization problem aims to prune as many tree states as possible while keeping the relative degradation in entropic relevance within the threshold τ for every client. In this way, the final model balances compactness with fidelity to local client behavior and the preservation of a meaningful global representation.

$$\max_{x,h} \quad \sum_{q \in Q_T} x_q \tag{OF}$$

To solve this MILP, we use Google OR-Tools (`com.google.ortools`), an optimization toolkit that provides solvers for integer and linear programming problems. Figure 6 shows the DFFA model obtained by pruning the A^{agg} model in Figure 4 with $\tau = 0.1$. The corresponding SDAG model is shown in Fig. 7.

The SDAGs in Figs. 5 and 7 have sizes 25 and 19, respectively. Thus, the optimized SDAG is smaller while preserving the same entropic relevance values, namely 2.103 and 1.820, with respect to the corresponding client logs.

4.2 Optimized Federated Stochastic Process Discovery

Algorithm 2 summarizes the *OptFedGASPD* procedure. The algorithm begins by invoking *FedGASPD* (cf. Algorithm 1) in line 1 to construct the aggregated model A^{agg}. Given a set of local event logs of clients, the server requests each client to perform process discovery on its local event log. Each client applies *GASPD* to discover Pareto-optimal models. From these Pareto-optimal models, one model is selected and returned to the server. The server then merges the received local models to produce A^{agg}. Once A^{agg} has been constructed, the optimization phase of *OptFedGASPD* begins.

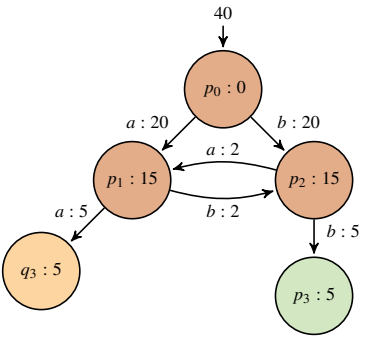


Fig. 6: DFFA A^{opt}

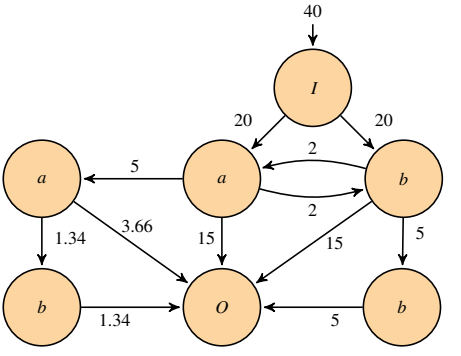


Fig. 7: SDAG A^{opt}

In line 2, the algorithm identifies the set of subtree states Q_T using the approach introduced in Section 4.1. For example, in Fig. 4, a merged model constructed from the event logs L_1 and L_2 introduced in Section 3, two subtrees are identified: one with the root state q_3 and another with the root state p_3. The subtree rooted at p_3 consists of the single state p_3. Therefore, it holds that $Q_T = \{q_3, q'_1, q'_3, q''_1, p_3\}$.

Algorithm 2: *OptFedGASPD*

Input: Initial population size p, number of generations m, number of parents k to generate offspring, entropic relevance degradation threshold τ, and a non-empty list of clients $C = \langle c_1, \ldots, c_n \rangle$

Output: A^{opt}, an optimized DFFA obtained by pruning the aggregated DFFA constructed from the local event logs of the clients in C

1 $A^{agg} \leftarrow FedGASPD(p, m, k, C.logs)$;
2 $Q_T \leftarrow GetSubtreeStates(A^{agg})$;
3 $ER \leftarrow \emptyset$;
4 **parallelfor** $c \in C$ **do**
5 $\quad ER_c \leftarrow c.EvaluateDFFA(A^{agg})$;
6 $\quad ERC \leftarrow \{(ER_c, \emptyset)\}$;
7 $\quad$ **for** $q \in Q_T$ **do**
8 $\quad\quad A \leftarrow PruneDFFAState(A^{agg}, q)$;
9 $\quad\quad ERC_{c,q} \leftarrow c.EvaluateDFFA(A) - ER_c$;
10 $\quad\quad ERC \leftarrow ERC \cup \{(ERC_{c,q}, q)\}$;
11 $\quad ER \leftarrow ER \cup \{(ERC, c)\}$;
12 $A^{opt} \leftarrow PruneDFFA(A^{agg}, Q_T, ER, \tau)$;
13 **return** A^{opt};

In line 3, the algorithm initializes the container *ER*, which stores the changes in entropic relevance for each client resulting from the pruning of each state in Q_T in A^{agg}. This container is populated by each client in the for loop of lines 4–11. Each

client computes the corresponding values independently, and the server collects the resulting client-specific containers into *ER*. In the iteration of this for loop for client c, first, the *EvaluateDFFA* function is invoked in line 5 to compute the entropic relevance of A^{agg} to the local event log of client c. The notation $c.EvaluateDFFA(A^{agg})$ denotes the evaluation of automaton A^{agg} with respect to the local event log stored at client c. For example, ER_c, of the A^{agg} DFFA in Fig. 4 is 2.103 for event log L_1 at client c_1 and 1.820 for L_2 at client c_2. In line 6, the container *ERC* is initialized, which stores the baseline entropic relevance and changes of entropic relevance values associated with pruning states in Q_T. The pair $(ER_c, \emptyset)$ represents the entropic relevance before pruning. Then, in the loop of lines 7–10, for each state in Q_T, the change in entropic relevance caused by its pruning is computed. First, the pruned automaton A is constructed in line 8. The function *PruneDFFAState* constructs the automaton obtained by pruning state q and the subtree rooted at q. Then, the change in entropic relevance $ERC_{c,q}$ is established in line 9; a positive change indicates a degradation in the model's quality. Finally, in line 10, the change value is stored in the *ERC* container. For example, the change values associated with pruning states in Q_T of the running example are $\left\{(0, q_3), \left(0, q_1'\right), \left(0, q_3'\right), \left(0, q_1''\right), (2.377, p_3)\right\}$ for event log L_1 at client c_1 and $\left\{(2.843, q_3), \left(2.819, q_1'\right), \left(0, q_3'\right), \left(0, q_1''\right), (0, p_3)\right\}$ for event log L_2 at client c_2. The entry $(2.377, p_3)$ for L_1 indicates that pruning state p_3 in A^{agg} increases the entropic relevance from 2.103 to 2.103 + 2.377 = 4.480, resulting in a degradation of model quality of 2.377. In contrast, removing p_3 does not affect client c_2 as the corresponding degradation in entropic relevance is zero. After processing all states in Q_T, the container *ERC* of the client is added to *ER* in line 11.

Once all clients have computed the effects of candidate pruning operations on entropic relevance, the server solves the optimization problem introduced in Section 4.1 to identify the states to be pruned and then applies the pruning to construct the automaton A^{opt}, refer to line 12. The resulting automaton is the final output of *OptFedGASPD*. Considering the running example, the resulting automaton is shown in Fig. 6. Finally, A^{opt} can be converted into the SDAG representation shown in Figure 7 [5].

To construct this result, the $\tau = 0.1$ threshold was used. When $\tau = 0.1$, the total degradation for client c_1 must not exceed $0.1 \times 2.103 \approx 0.210$, while for client c_2 it must not exceed $0.1 \times 1.820 = 0.182$. Consider the state $p_3 \in Q_T$. The degradation caused solely by pruning p_3 is 2.377, which exceeds the allowable threshold. Therefore, p_3 is not pruned. Similarly, pruning q_3 or q_1' would violate the degradation threshold. Thus, for $\tau = 0.1$, the set of pruned states is $\{q_3', q_1''\}$. These states can be pruned because their removal is not expected to increase the entropic relevance for either client.

5 Evaluation

This section presents an evaluation of *OptFedGASPD* and compares it with its predecessors, *FedGASPD* [28] and *GASPD* [5], which serve as federated and centralized baselines, respectively.

All experiments were conducted on a machine equipped with a 13th Gen Intel® Core™ i7-1355U processor running at 1.70 GHz and 16 GB of RAM. We implemented the *FedGASPD* and *OptFedGASPD* algorithms and made the source code publicly

Table 1: Characteristics of the event logs used in experiments.

Event log	#Actions	#Traces	#Distinct traces	Max. trace length	Avg. trace length
BPI-2012	24	13,087	4,366	175	20.03
BPI-2013	4	7,554	1,511	123	8.67
BPI-2017	26	31,509	15,930	180	38.15
BPI-2018	34	2,861	2,498	680	61.58
BPI-2019	42	251,734	11,973	990	6.33
Request for Payment	34	6,449	753	27	11.18
Prepaid Travel Cost	19	6,886	89	20	5.34
Sepsis Cases	16	1,050	846	185	14.48
Hospital Billing	18	10,000	1,020	217	4.51

available.[1] For evaluation, we used nine real-world event logs obtained from the IEEE Task Force on Process Mining repository.[2] These logs cover a broad range of process behaviors. Table 1 summarizes their main characteristics.

Table 2 lists the parameters used in the evaluation. The parameters p and m are used by *GASPD* to discover local process models at clients, denoting the initial population size and the number of generations, respectively, to use in the genetic optimization of *GASPD*. The parameter *mms* limits the size of the model extracted at each client; specifically, models with more than 200 nodes and arcs are discarded. In addition, models with more than 200 nodes and arcs produced by *FedGASPD* after merging local models are rejected. The parameter $\tau = 0.2$ specifies the maximum acceptable degradation in entropic relevance (ER) for each client in the optimization problem. Finally, n denotes the number of computational nodes, or clients, used to simulate cross-organizational environments. Each event log is randomly split across up to eight nodes, ensuring that each node receives approximately the same share of the log traces.

Table 2: Experiment settings.

Parameter	Description	Value(s)
p	Population size	50
m	Number of generations	50
mms	Maximum accepted model size	200
τ	Maximum accepted ER degradation	0.2
n	Number of clients	2, 4, 8

Figure 8 shows the characteristics of the Pareto-optimal models discovered from the event logs by *OptFedGASPD* and *FedGASPD* for different numbers of computational nodes (clients), while Table 3 compares the overall quality of the Pareto-optimal models. The numeric entries in Table 3 report by how many more models the number of *OptFedGASPD* models included in the Pareto front of all models, both discovered by *OptFedGASPD* and *FedGASPD* models, exceeds the number of *FedGASPD* models on this global Pareto front. For example, the value +11 for the BPI-2012 event log and 2 nodes indicates that, after jointly comparing all models and removing those dominated

[1] https://github.com/HzhianUnimelb/BPM2026Code

[2] https://www.tf-pm.org/resources/logs

Table 3: Difference in the number of models contributed to the global Pareto front.

Event log	*FedGASPD* vs *OptFedGASPD*		
	2 nodes	4 nodes	8 nodes
BPI-2012	+11	+7	+8
BPI-2013	+1	+1	+2
BPI-2017	+8	+12	+5
BPI-2018	+3	+8	+5
BPI-2019	+5	+4	+2
Request for Payment	-3	+4	+7
Prepaid Travel Cost	+7	+10	+9
Sepsis Cases	+4	+5	+9
Hospital Billing	+5	+3	+4

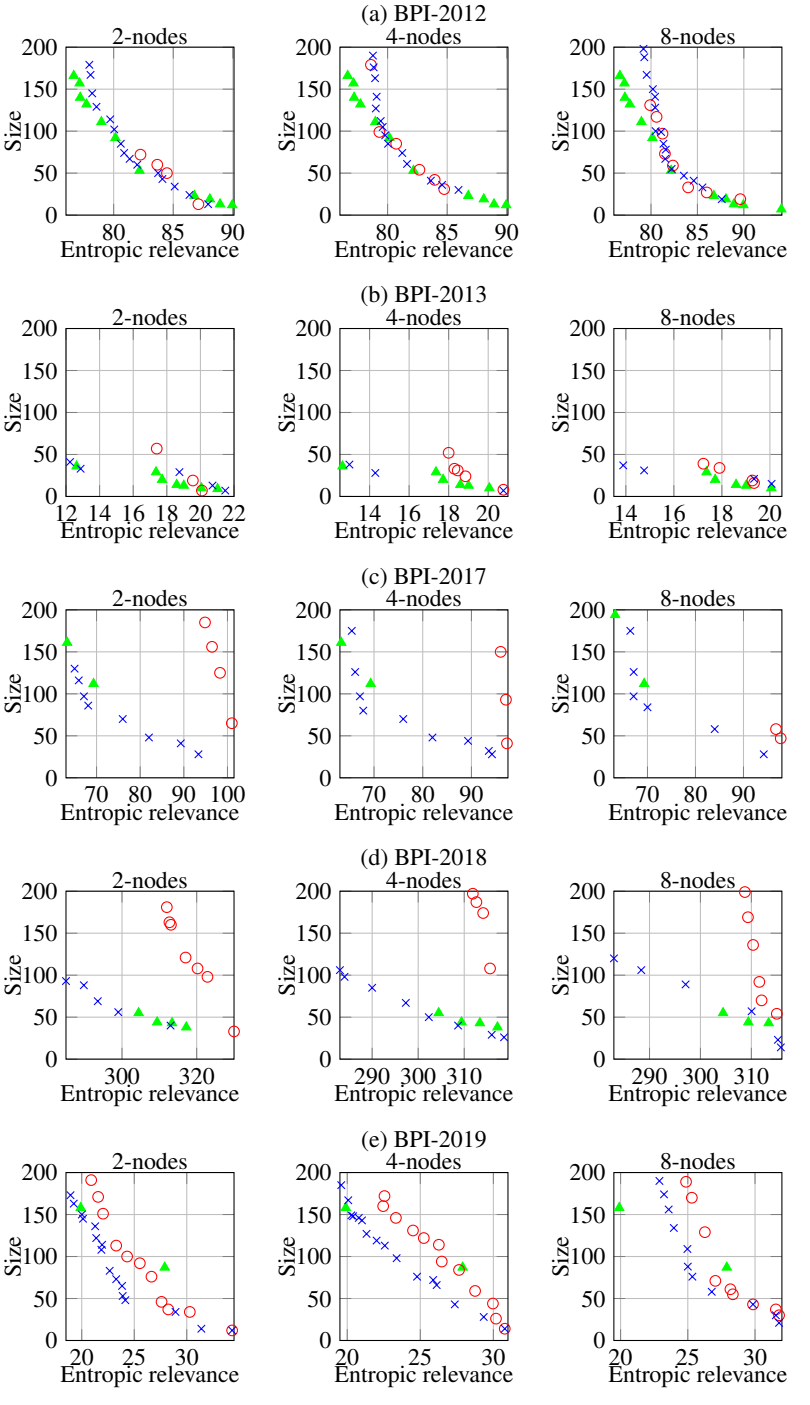
by others, *OptFedGASPD* contributes 11 more models to the global Pareto front than *FedGASPD*. Overall, Table 3 shows that, for most event logs used in our evaluation, a larger proportion of models discovered by *OptFedGASPD* is in the global Pareto front than models discovered by *FedGASPD*. The only exception is the Request for Payment log in the 2-node scenario.

The results indicate that *OptFedGASPD* discovers more accurate and more compact models than *FedGASPD* more often. This improvement stems from the collaborative pruning process carried out by clients under server coordination. Specifically, the optimization removes infrequent local behavior that would otherwise add unnecessary complexity to the global model discovered. At the same time, behavior that is globally important is naturally emphasized by its recurring presence in local models contributed by multiple clients, ensuring its preservation in the optimized global model. These findings support a positive answer to RQ1: *yes*, in *OptFedGASPD*, clients collaboratively optimize the quality and size of the global process model constructed by FedGASPD.

Table 4: Execution times of *GASPD*, *FedGASPD*, and *OptFedGASPD* algorithms.

Event log	Execution time (in seconds)						
	GASPD	*FedGASPD*			*OptFedGASPD*		
		2 nodes	4 nodes	8 nodes	2 nodes	4 nodes	8 nodes
BPI-2012	417.0	178.2	83.5	61.0	179.5	84.2	61.6
BPI-2013	84.1	35.4	26.2	11.1	36.7	27.2	11.5
BPI-2017	3,122.8	1,459.9	514.8	342.4	1,451.4	516.2	343.2
BPI-2018	2,891.0	1,325.3	165.4	76.3	1,326.8	166.3	76.9
BPI-2019	2,279.5	2,216.2	471.2	251.0	2,217.3	472.5	251.8
Request for Payment	1,402.2	685.9	332.6	229.4	686.0	342.2	236.3
Prepaid Travel Cost	172.4	94.0	57.2	46.2	95.7	58.6	46.9
Sepsis Cases	422.7	233.7	64.0	31.4	235.8	65.4	32.0
Hospital Billing	63.0	32.5	21.1	16.2	33.6	21.3	16.4

Table 4 summarizes runtimes of *GASPD*, *FedGASPD*, and *OptFedGASPD*. Overall, the results show that the federated approaches consistently achieve substantial runtime reductions compared to the centralized *GASPD*. In addition, because the model


(a) BPI-2012
2-nodes
4-nodes
8-nodes
(b) BPI-2013
2-nodes
4-nodes
8-nodes
(c) BPI-2017
2-nodes
4-nodes
8-nodes
(d) BPI-2018
2-nodes
4-nodes
8-nodes
(e) BPI-2019
2-nodes
4-nodes
8-nodes
Size
Entropic relevance

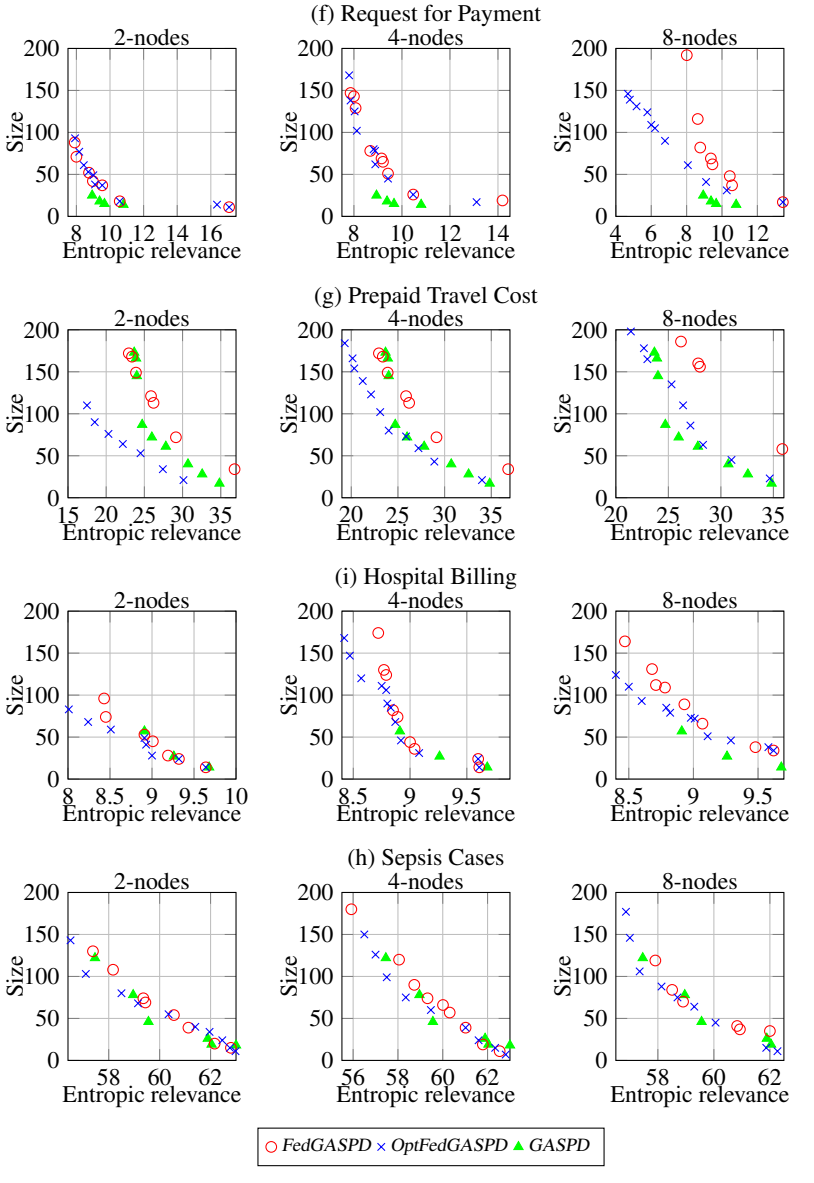


Fig. 8: Size and entropic relevance of DFFAs discovered by *GASPD*, *FedGASPD* and *OptFedGASPD*; smaller sizes and entropic relevance values signify better models.

size is capped at 200, the computational cost of the pruning phase in *OptFedGASPD* remains much lower than that of constructing the aggregated model itself. As a result, the extra runtime introduced by *OptFedGASPD* relative to *FedGASPD* is negligible. For instance, the maximum pruning time is approximately 2.8 seconds for the BPI-2017 event log in the 2-node scenario. These results suggest a positive answer to RQ2: *yes*,

compared with centralized *GASPD*, *OptFedGASPD* is often faster, while its runtime remains comparable to that of *FedGASPD*.

6 Conclusions

This paper introduces *OptFedGASPD*, an approach to federated stochastic process discovery that addresses the limitations of existing methods in cross-organizational environments. By formulating the optimization of the resulting process model as a decision-tree pruning problem, *OptFedGASPD* systematically reduces the unnecessary complexity of the constructed model while preserving its quality for each participating organization. The proposed approach enables organizations to collaboratively optimize the constructed model through orchestration by a central server, ensuring privacy preservation and minimizing communication overhead. Experimental evaluations on real-world industrial event logs demonstrate that *OptFedGASPD* often produces smaller, more accurate models than the approach without pruning, namely *FedGASPD*. The pruning effectively eliminates unvisited states introduced during the aggregation process. These results highlight the potential of *OptFedGASPD* to improve the practicality of federated process discovery in cross-organizational settings. While the current work focuses on pruning states in subtrees disconnected from the main loops of the aggregated model, future research will aim to extend the pruning strategy to encompass all elements that contribute to the aggregated model.

References

[1] van der Aalst, W.: Decomposing Petri nets for process mining: A generic approach. Distributed Parallel Databases **31**(4), 471–507 (2013)
[2] van der Aalst, W.: Process Mining—Data Science in Action. Springer, 2nd edn. (2016)
[3] van der Aalst, W.: A practitioner's guide to process mining: Limitations of the directly-follows graph. Procedia Computer Science **164**, 321–328 (2019)
[4] van der Aalst, W.: Federated process mining: Exploiting event data across organizational boundaries. In: SMDS, pp. 1–7, IEEE (2021)
[5] Alkhammash, H., Polyvyanyy, A., Moffat, A.: Stochastic directly-follows process discovery using grammatical inference. In: CAiSE, LNCS, vol. 14663, pp. 87–103, Springer (2024)
[6] Alkhammash, H., Polyvyanyy, A., Moffat, A., García-Bañuelos, L.: Entropic relevance: A mechanism for measuring stochastic process models discovered from event data. Information Systems **107**, 101922 (2022)
[7] Andersen, J., Rathje, P., Imenkamp, C., Koschmider, A., Landsiedel, O.: EdgeMiner: Distributed process mining at the data sources. In: SAC, pp. 705–713, ACM (2025)
[8] Augusto, A., Dumas, M., La Rosa, M.: Metaheuristic optimization for automated business process discovery. In: BPM, LNCS, vol. 11675, pp. 268–285, Springer (2019)
[9] Augusto, A., Dumas, M., La Rosa, M., Leemans, S.J.J., vanden Broucke, S.K.L.M.: Optimization framework for DFG-based automated process discovery approaches. Software and Systems Modeling **20**(4), 1245–1270 (2021)
[10] Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.: Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity. International Journal of Cooperative Information Systems **23**(01), 1440001 (2014)

[11] Carmona, J., Cortadella, J., Kishinevsky, M.: Divide-and-conquer strategies for process mining. In: BPM, LNCS, vol. 5701, pp. 327–343, Springer (2009)
[12] Carrasco, R.C., Oncina, J.: Learning stochastic regular grammars by means of a state merging method. In: Grammatical Inference and Applications, LNCS, vol. 862, pp. 139–152, Springer (1994)
[13] Gatta, R., Vallati, M., Lenkowicz, J., Masciocchi, C., Cellini, F., Boldrini, L., Fernandez Llatas, C., Valentini, V., Damiani, A.: On the feasibility of distributed process mining in healthcare. In: ICCS, LNCS, vol. 11540, pp. 445–452, Springer (2019)
[14] Harviainen, J., Sommer, F., Sorge, M., Szeider, S.: Optimal decision tree pruning revisited: Algorithms and complexity. In: ICML, Proceedings of Machine Learning Research, vol. 267, PMLR / OpenReview.net (2025)
[15] Hyafil, L., Rivest, R.L.: Constructing optimal binary decision trees is NP-complete. Information Processing Letters **5**(1), 15–17 (1976)
[16] Khan, A., Ghose, A., Dam, H.: Cross-silo process mining with federated learning. In: ICSOC, LNCS, vol. 13121, pp. 612–626, Springer (2021)
[17] Muñoz-Gama, J., Carmona, J.: A fresh look at precision in process conformance. In: BPM, LNCS, vol. 6336, pp. 211–226, Springer (2010)
[18] Nucciarelli, L., Gatta, R., Tudor, A.M., Tavazzi, E., Arcuri, G., Vallati, M., Ibanez-Sanchez, G., Valero-Ramon, Z., Llatas, C.F., Damiani, A.: Towards distributed process discovery in healthcare: Testing and proving the feasibility of the federated alpha+ algorithm. In: Artificial Intelligence in Medicine, LNCS, vol. 15735, pp. 294–299, Springer (2025)
[19] Polyvyanyy, A.: Notes of a process scientist: Process discovery. In: Mining a Scientist's Process, pp. 368–383, Springer (2026)
[20] Polyvyanyy, A., Moffat, A., Garcia-Banuelos, L.: An entropic relevance measure for stochastic conformance checking in process mining. In: ICPM, pp. 97–104, IEEE (2020)
[21] Rafiei, M., van der Aalst, W.: An abstraction-based approach for privacy-aware federated process mining. IEEE Access **11**, 33697–33714 (2023)
[22] Rafiei, M., Pourbafrani, M., van der Aalst, W.: Federated conformance checking. Information Systems **131**, 102525 (2025)
[23] Rennert, C., Albers, J., Leemans, S.J.J., van der Aalst, W.: Your secret is safe with me: Federated directly-follows graph discovery. In: ICPM, pp. 1–8, IEEE (2025)
[24] Rojo, J., Garcia-Alonso, J., Berrocal, J., Hernández, J., Murillo, J.M., Canal, C.: SOWCompact: A federated process mining method for social workflows. Information Sciences **595**, 18–37 (2022)
[25] Yan, J., Liu, C., Zeng, Q., Cao, J., Wu, Y., Ouyang, C., Cheng, L.: Enhancing process discovery by optimizing imprecise sub-processes. IEEE Transactions on Services Computing **19**(1), 337–350 (2026)
[26] Yan, Z., Sun, B., Chen, Y., Wen, L., Hu, L., Wang, J., Yang, M., Wang, L.: Decomposed and parallel process discovery: A framework and application. Future Generation Computer Systems **98**, 392–405 (2019)
[27] Zhian, H.: Federated stochastic process discovery: Definition, benefits, and challenges. In: BPM Doctoral Consortium, CEUR Workshop Proceedings, vol. 4032, pp. 40–45, CEUR-WS.org (2025)
[28] Zhian, H., Buyya, R., Polyvyanyy, A.: Federated stochastic process discovery using grammatical inference. In: CAiSE, LNCS, vol. 15702, pp. 76–93, Springer (2025)
[29] Zhian, H., Buyya, R., Polyvyanyy, A.: Multi-objective metaheuristics for effective and efficient stochastic process discovery. In: BPM, LNCS, vol. 16044, pp. 469–486, Springer (2025)