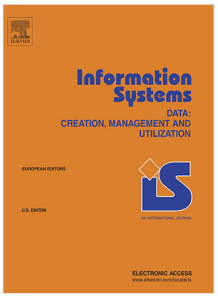
Journal Pre-proof



Sharing pearls, not pebbles: Privacy-aware federated stochastic process discovery with selective model sharing

Hootan Zhian, Rajkumar Buyya, Artem Polyvyanyy

PII: S0306-4379(26)00147-X
DOI: https://doi.org/10.1016/j.is.2026.102832
Reference: IS 102832

To appear in: *Information Systems*

Received date : 1 December 2025
Revised date : 9 November 2026
Accepted date : 16 November 2026

Please cite this article as: H. Zhian, R. Buyya and A. Polyvyanyy, Sharing pearls, not pebbles: Privacy-aware federated stochastic process discovery with selective model sharing, *Information Systems* (2026), doi: https://doi.org/10.1016/j.is.2026.102832.

This is a PDF of an article that has undergone enhancements after acceptance, such as the addition of a cover page and metadata, and formatting for readability. This version will undergo additional copyediting, typesetting and review before it is published in its final form. As such, this version is no longer the Accepted Manuscript, but it is not yet the definitive Version of Record; we are providing this early version to give early visibility of the article. Please note that Elsevier's sharing policy for the Published Journal Article applies to this version, see: https://www.elsevier.com/about/policies-and-standards/sharing#4-published-journal-article. Please also note that, during the production process, errors may be discovered which could affect the content, and all legal disclaimers that apply to the journal pertain.

© 2026 Published by Elsevier Ltd.

Journal Pre-proof

Sharing Pearls, Not Pebbles: Privacy-Aware Federated Stochastic Process Discovery with Selective Model Sharing

Hootan Zhian
The University of Melbourne, Australia
hzhian@student.unimelb.edu.au

Rajkumar Buyya
The University of Melbourne, Australia
rbuyya@unimelb.edu.au

Artem Polyvyanyy
The University of Melbourne, Australia
artem.polyvyanyy@unimelb.edu.au

Tuesday 10th November, 2026

Abstract

Process discovery constructs process models from event logs to describe business processes executed by information systems. Traditional process discovery methods use centrally stored event logs. This approach raises privacy and communication concerns when multiple organizations execute the same process and seek a global model representing their collective behavior. This paper addresses these concerns through federated stochastic process discovery, where organizations retain their event logs locally and share discovered models that capture process behavior and its frequencies with a central server for aggregation. We present *FedGASPD*, which merges complete local models into a single global model, and *FedCGASPD*, which clusters local models before merging to produce multiple global models. To further limit information disclosure, we introduce *P*^{2}*FedGASPD*, which enables organizations to share partial local models while maintaining comparable global model quality. We formulate partial model extraction as a maximum-weight connected subgraph problem, allowing organizations to selectively disclose frequent, representative behavior within a size budget. The amount of disclosed information is adjusted iteratively based on global model quality. We also introduce an evaluation framework based on trace reconstruction attacks and propose Trace Reconstruction Complexity (TRC), a precision-based measure of reconstruction ambiguity under the specified attack model. Experiments on real-world event logs show that complete-model federated discovery achieves model quality comparable to centralized discovery, while partial-model sharing can maintain comparable quality with smaller disclosed models and favorable trade-offs between model size and reconstruction ambiguity.

Keywords: Stochastic process mining, federated process discovery, privacy, model sharing, optimization, cross-silo process discovery, grammatical inference

Journal Pre-proof

Journal Pre-proof

1 Introduction

Process mining uses event logs recorded by information systems during the execution of business processes to gain insights into past behavior and improve future processes [44]. Process discovery, a core subarea of process mining, focuses on automatically constructing process models from event logs that describe the behavior of the underlying system [27, 47]. Discovered models are commonly evaluated in terms of recall (fitness), precision, generalization, and simplicity [4]. Recall measures how well a model reproduces observed behavior, while precision measures how well it avoids allowing unobserved behavior [30]. Generalization concerns plausible behavior not observed in the log [31], while simplicity favors models without unnecessary constructs.

Discovered models can be represented as Directly-Follows Graphs (DFGs) [3, 45]. A DFG is a directed graph whose nodes represent activities, edges indicate which activities directly follow one another in observed executions, and numeric weights indicate the observed frequencies of activities and directly-follows relations.

In cross-organizational process discovery, multiple organizations execute the same process and seek a global process model that represents their collective behavior. In centralized process discovery, participating organizations transfer their event logs to a server, where the logs are combined into a single global event log from which a global process model is discovered. This approach poses significant challenges because organizations are often reluctant to share raw event data [17, 46, 51, 54]. Federated process discovery (FPD) has been proposed to address limitations of centralized discovery, including concerns about data privacy and security [17, 35], data transfer costs [17, 39], and scalability and efficiency issues [6, 39, 43]. However, keeping event logs local does not eliminate disclosure risks: models derived from these logs may reveal rare or distinctive execution paths and their frequencies, potentially enabling inferences about individual process executions. The challenge is therefore not only to avoid sharing event logs, but also to limit the information disclosed through local models while retaining sufficient information to discover useful global models.

Privacy concerns how individuals and organizations control the disclosure of information about them [49]. In process mining, event logs may contain sensitive information [22], and regulations such as GDPR govern the processing of personal data [13]. An event log is a multiset of traces, each recording a sequence of events associated with a case. Events typically record a *case ID*, the *activity* that triggered the event, and a *timestamp*; they may also include further attributes such as the *resource* that executed the activity. Even when extracted from raw system data, event logs may expose sensitive information (e.g., when a case ID identifies a hospital patient), especially in combination with background knowledge [26].

Van der Aalst et al. [48] and Mannhardt et al. [22] discuss privacy challenges in process mining, while Elkoumy et al. [10] highlight threats including re-identification and reconstruction of execution behavior. Existing approaches include group-based techniques, which make individual records less distinguishable [33, 34, 36]; encryption-based techniques, which restrict access to data [5]; and differential privacy, which limits certain inferences about individual contributions through controlled randomization [14, 15, 36]. In federated discovery based on model sharing, a further question is which parts of the locally discovered models organizations need to disclose.

Journal Pre-proof

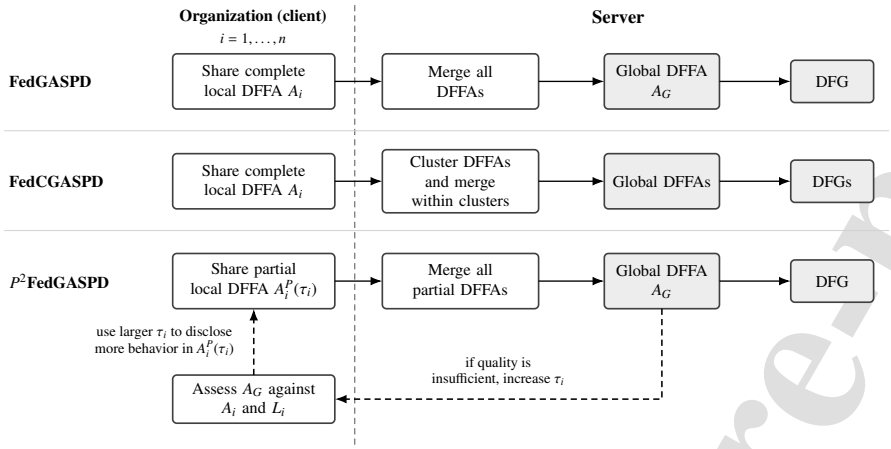


Figure 1: Architectures of *FedGASPD*, *FedCGASPD*, and *P²FedGASPD* for federated stochastic process discovery.

This setting presents three technical challenges. First, stochastic models discovered independently from local event logs must be merged without access to those logs, while retaining meaningful behavioral and frequency information. Second, partial model sharing requires selecting connected, representative behavior under a size budget and rebalancing the retained frequencies to construct consistent models suitable for merging. Third, the potential for reconstructing process executions from disclosed models must be assessed under an explicit attack model; model size alone does not characterize reconstruction ambiguity.

In this paper, we present three approaches to federated stochastic process discovery that let organizations collaboratively discover process models without sharing their event logs. Figure 1 illustrates their architectures. A stochastic process model describes not only which sequences of activities a process may execute, but also their relative likelihoods [18]. In all three approaches, each organization uses the Genetic Algorithm for Stochastic Process Discovery (*GASPD*) [3], a process discovery algorithm based on grammatical inference, to discover a local stochastic process model from its event log. *GASPD* uses a multi-objective genetic algorithm to identify Pareto-optimal parameter configurations with respect to model size and entropic relevance, a measure of how well a stochastic model describes observed behavior and its frequencies. These configurations guide the construction of Deterministic Frequency Finite Automata (DFFAs) using *ALERGIA*, a state-merging algorithm. DFFAs record the frequencies of transitions and trace terminations at individual states, from which probabilities of alternative process executions can be derived. DFFAs serve as intermediate representations that can be translated into DFGs, which summarize stochastic behavior through activity frequencies and direct-follows relations [3]. Each organization selects one local DFFA and, depending on the approach, shares either the complete model or a partial model with a central server for aggregation.

The approaches differ in how they share and aggregate local models. *FedGASPD*

Journal Pre-proof

shares complete local DFFAs and merges them into a single global DFFA. *FedCGASPD* also shares complete local DFFAs but clusters them based on model characteristics before merging models within each cluster to produce multiple global DFFAs. *P²FedGASPD* introduces partial model sharing to limit the disclosure of potentially sensitive process behavior. Each organization selects and shares a connected partial DFFA that prioritizes frequent behavior under a disclosure budget τ, which bounds the combined number of states and transitions in the shared model. The frequencies of the selected partial DFFA are rebalanced to preserve consistency before merging. The server merges the received partial DFFAs into a global DFFA, which is returned to the organizations for quality evaluation. If the global model does not meet an organization's quality requirement, that organization increases τ to share a larger portion of its local model. In all three approaches, the resulting global DFFAs are translated into DFGs. Partial model sharing restricts what is disclosed in a given exchange to improve privacy. We examine reconstruction ambiguity under an attack model in which an attacker accesses a disclosed model without case-specific background knowledge.

The contributions of this article are as follows:

- It develops a DFFA merging procedure based on truncated unfolding and embedding, and establishes the consistency and simulation properties of merged models. Building on this foundation, it introduces *FedGASPD* and *FedCGASPD*, which merge all local models into one global model or cluster them before merging to obtain multiple global models, respectively.
- It introduces *P²FedGASPD* for iterative, quality-guided partial model sharing. It formulates partial DFFA extraction as a generalized maximum-weight connected subgraph problem with frequency-based weights, solves it using mixed-integer linear programming, and rebalances the extracted model's frequencies to preserve consistency before merging.
- It defines a trace reconstruction attack model and introduces Trace Reconstruction Complexity (TRC), a precision-based measure for characterizing ambiguity in reconstructing traces from disclosed models under that attack model, rather than a general privacy guarantee.
- It evaluates the approaches on real-world event logs, comparing complete-model federated discovery with centralized *GASPD*, and partial-model sharing with complete-model sharing in terms of model quality, disclosed model size, and TRC.

This article extends our conference paper [52], which introduced the DFFA merging technique, its formal properties, *FedGASPD*, and *FedCGASPD*, together with their initial evaluation. The additional contributions of this journal article are *P²FedGASPD*, including partial model extraction and frequency rebalancing, the trace reconstruction evaluation framework and TRC, and the extended experimental evaluation of partial model sharing.

The remainder of this paper is organized as follows. Section 2 reviews related work on federated and privacy-aware process mining. Section 3 introduces the concepts and notation used throughout the paper. Section 4 presents the proposed federated stochastic process discovery algorithms, including DFFA merging, complete model sharing through *FedGASPD* and *FedCGASPD*, and selective model sharing through *P²FedGASPD*. Section 5 introduces a framework for assessing trace reconstruction

Journal Pre-proof

ambiguity and evaluates the proposed algorithms through experiments on real-world event logs using our open-source implementation. Finally, Section 6 summarizes the contributions and outlines directions for future research.

2 Related Work

This section discusses related work in federated process mining and privacy-aware strategies for process discovery.

2.1 Federated Process Mining

Federated process mining has emerged as a promising approach. Van der Aalst [46] discusses federated process mining as a means of obtaining unified views of inter-organizational workflows by mapping organizational event logs into a single federated event log. The proposed framework considers two main strategies: sharing filtered event data and sharing higher-level abstractions such as DFGs to limit data exposure. However, it leaves the actual merging of these abstractions as an open challenge. Rennert et al. [38] propose an approach for discovering DFGs from federated event logs without requiring handover of work or a trusted third party. Their approach uses fully homomorphic encryption to compute a global DFG without disclosing the underlying event logs, although the released DFG may still reveal information about the process. Rojo et al. [39] explore federated process mining in distributed environments, such as networks of smartphones capturing human activities. They propose two alternatives: discovering process models locally on each device or merging raw traces centrally to construct process models. While the latter approach involves transferring traces from multiple devices to a central repository, it often incurs substantial data transfer costs. Our approach likewise avoids transferring raw traces, but specifically merges independently discovered stochastic process models. Khan et al. [17] develop a federated approach based on dependency graphs and design a protocol tailored to the Heuristic Miner algorithm that relies on a trusted central server to coordinate communication between parties holding portions of an event log. Our approach can be embedded into similar protocols for federated process discovery. Rafiei and van der Aalst [35] propose an abstraction-based approach for privacy-aware federated process discovery in inter-organizational scenarios, using directly-follows relations to represent event log abstractions. Their approach employs a handover mechanism to capture transitions of cases between organizations, enabling collaboration while limiting event log exposure. It constructs abstractions from trace fragments visible to individual organizations and merges them into a unified representation of the event log. By contrast, our setting assumes that each organization holds complete traces, such as those recorded by a subsidiary or business unit, and discovers a local stochastic model before merging it with models from other organizations.

Decomposition and divide-and-conquer strategies are alternatives to centralized process discovery. Carmona et al. [6] propose two approaches to obtain partial behavioral views from the entire log: one uses region theory to construct and compose local model components, while the other partitions the log into tightly related event

Journal Pre-proof

groups and discovers models for each partition. The resulting models can be assembled into a global process model or presented separately. Similarly, van der Aalst [43] introduces a generic divide-and-conquer framework that partitions activities, projects logs onto these partitions, and composes the resulting models, with strategies such as SESE (single-entry-single-exit) fragments for partitioning. Yan et al. [50] propose a five-step approach for decomposed and parallel process discovery to enable the efficient analysis of large event logs by breaking them into smaller parts. They propose RPSTHD, which combines the Refined Process Structure Tree (RPST) [28], heuristic mining, and Integer Linear Programming (ILP) to support multi-level decomposition. RPSTHD organizes a dependency graph derived from the event log into a hierarchy of Single-Entry-Single-Exit (SESE) fragments, each representing a subset of activities that can be treated as a subgraph. Unlike these decomposition techniques, we discover models from separate collections of complete traces rather than activity-based projections, and merge the resulting models instead of composing components discovered for different activity subsets.

Taken together, these approaches motivate our focus on merging independently discovered stochastic process models, including the clustering of local models and the selective disclosure of model fragments, while assessing the quality of the resulting global models.

2.2 Privacy-Aware Process Mining

Mannhardt et al. [22] analyze the privacy challenges associated with applying process mining to highly personalized data, particularly with the advent of wearable sensor technologies, and propose a framework and preliminary guidelines for privacy-aware process mining. Elkoumy et al. [10] address privacy concerns in process mining, highlighting key threats such as re-identification and reconstruction. Existing techniques differ in whether they protect event data during processing, perturb information before publication, or restrict the release of derived process information.

Burattin et al. [5] introduce a basic method for event log encryption. Encryption can restrict access to event data, but does not by itself prevent inferences from information subsequently released. Michael et al. [24] introduce a system design approach that integrates privacy-preserving mechanisms into process mining for Internet of Things (IoT). It transparently tracks and documents who accesses personal data, as well as when, why, where, and how such data is used. Fahrenkrog-Petersen et al. [15] introduce a prefix-based method in which a noisy activity is added to the prefix of trace variants as long as the perturbed prefix is within an acceptable distance of the original one. This solution, unlike [36], might produce new trace variants that are not in the event log.

Elkoumy et al. [11] introduce a differential privacy approach for event log anonymization. It groups traces using Deterministic Acyclic Finite State Automata (DAFSA) to capture shared prefixes and suffixes. The method estimates attacker knowledge for each event group and filters out entire cases above a risk threshold. Noise is added by sampling cases and perturbing timestamps.

Ueck et al. [42] introduce DP-ELCA, a framework for anonymizing event logs with case attributes. It combines differentially private synthetic data generation for control-flow and case attributes with TraVaS [36], which filters infrequent trace variants.

Journal Pre-proof

PRIPEL [14] applies differential privacy at the level of individual cases, rather than the entire event log. It perturbs trace-variant frequencies, potentially removing observed variants or introducing new ones. In the next step, each anonymized sequence is paired with attribute values and timestamps from the most similar trace in the original log, or filled with random values if no match exists. The timestamps are perturbed slightly to limit linkage to original event times.

Kabierski et al. [16] propose a framework for computing process performance indicators (PPIs) from business process event logs using differential privacy to limit disclosure of individual information.

Rafiei and van der Aalst [32] introduce a method that perturbs activity frequencies to mitigate frequency-based disclosure risks. Elkoumy et al. [9] introduce a differentially private DFG construction method that adds noise to edge frequencies.

Lim et al. [19] propose a pipeline for privacy-aware process discovery by first partitioning the event log and then applying differential privacy. The partitioning is achieved through event abstraction, where low-level activities in the event log are grouped into higher-level activities (sub-processes) using a mapping function, resulting in multiple sub-logs that each represent different parts of the process. After partitioning, each sub-log is anonymized separately using two differential privacy mechanisms, SaCoFa [15] and the Laplace mechanism [23].

Rafiei and van der Aalst [33] propose a TLKC-privacy approach that suppresses selected infrequent events to limit attribute linkage under specified privacy requirements. However, this suppression can introduce new behavior in the event log that did not exist in the original event log. Rafiei and van der Aalst [34] extend the TLKC-privacy approach to support the main process mining perspectives (control-flow, organizational, case, and time), and introduce an improved event filtering score to cover different perspectives. Such group-based protections depend on the background knowledge and attributes considered, rather than preventing every possible inference.

Rafiei et al. [36] propose TraVaS, which uses differentially private partition selection to determine which trace variants to publish. It perturbs the frequency of each trace variant using the k-Truncated Symmetric Geometric Distribution (k-TSGD) mechanism and publishes only those variants whose noisy frequencies exceed a threshold, removing infrequent traces.

In related work on federated conformance checking, Rafiei et al. [37] propose an approach in which organizations share interaction events while withholding internal activities.

Most of these approaches protect or perturb event data or released statistics. Our work instead focuses on controlling which parts of independently discovered stochastic process models are shared. Partial model disclosure does not guarantee full privacy. Instead, we assess trace reconstruction ambiguity under an explicit, restricted attack model and examine its trade-off with the quality of the global models.

3 Preliminaries

This section reviews the concepts underlying our federated stochastic process discovery approaches. Section 3.1 introduces Deterministic Frequency Finite Automata (DF-

Journal Pre-proof

FAs), together with the notions of frequency consistency and simulation used to characterize model merging. Section 3.2 reviews process model precision, including its eigenvalue-based formulation, which forms the basis of our measure of trace reconstruction ambiguity. Finally, Section 3.3 summarizes *GASPD*, the stochastic process discovery algorithm used to construct local process models.

3.1 Automata

A Deterministic Frequency Finite Automaton (DFFA) is an automaton that associates frequencies with states and transitions.

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*;
- $\mathbb{I} : Q \to \mathbb{R}_{\geq 0}$ is the *initial state frequency function*, where $q_0 \in Q$ is the unique initial state, $\mathbb{I}(q_0) > 0$, and $\mathbb{I}(q) = 0$ for all $q \in Q \setminus \{q_0\}$.
- $\mathbb{F} : Q \to \mathbb{R}_{\geq 0}$ is the *final state frequency function*; and
- $\delta : Q \times \Lambda \times Q \to \mathbb{R}_{\geq 0}$ is the *transition frequency function*, such that $\forall q \in Q \forall a \in \Lambda \forall q' \in Q \forall q'' \in Q : ((\delta(q, a, q') > 0 \wedge \delta(q, a, q'') > 0) \Rightarrow (q' = q''))$. ⌟

A transition from state q to state q' labeled with a exists if and only if $\delta(q, a, q') > 0$. The value $\delta(q, a, q')$ specifies its frequency.

Figure 2 shows a DFFA with states $\{p_0, \ldots, p_4\}$, initial state p_0, $\mathbb{I}(p_0) = 100$, $\mathbb{F} = \{(p_0, 0), (p_1, 25), (p_2, 70), (p_3, 0), (p_4, 5)\}$, and $\delta(p_0, a, p_1) = 50$, $\delta(p_0, b, p_2) = 50$, $\delta(p_1, a, p_2) = 25$, $\delta(p_1, b, p_3) = 10$, $\delta(p_2, a, p_1) = 10$, $\delta(p_3, b, p_2) = 5$, and $\delta(p_3, a, p_4) = 5$.

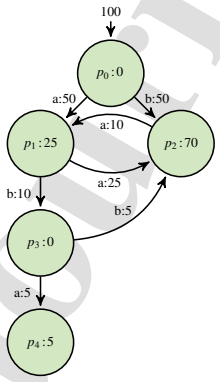


Figure 2: DFFA A_1.

A DFFA is consistent if, for every state, the sum of its initial frequency and incoming transition frequencies equals the sum of its final frequency and outgoing transition frequencies.

Journal Pre-proof

Definition 3.2 (Consistency, de la Higuera [8])
A DFFA $(Q, \Lambda, q_0, \mathbb{I}, \mathbb{F}, \delta)$ is *consistent* if and only if it holds that:

$$\forall q \in Q : \left(\mathbb{I}(q) + \sum_{q' \in Q, a \in \Lambda} \delta(q', a, q) \right) = \left(\mathbb{F}(q) + \sum_{q' \in Q, a \in \Lambda} \delta(q, a, q') \right).$$

⌟

The DFFA in Figure 2 is consistent.

The *simulation* relation associates automata that behave in a similar way, that is, one automaton can "mimic" the actions of the other.

Definition 3.3 (Simulation [40])
DFFA $A_1 = \left(Q^1, \Lambda^1, q_0^1, \mathbb{I}^1, \mathbb{F}^1, \delta^1\right)$ *simulates* DFFA $A_2 = \left(Q^2, \Lambda^2, q_0^2, \mathbb{I}^2, \mathbb{F}^2, \delta^2\right)$, denoted by $A_2 \preceq A_1$, if and only if there exists a relation $\mathbb{S} \subseteq Q^1 \times Q^2$ such that:

- $\left(q_0^1, q_0^2\right) \in \mathbb{S}$, and
- for every $(p, q) \in \mathbb{S}$, every $a \in \Lambda^2$, and every $q' \in Q^2$, if $\delta^2(q, a, q') > 0$, then $a \in \Lambda^1$ and there exists $p' \in Q^1$ such that $\delta^1(p, a, p') > 0$ and $(p', q') \in \mathbb{S}$. ⌟

3.2 Precision

Precision measures how well a process model avoids allowing behavior that was not observed in an event log. Let L be an event log and lang(M) denote the language of a process model M, i.e., the set of traces that M can generate. A model has higher precision when it allows less behavior beyond the traces observed in L.

We use eigenvalue-based precision [30], which implements a monotone language-quotient framework for comparing process behavior. It supports models with infinite languages by using the short-circuit eigenvalue measure:

$$\text{precision}(L, M) = \frac{\text{eig}^\bullet(\text{lang}(M) \cap \text{supp}(L))}{\text{eig}^\bullet(\text{lang}(M))},$$

where supp(L) denotes the set of distinct traces in L, and $\text{eig}^\bullet(\,.\,)$ is the short-circuit eigenvalue measure. Intuitively, the short-circuit eigenvalue measures the richness of a language's behavior through the growth of possible execution sequences. For a fixed set of observed traces allowed by the model, permitting additional unobserved behavior increases the denominator without changing the numerator, resulting in lower precision. Eigenvalue-based precision satisfies all precision propositions evaluated by Syring et al. [41].

In this paper, we use eigenvalue-based precision to assess trace reconstruction ambiguity from disclosed process models. Specifically, we use its complement to define Trace Reconstruction Complexity (TRC), which characterizes the ambiguity of reconstructing individual traces from automata under the attack model introduced in Section 5.2.

Journal Pre-proof

3.3 Genetic Algorithm for Stochastic Process Discovery

Our approach to federated process discovery is based on *GASPD*, an evolutionary stochastic process discovery algorithm grounded in grammatical inference [3]. Multi-objective optimization has also been investigated for stochastic process discovery to improve the trade-off between model quality and complexity [53]. *GASPD* uses *ALERGIA* to construct stochastic process models from an input event log. *ALERGIA* first constructs a prefix automaton from the event log and then generalizes it by recursively merging states with statistically compatible trace termination and outgoing-transition frequencies into a consistent DFFA [7].

Algorithm 1 summarizes *GASPD* in pseudocode.

Algorithm 1: *GASPD*, adapted from [3]

Data: Initial population size p, number of generations m, number of parents k to generate offspring, and event log L
Result: Pareto-optimal parameter triples

```
1 g ← 0;
2 P ← POPULATION(p);
3 while g < m do
4     F ← SELECT(P, L);
5     U ← CROSSOVER-MUTATION(F, k);
6     P ← REPLACE-ELITE(U, F, L);
7     g ← g + 1;
8 F ← SELECT(P, L);
9 return F;
```

GASPD uses a multi-objective genetic search to identify parameter configurations for discovering stochastic process models from an input event log. In Line 1, the current generation counter g is initialized to zero. In the second initialization step (Line 2), we generate an initial population P of p parameter triples. Each parameter triple consists of one parameter that determines which infrequent traces are excluded from the event log and two parameters that guide *ALERGIA* in constructing a DFFA from the remaining frequent traces. These parameter triples determine the candidate models evaluated during the selection phase.

In the selection phase (Line 4), *ALERGIA* is applied using the parameter triples, and the Pareto-optimal population members F are selected. Candidates are evaluated using two objectives: model size (smaller models are preferred) and entropic relevance (lower values are preferred). Entropic relevance serves as a quality criterion because it enables rapid model scoring, balances precision and recall relative to the event log, and quantifies how well the model captures the likelihood of observed traces [2, 3].

Next, in Line 5, offspring U are generated by applying crossover and mutation operations to up to k randomly selected parents from F. These operations explore new parameter configurations by combining and modifying existing ones. In the replacement phase (Line 6), *GASPD* updates the population by preserving Pareto-optimal members in F and incorporating new candidates from U. In Line 7, the generation counter g is incremented by one, and the while loop (Line 3) continues until g reaches the maximum number of generations, m. After the final iteration of the while loop, a final selec-

Journal Pre-proof

tion step is performed by applying *SELECT* to the last population P. This updates the Pareto-optimal set F by incorporating any newly discovered non-dominated parameter triples generated in the final generation (Line 8). Finally, in Line 9, the Pareto-optimal (non-dominated) parameter triples F are returned. Each returned parameter triple can be used to construct a DFFA using *ALERGIA*. Our federated process discovery approaches select one such triple for each local event log and merge the corresponding DFFAs. The model conversion procedure employed by *GASPD* is then used to obtain global DFGs from the merged DFFAs.

4 Federated Stochastic Process Discovery

This section presents our approaches to federated stochastic process discovery. We first define the federated process discovery problem (Section 4.1) and introduce a technique for merging DFFAs based on unfolding and embedding (Section 4.2). We then present *FedGASPD*, which discovers one DFFA from each input event log and merges them into a single global model (Section 4.3), and *FedCGASPD*, which clusters similar local models before merging them to obtain multiple global models (Section 4.4). Finally, we introduce *P*^{2}*FedGASPD*, which extends federated discovery with selective model sharing by extracting and rebalancing partial DFFAs and iteratively adjusting the amount of shared information based on global model quality (Section 4.5).

4.1 Definition

Let X be a set. By $\mathcal{B}(X)$, we denote the set of all finite multisets over X. By X^* and X^+, we denote the set of all finite sequences and the set of all finite non-empty sequences over X, respectively. Let $\mathbb{X} \in X^*$ be a sequence over X. Then, by $\mathbb{X}(i)$, $i \in [1 \,..\, |\mathbb{X}|]$, we denote the element at position i in $\mathbb{X}$.

Let $\mathcal{A}$ be the universe of *actions*. Then, $\mathcal{L} = \mathcal{B}(\mathcal{A}^*)$ is the universe of *event logs*. Let $\mathcal{M}$ be the universe of *process models*. Given an event log $L \in \mathcal{L}$, the conventional *process discovery* problem studies ways to construct a process model that describes the system that generated L. A solution to the process discovery problem can be captured as a function $d : \mathcal{L} \to \mathcal{M}$. The *federated process discovery* problem studies ways to construct a process model from a list of event logs, where each event log from the list can be stored on a dedicated device at a particular organization. That is, a solution to the federated process discovery problem can be given as a function $f : \mathcal{L}^+ \to \mathcal{M}$. This formulation considers federated discovery that produces a single model. We also study a variant that produces multiple models, presented in Section 4.4.

Let $q : \mathcal{L} \times \mathcal{M} \to [0, 1]$ be a normalized model quality measure, where larger values indicate better quality. For instance, measure q can be precision, recall, simplicity, or generalization [4]. A desirable goal of a federated process discovery approach f is to construct a model from multiple event logs that is comparable in quality to the model constructed from the corresponding centralized event log:

$$\forall\, \mathbb{L} \in \mathcal{L}^+ : q(L, f(\mathbb{L})) \approx q(L, d(L)), \text{ where } L = \uplus_{i=1}^{|\mathbb{L}|} \mathbb{L}(i)\,.$$

Journal Pre-proof

In this paper, we solve the federated process discovery problem by first constructing process models from individual event logs using the conventional approach and then merging these models into the final process model. Let $m : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$ be a function that merges models; in Section 4.2, we instantiate m with a parameterized procedure for merging DFFAs. Specifically, given a non-empty, finite list of event logs $\langle L_1, \ldots, L_n \rangle \in \mathcal{L}^+$, $n \in \mathbb{N}$, we first discover the sequence of models $\langle M_1, \ldots, M_n \rangle$, $M_i = d(L_i)$, $i \in [1 \,..\, n]$, using the conventional approach on each event log. Each model can be constructed on a dedicated device at the organization that owns the data, providing an abstract representation of the corresponding event log without transferring the event log itself. We then merge the models incrementally using function $F_m : \mathcal{M}^+ \to \mathcal{M}$:

$$F_m(\langle M_1, M_2, \ldots, M_n \rangle) = F_m(\langle m(M_1, M_2), \ldots, M_n \rangle) \text{ for } n \geq 2, \text{ with } F_m(\langle M_1 \rangle) = M_1 \,.$$

In general, different orders and approaches to model merging can lead to different resulting models. Next, we discuss an approach to merge DFFAs.

4.2 Model Merging

We refer to the procedure for merging a DFFA A_2 into a DFFA A_1 as *MergeDFFA*. In addition to the two automata, the procedure takes as input a parameter $d \in \mathbb{N}$, $d \geq 1$. The first step is to construct a prefix of the unfolding of A_2, limiting the number of occurrences of each original state along an unfolding path to at most d [12]. In this prefix automaton, state and transition frequencies are derived from those in A_2 by propagating frequencies along the unfolding branches, using probabilities calculated from the original frequencies in A_2. *MergeDFFA* then "embeds" this prefix automaton into A_1 by reusing the structure of A_1 and extending it as necessary to incorporate parts of the prefix not already represented in A_1. During this embedding process, the frequencies from A_1 and the constructed prefix of A_2 are combined. We denote the result of merging A_2 into A_1 using parameter d as $A_1 \triangleleft_d A_2$.

4.2.1 Unfolding

The unfolding procedure constructs a finite prefix of a consistent DFFA. It employs Breadth-First Search (BFS) to traverse the original model, starting from its initial state. Each unfolded state corresponds to a state in the original DFFA. Since the original model may contain cycles, the same original state may be encountered multiple times during unfolding, including along a single path. Each occurrence is represented by a fresh state, ensuring that the unfolded model has a tree structure. The initial state q_0 is retained, with the initial state frequency $\mathbb{I}''(q_0) = \mathbb{I}(q_0)$, while all other unfolded states have initial frequency zero.

To ensure termination, the unfolding is bounded by parameter d. For each unfolded state, every outgoing transition of its corresponding original state is retained if and only if extending the current path with that transition does not cause any original state to occur more than d times along that path. This bound ensures termination of the unfolding while retaining all transitions along paths that satisfy the bound. The bound is applied independently to each unfolding path, rather than to the total number of occurrences of a state in the unfolded model.

Journal Pre-proof

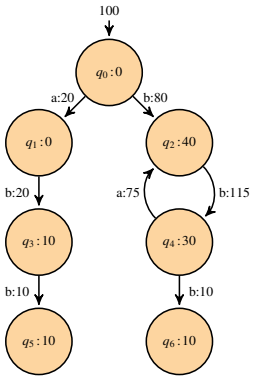


Figure 3: DFFA A_2.

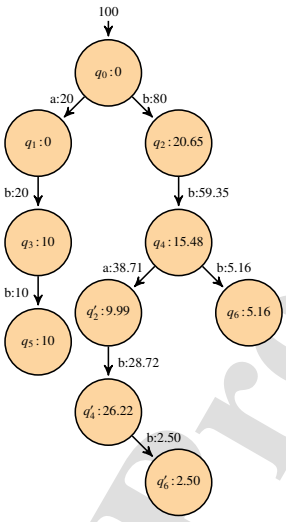


Figure 4: Prefix of the unfolding of DFFA A_2 for $d = 2$.

The frequencies of the unfolded model are calculated by propagating frequencies along its branches. The key idea is to preserve the relative frequencies of retained outgoing transitions at each original state. Since different occurrences of the same state may receive different incoming frequencies, the frequency reaching each unfolded state is distributed among its retained outgoing transitions according to the corresponding proportions in the original DFFA. The remaining frequency is assigned as the final frequency.

Let u be an unfolded state corresponding to an original state q, and let $inFreq(u)$ denote its incoming frequency; for the initial state q_0, it holds that $inFreq(q_0) = \mathbb{I}(q_0)$. Let $outFreq(q)$ denote the sum of the final frequency and all outgoing transition frequencies of the original state q. Then, for each transition (q, a, r) retained during unfolding, a fresh state v corresponding to r is created. The frequency of the unfolded transition is calculated as:

$$\delta^u(u, a, v) = \frac{inFreq(u) \cdot \delta(q, a, r)}{outFreq(q)}.$$

The incoming frequency of v is then set to $\delta^u(u, a, v)$.

To preserve consistency, the final frequency of each unfolded state is calculated by subtracting the frequencies of its retained outgoing transitions from its incoming frequency:

$$\mathbb{F}^u(u) = inFreq(u) - \sum_{a \in \Lambda,\ v \in Q^u} \delta^u(u, a, v).$$

Consequently, the frequencies of transitions excluded by the unfolding bound are incorporated into the final frequency of their source state.

Journal Pre-proof

Consider DFFA A_2 in Figure 3. Figure 4 shows its unfolding for $d = 2$. The initial state q_0 has an initial frequency of 100, which is propagated along transitions labeled a and b with frequencies 20 and 80, respectively. The left branch through q_1, q_3, and q_5 is retained without modification because it contains no cycles.

For the right branch, state q_2 receives an incoming frequency of 80. In the original DFFA, q_2 has a final frequency of 40 and an outgoing transition frequency of 115. Thus, the outgoing transition frequency in the unfolded model is $\delta^u(q_2, b, q_4) = 80 \cdot (115/155) \approx 59.35$. The final frequency is then $\mathbb{F}^u(q_2) = 80 - 59.35 \approx 20.65$. The incoming frequency of q_4 in the unfolded automaton is, therefore, approximately 59.35. Using the original frequencies of q_4, its final frequency becomes 15.48, while its outgoing transitions receive frequencies 38.71 and 5.16. The transition with frequency 38.71 leads to a new occurrence q_2' of the original state q_2.

The same calculation is repeated for q_2', yielding a final frequency of 9.99 and a transition with frequency 28.72 to a new occurrence q_4'. At this point, the original states q_2 and q_4 have each been visited twice along the unfolding path. Therefore, the transition from q_4' back to q_2 is excluded by the bound $d = 2$. However, the transition from q_4' to q_6' is retained, with frequency $\delta^u(q_4', b, q_6') = 28.72 \cdot (10/115) \approx 2.50$. The final frequency of q_4' is consequently: $\mathbb{F}^u(q_4') \approx 28.72 - 2.50 = 26.22$. This final frequency includes the frequency of the transition excluded by the unfolding bound. The resulting prefix is frequency-consistent, as the incoming frequency of each unfolded state equals the sum of its final and outgoing transition frequencies.

4.2.2 Embedding

The embedding procedure combines DFFA A_1 with the prefix obtained by unfolding DFFA A_2, as described in the previous section. The procedure constructs a merged DFFA A_G by preserving the states and transitions of A_1 and extending its structure to represent the behavior of the prefix. When a transition of the prefix can be represented by an existing transition in the merged model, that transition is reused. Otherwise, a new state and transition are introduced. The frequencies of the two models are combined while preserving consistency.

Let A_2^u be the prefix of the unfolding of automaton A_2. Initially, A_G is a copy of A_1, with its alphabet extended to the union of the alphabets of A_1 and A_2. The initial states of A_1 and A_2^u are represented by the initial state of A_G, whose initial frequency is the sum of the initial frequencies of A_1 and A_2. All other states retain zero initial frequency. The states and transitions of A_1 are retained, and their original frequencies are used as the initial values to which contributions from the unfolded prefix are added.

The embedding procedure traverses A_2^u in the BFS order, starting from its initial state. Each unfolded state u is associated with a state $\mu(u)$ in the merged model, where μ denotes the embedding mapping. Initially, the initial state of A_2^u is mapped to the initial state of A_1. For each transition (u, a, v) in A_2^u, let $p = \mu(u)$ be the state representing its source in the merged model. If p already has an outgoing transition labeled a to a state p', this transition is reused, and v is mapped to p'. Otherwise, a fresh state p' and a transition (p, a, p') are added to the merged model, and v is mapped to p'. In either case, the frequency of the unfolded transition is added to the frequency of the corresponding transition in the merged model. Since every non-initial state of the unfolded prefix has

Journal Pre-proof

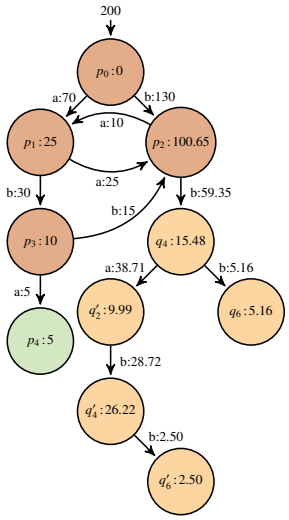


Figure 5: DFFA $A_G = A_1 \triangleleft_2 A_2$.

a unique predecessor, this procedure determines the mapping of every unfolded state. Reusing existing transitions also preserves determinism.

The final frequency of each unfolded state is added to the final frequency of its corresponding state in the merged model. Thus, several unfolded states may contribute to the final frequency of the same merged state. Every state and transition of the prefix contributes its frequency exactly once, while the original frequency contributions of A_1 are counted exactly once.

Consider DFFAs A_1 and A_2 in Figures 2 and 3, respectively. Figure 5 shows the result of embedding the prefix of A_2 from Figure 4, obtained with $d = 2$, into A_1; that is, $A_G = A_1 \triangleleft_2 A_2$. The initial states p_0 and q_0 are represented by p_0 in the merged automaton, whose initial frequency becomes $100 + 100 = 200$. Both automata have transitions labeled a and b from their initial states. These transitions are therefore combined, resulting in frequencies $50 + 20 = 70$ for the transition from p_0 to p_1 and $50 + 80 = 130$ for the transition from p_0 to p_2. The corresponding unfolded states q_1 and q_2 are mapped to p_1 and p_2, respectively.

The same procedure continues along the remaining transitions of the prefix. For example, the transition from q_1 to q_3 labeled b is embedded into the existing transition from p_1 to p_3, increasing its frequency from 10 to 30. Similarly, the transition from q_3 to q_5 labeled b is embedded into the existing transition from p_3 to p_2, increasing its frequency from 5 to 15. Consequently, both q_2 and q_5 are represented by p_2. Their final frequencies are added to the original final frequency of p_2, yielding $\mathbb{F}_G(p_2) = 70 + 20.65 + 10 \approx 100.65$.

In contrast, the transition from q_2 to q_4 labeled b cannot be embedded into an existing transition from p_2, because p_2 has no outgoing transition labeled b in A_1. A new state q_4 and a transition from p_2 to q_4 with frequency 59.35 are thus introduced. The

Journal Pre-proof

remaining transitions of the unfolded prefix are embedded in the same way, introducing further states and transitions where necessary. In Figure 5, the khaki states correspond to states of A_1 reused during embedding, the green state belongs to A_1 but is not visited during embedding, and the orange states are newly introduced from the prefix.

The embedding procedure terminates because the unfolded prefix is finite and every state and transition in the prefix is processed only once. With efficient lookup of outgoing transitions, the embedding time is linear in the combined sizes of A_1 and the unfolded prefix. However, the prefix itself can be exponentially larger than A_2. Increasing the unfolding bound d allows longer cyclic behaviors to be represented, potentially increasing the size of the prefix and the resulting merged automaton.

Lemma 4.1 (Consistency)
If A_1 and A_2 are consistent DFFAs, then DFFA A, such that $A = A_1 \triangleleft_d A_2$, $d \in \mathbb{N}$, $d \geq 1$, is consistent. ⌟

Lemma 4.1 follows from the consistency of A_1 and the unfolded prefix A_2^u. By construction, each state of the unfolded prefix is mapped to a state of the merged model, and each transition is mapped to a transition with the same label between the corresponding states. The frequencies of the merged model are obtained by adding the contributions of A_1 and the unfolded prefix. For every merged state, summing the consistency equations of all states mapped to it, together with the consistency equation of the corresponding state of A_1 when present, preserves the equality between total incoming and total outgoing frequencies, including initial and final frequencies. Hence, the merged DFFA is consistent.

The merged automaton also preserves the behavior of A_1 and, when A_2 is acyclic, the behavior of A_2.

Lemma 4.2 (Simulation)
If A_1 and A_2 are consistent DFFAs and $A = A_1 \triangleleft_d A_2$, $d \in \mathbb{N}$, $d \geq 1$, it holds that:
- *A simulates A_1, that is, $A_1 \preceq A$; and*
- *if A_2 is acyclic, then A simulates A_2, that is, $A_2 \preceq A$.* ⌟

The first claim of Lemma 4.2 follows because embedding preserves every state and transition of A_1, extending its structure without removing existing transitions. For the second claim, observe that when A_2 is acyclic, no state can occur more than once along an unfolding path. Therefore, for any $d \geq 1$, the unfolding bound excludes no transitions, and the resulting prefix simulates A_2. The embedding mapping preserves every transition of this prefix in the merged DFFA. Consequently, the merged DFFA simulates A_2 whenever A_2 is acyclic.

Finally, the merging operation is not commutative. In general, $A_1 \triangleleft_d A_2 \neq A_2 \triangleleft_d A_1$, because embedding reuses the structure of the first DFFA and extends it with behavior from the unfolded prefix of the second. Thus, different orders of merging local DFFAs may produce different merged models.

4.3 Discovering Single Model through Model Merging

FedGASPD is a federated process discovery approach that constructs a global process model from multiple event logs without requiring organizations to share their event

Journal Pre-proof

data. We assume that the event logs originate from the same process but are recorded by different organizations or departments within an organization. *FedGASPD* discovers a local DFFA from each event log using *GASPD* and then merges these DFFAs into a single global model using the *MergeDFFA* procedure described in Section 4.2. Algorithm 2 summarizes the approach.

Algorithm 2: *FedGASPD*

Input: Initial population size p, number of generations m, number of parents k to generate offspring, unfolding bound $d \in \mathbb{N}$, $d \geq 1$, and a non-empty list of event logs $L = \langle L_1, \ldots, L_n \rangle$

Output: A global DFFA A_G

1 **parallelfor** $i \in [1 \,..\, n]$ **do**

2 $\quad M_i \leftarrow \mathit{SELECTDFFA}(\mathit{GASPD}(p, m, k, L_i), L_i)$;

3 $M \leftarrow \langle M_1, \ldots, M_n \rangle$;

4 **return** $F_{\mathit{MergeDFFA}}(M, d)$;

For each event log L_i, *FedGASPD* first applies *GASPD* (Algorithm 1) to identify Pareto-optimal parameter triples with respect to model size and entropic relevance. The *SELECTDFFA* function selects one of these triples according to a model selection policy and constructs the corresponding DFFA from L_i using *ALERGIA*. Since local model discovery is independent for each event log, these computations are performed in parallel (Lines 1 and 2). Different implementations of *SELECTDFFA* can employ different model selection policies.

The selected DFFAs are collected into the ordered sequence $M = \langle M_1, \ldots, M_n \rangle$. The aggregation function $F_{\text{MergeDFFA}}$ merges these models incrementally using the *MergeDFFA* procedure. Specifically, the models are merged from left to right:

$$F_{\text{MergeDFFA}}(\langle M_1 \rangle, d) = M_1, \quad \text{and, for } n \geq 2,$$

$$F_{\text{MergeDFFA}}(\langle M_1, \ldots, M_n \rangle, d) = F_{\text{MergeDFFA}}(\langle M_1, \ldots, M_{n-1} \rangle, d) \triangleleft_d M_n.$$

Thus, the first two models are merged, the resulting model is merged with the third, and the procedure continues until all models have been incorporated. The order is fixed because *MergeDFFA* is not commutative and different merging orders may produce different global models. The final merged DFFA is returned as the result (Line 4).

In the federated setting, each organization performs *GASPD* and *SELECTDFFA* locally using its own event log. Only the selected DFFA is transmitted to the central server, where the models are merged. Consequently, the organizations can collaboratively discover a global process model without transferring their event logs.

After merging, the global DFFA can be converted into a probabilistic automaton using Algorithm 16.1 by de la Higuera [8], and subsequently into a DFG using the procedure described in Definition 4.2 by Alkhammash et al. [3]. Since DFFAs serve as the intermediate representation used by *GASPD*, we describe the proposed federated discovery techniques in terms of DFFAs.

Journal Pre-proof

4.4 Discovering Multiple Models through Model Merging

FedGASPD discovers one model from each input event log and merges them into a single global model. However, when local models exhibit substantially different characteristics, merging them into a single model may obscure these differences by combining distinct behaviors. To address this limitation, we propose a variation of *FedGASPD* that groups local models with similar characteristics into clusters and merges the models within each cluster. This approach produces multiple global models, each representing a group of local models with similar characteristics. We refer to this variant as *FedCGASPD*, which is summarized in Algorithm 3.

Algorithm 3: *FedCGASPD*

Input: Initial population size p, number of generations m, number of parents k to generate offspring, unfolding bound $d \in \mathbb{N}$, $d \geq 1$, and a non-empty list of event logs $L = \langle L_1, \ldots, L_n \rangle$

Output: A non-empty list of global DFFAs

1 **parallelfor** $i \in [1 \,..\, n]$ **do**
2 $\quad M_i \leftarrow \mathit{SELECTDFFA}(\mathit{GASPD}(p, m, k, L_i), L_i)$;
3 $\quad r_i \leftarrow \mathit{Er}(M_i, L_i)$;
4 $M \leftarrow \langle M_1, \ldots, M_n \rangle$;
5 $R \leftarrow \langle r_1, \ldots, r_n \rangle$;
6 $C \leftarrow \mathit{KMeansCluster}(M, R, n)$;
7 $\mathbb{M} \leftarrow \langle\rangle$;
8 **for** $i \in [1 \,..\, n]$ **do**
9 $\quad$ **if** $|C(i)| \geq 1$ **then** $\mathbb{M} \leftarrow \mathbb{M} \circ \langle F_{\mathit{MergeDFFA}}(C(i), d) \rangle$;
10 **return** $\mathbb{M}$;

For each event log L_i, *FedCGASPD* discovers a local DFFA M_i using *GASPD* and *SELECTDFFA*, following the procedure described in Section 4.3. Since local model discovery is independent for each event log, these computations are performed in parallel. Each organization also computes the entropic relevance r_i of its selected DFFA with respect to its local event log. The selected DFFAs and their entropic relevance values are transmitted to the central server without sharing the underlying event logs.

The server clusters the local models using the k-means algorithm [21] via the *KMeansCluster* function (Line 6). This function takes as input the sequence of models M, their entropic relevance values R, and the number of centroids n, and returns a sequence of clusters C, where each cluster contains a subsequence of models from M.

For clustering, each DFFA is represented as a point in a two-dimensional space, defined by its normalized size and entropic relevance. Normalization is achieved by dividing the size and entropic relevance of each model by the respective maximum values across all models in M. We initialize n random centroids (initial cluster centers), each represented as a point in the space $[0, 1] \times [0, 1]$. The k-means algorithm iteratively assigns models to their nearest centroids and updates the centroids based on their assigned models. Some centroids may remain unused, resulting in fewer than n non-empty clusters. Thus, the resulting number of global models depends on the clustering

Journal Pre-proof

outcome rather than being fixed in advance.

After clustering, the *MergeDFFA* procedure is applied to each non-empty cluster. In each cluster, the models retain their original order in M and are merged using the aggregation function $F_{\text{MergeDFFA}}$ defined in Section 4.3, with unfolding bound d. Empty clusters are ignored, and each resulting merged DFFA is appended to the output sequence $\mathbb{M}$ (Line 9). Thus, *FedCGASPD* produces at most n models, each representing the behavior of one or more organizations assigned to the corresponding cluster.

4.5 Discovering Single Model with Selective Model Sharing

As discussed in the previous sections, federated process discovery through model merging enables organizations to collaboratively discover process models without sharing their raw event logs. This approach allows the discovered models to reflect the behavior captured in local event logs while reducing the exposure of sensitive information. However, sharing models instead of event logs does not eliminate privacy risks. Complete local models may still reveal rare or unique behaviors that could enable trace reconstruction or the re-identification of other sensitive business information.

The key idea of *P²FedGASPD* is to control the amount of information disclosed from each local model through a threshold τ. For a local DFFA A, τ specifies the maximum combined number of states and transitions that can be retained in the shared partial model. Given τ, the Maximum Frequency Partial DFFA (MFPDFFA) Problem selects a connected subset of the model that maximizes the total frequency of the selected states and transitions. The frequencies of the resulting partial model are then rebalanced to preserve consistency, and the model is sent to the server for merging. Each client evaluates the quality of the resulting global model against its local model. If the quality is insufficient, the client increases τ and extracts a new partial model with a larger size budget. Thus, τ controls the trade-off between the amount of model information disclosed and the quality of the resulting global model.

The following subsections detail the three main components of this approach. First, we define partial DFFAs and formulate the MFPDFFA Problem, which determines which states and transitions to retain under the size budget τ (Section 4.5.1). Second, we describe the frequency rebalancing procedure used to preserve consistency in the extracted partial model (Section 4.5.2). Finally, we present the *P²FedGASPD* algorithm, which iteratively adjusts τ, extracts partial models, and evaluates whether the resulting global model achieves sufficient quality relative to each client's local model (Section 4.5.3).

4.5.1 Partial DFFAs

A partial DFFA is extracted from each local model by selecting frequently observed behavior under a size budget. This approach prioritizes representative behavior while limiting the disclosure of rare or infrequent behaviors that could reveal sensitive information about individual process executions.

Definition 4.1 (Partial DFFAs)
Let $A = (Q, \Lambda, q_0, \mathbb{I}, \mathbb{F}, \delta)$ be a DFFA and $\tau \in \mathbb{N}$, $\tau \geq 1$, a size budget. A *partial DFFA*

Journal Pre-proof

of A under budget τ is a DFFA $A^P = \left(Q^P, \Lambda^P, q_0, \mathbb{I}^P, \mathbb{F}^P, \delta^P\right)$ such that $q_0 \in Q^P \subseteq Q$, $\Lambda^P \subseteq \Lambda$, $\mathbb{I}^P = \mathbb{I}|_{Q^P}$, $\mathbb{F}^P : Q^P \to \mathbb{R}_{\geq 0}$, $\delta^P : Q^P \times \Lambda^P \times Q^P \to \mathbb{R}_{\geq 0}$, $E^P \subseteq E$, where $E^P = \{(q, a, q') \mid \delta^P(q, a, q') > 0\}$, $E = \{(q, a, q') \mid \delta(q, a, q') > 0\}$, $|Q^P| + |E^P| \leq \tau$, and every state in Q^P is reachable from q_0 through transitions in E^P. ⌟

The objective of the Maximum Frequency Partial DFFA (MFPDFFA) problem is to select a partial DFFA that maximizes the total final-state and transition frequencies of its selected states and transitions, subject to a size budget τ on their combined number. MFPDFFA is related to the Maximum-Weight Connected Subgraph (MWCS) problem, which is NP-hard [20]. In MWCS, a connected subgraph of an undirected graph is selected to maximize the total weight of its vertices. In contrast, MFPDFFA considers a directed graph with weights on both states and transitions, requires all selected states to be reachable from the initial state, and imposes a size budget. We formulate MFPDFFA as a Mixed Integer Linear Programming (MILP) problem.

The decision variables x_q and $y_{q,a,q'}$, refer to Eqs. (C1) and (C2), are binary, indicating whether a state or transition is selected, respectively. Equation (C3) ensures that the combined number of selected states and transitions does not exceed τ. The initial state q_0 must be selected, cf. Eq. (C4), and a transition can only be selected if both its source and target states are selected, Eq. (C5).

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

$$y_{q,a,q'} \in \{0, 1\} \qquad \forall q, a, q' \,:\, \delta(q, a, q') > 0 \tag{C2}$$

$$\sum_{q \in Q} x_q + \sum_{(q,a,q') \,:\, \delta(q,a,q') > 0} y_{q,a,q'} \leq \tau \tag{C3}$$

$$x_{q_0} = 1 \tag{C4}$$

$$y_{q,a,q'} \leq x_q, \quad y_{q,a,q'} \leq x_{q'} \quad \forall (q, a, q') \,:\, \delta(q, a, q') > 0 \tag{C5}$$

To ensure that every selected state is reachable from the initial state q_0, we use a flow-based formulation [1]. For each transition (q, a, q') with $\delta(q, a, q') > 0$, we introduce a nonnegative flow variable $f_{q,a,q'}$. These variables represent auxiliary flows used to enforce reachability, rather than transition frequencies. Equation (C6) requires the initial state to supply one unit of flow for each selected non-initial state, with each such state consuming one unit. Equation (C7) restricts flow to selected transitions. Together with Equation (C5), these constraints ensure that every selected state is reachable from q_0 through selected transitions.

$$\sum_{(q,a,q') \,:\, \delta(q,a,q') > 0} f_{q,a,q'} - \sum_{(q',a,q) \,:\, \delta(q',a,q) > 0} f_{q',a,q} = \begin{cases} \sum_{q \in Q} x_q - 1 & q = q_0 \\ -x_q & q \neq q_0 \end{cases} \quad \forall q \in Q \tag{C6}$$

$$0 \leq f_{q,a,q'} \leq (\tau - 1) \cdot y_{q,a,q'} \quad \forall (q, a, q') \,:\, \delta(q, a, q') > 0 \tag{C7}$$

The objective function, refer to Equation (OF), maximizes the sum of the original final-state and transition frequencies of the selected components. It prioritizes frequently observed behavior while respecting the size and reachability constraints.

$$\max_{x,y} \quad \sum_{q \in Q} F(q) \cdot x_q + \sum_{(q,a,q') \,:\, \delta(q,a,q') > 0} \delta(q, a, q') \cdot y_{q,a,q'} \tag{OF}$$

Journal Pre-proof

We solve the MILP using the Google OR-Tools optimization library [25]. The resulting solution determines the states and transitions retained in the partial DFFA. Their frequencies are subsequently rebalanced as described in Section 4.5.2.

4.5.2 Frequency Rebalancing

After extracting A^P from A, its frequencies must be rebalanced to preserve consistency. Let $\mathbb{P}_q$ denote the final probability of state q, and $\mathbb{P}_{q,a,q'}$ denote the probability of a transition from state q to state q' via action a. These probabilities are calculated by dividing the final frequency $\mathbb{F}(q)$ and each outgoing transition frequency $\delta(q, a, q')$, respectively, by the sum of the final frequency and all outgoing transition frequencies of state q. We assume that this sum is positive for every state considered; states with zero total frequency can be excluded from the calculation. The resulting final and outgoing transition probabilities sum to one at each state.

When states and transitions are removed from A, the probabilities of the retained transitions may no longer sum to one together with the original final probability. To restore normalization, the probabilities of retained transitions are preserved, while those of removed transitions are set to zero. The remaining probability mass is assigned to the final probability of the corresponding source state. Thus, for every state $q \in Q^P$, the updated final probability is:

$$\mathbb{P}^P_q = 1 - \sum_{q' \in Q^P,\, a \in \Lambda^P} \mathbb{P}^P_{q,a,q'}.$$

The updated probabilities are then used to calculate consistent frequencies for A^P. Let $h(q)$ denote the total frequency reaching state q, including its initial frequency and the frequencies of all incoming transitions. The frequency reaching each state equals its initial frequency plus the frequencies propagated from its predecessor states. This gives the following system of linear equations:

$$h(q) = \mathbb{I}^P(q) + \sum_{r \in Q^P,\, a \in \Lambda^P} \mathbb{P}^P_{r,a,q}\, h(r), \qquad q \in Q^P.$$

Once this system is solved, the rebalanced frequencies are obtained by distributing the frequency reaching each state according to its updated probabilities:

$$\delta^P(q, a, q') = \mathbb{P}^P_{q,a,q'} \cdot h(q), \qquad \mathbb{F}^P(q) = \mathbb{P}^P_q \cdot h(q).$$

Since the final and outgoing transition probabilities sum to one at each state, the resulting frequencies satisfy the consistency condition in Definition 3.2. Consequently, the total final frequency equals the total initial frequency. The system of linear equations can be solved using standard numerical methods to obtain the rebalanced transition and final frequencies of A^P.

Figure 6 shows the transition and final probabilities of DFFA A_1 in Fig. 2. Figure 7 shows the updated probabilities of its partial DFFA A^P_1 for $\tau = 6$. Finally, Fig. 8 shows the rebalanced frequencies of A^P_1, rounded to the nearest integer for presentation.

Journal Pre-proof

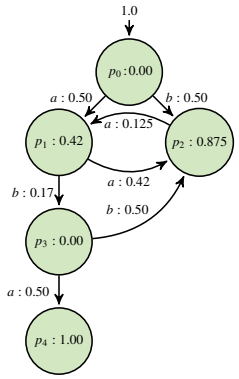


Figure 6: Transition and final probabilities of DFFA A_1.

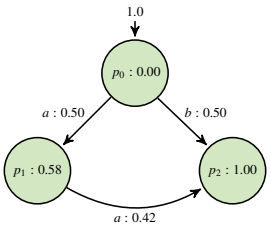


Figure 7: Updated probabilities of partial DFFA A_1^P.

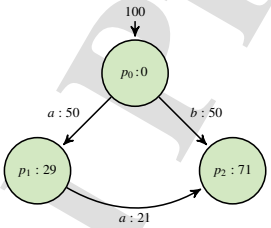


Figure 8: Rebalanced frequencies of partial DFFA A_1^P.

4.5.3 Discovery

The detailed *P*^{2}*FedGASPD* technique is presented in Algorithms 4 and 5. It discovers a global model A_G through two interacting procedures: the server and client phases.

Server phase: The server is responsible for merging the partial models sent by the clients, refer to Algorithm 4. First, the server initializes the global model and an update flag (Lines 1 and 2). Then, it iteratively invokes the *P*^{2}*FedGASPD* client algorithm for each client in parallel, providing the current global model A_G and the quality tolerance ϵ (Lines 4 and 5). In the first iteration, each client discovers its local model and returns an initial partial model. In subsequent iterations, each client evaluates the quality of the current global model and determines whether to update its partial model. Each client returns its current partial model and a flag indicating whether the model was computed during initialization or recomputed after increasing its size budget. After collecting the responses, the server constructs an ordered sequence of partial models and, if at least one client has performed an update, merges them using the aggregation function $F_{\text{MergeDFFA}}$ defined in Section 4.3, with unfolding bound d (Line 9). The updated global model is then provided to the clients in the next iteration. The procedure terminates when no client performs an update.

Journal Pre-proof

Client phase: Each client maintains its size budget τ, local model A, and partial model A^P between invocations by the server. Let $size(A) = |Q| + |E|$ denote the combined number of states and transitions in the local DFFA A, where E is its set of positive-frequency transitions.

When the client is invoked by the server for the first time, it initializes τ to 1 (Line 2), allowing the partial model to contain only the initial state. It then discovers the local model A using *GASPD* and the *SELECTDFFA* routine (Line 3). Next, the client extracts a partial model A^P by solving MFPDFFA under the size budget τ and rebalances its frequencies using the procedure described in Section 4.5.2 (Line 4). The client then returns the partial model and an update flag set to *true* (Line 5).

On subsequent invocations, the client compares $ER(A_G, L)$ with $ER(A, L)$, the entropic relevance of the global model and the local model A, respectively, with respect to its event log L (Line 6). If the entropic relevance of the global model does not exceed that of the local model by more than ϵ, the client returns its current partial model with the update flag set to *false*, indicating that no additional information needs to be shared (Line 7). Otherwise, the client checks whether τ has reached the size of the local model (Line 8). If so, no further disclosure is possible, and the client returns its current partial model without updating it (Line 9). Otherwise, the client increases τ by one (Line 10), extracts a new partial model under the increased size budget, and rebalances its frequencies (Line 11). The updated partial model is returned with the update flag set to *true* (Line 12).

Each client reevaluates the global model on every subsequent invocation because updates from other clients may change its quality. Thus, a client that previously satisfied its quality requirement may resume sharing additional information if the global model's quality deteriorates. The procedure terminates when every client either satisfies its quality requirement or has exhausted its available size budget.

Algorithm 4: P^2*FedGASPD* Server

Input: A non-empty list of clients $C = \langle C_1, \ldots, C_n \rangle$, quality tolerance $\epsilon \geq 0$, and unfolding bound $d \in \mathbb{N}$, $d \geq 1$, initial population size p, number of generations m, number of parents k to generate offspring

Output: A global DFFA A_G

1 $updated \leftarrow true$;
2 $A_G \leftarrow$ NIL;
3 **while** *updated* **do**
4 **parallelfor** $i \in [1 \,..\, n]$ **do**
5 $(M_i, u_i) \leftarrow P^2FedGASPD.Client.C_i(p, m, k, A_G, \epsilon)$;
6 $updated \leftarrow \bigvee_{i=1}^{n} u_i$;
7 **if** *updated* **then**
8 $M \leftarrow \langle M_1, \ldots, M_n \rangle$;
9 $A_G \leftarrow F_{MergeDFFA}(M, d)$;
10 **return** A_G;

Journal Pre-proof

Algorithm 5: P^2*FedGASPD* Client

Input: Initial population size p, number of generations m, number of parents k to generate offspring, global model A_G, quality tolerance ϵ, and local event log L supplied by the client
Output: A partial DFFA and an update flag

```
// τ, A, and A^P persist between invocations by the server
```

if *first invocation* **then**
  $\tau \leftarrow 1$;
  $A \leftarrow SELECTDFFA(GASPD(p, m, k, L), L)$;
  $A^P \leftarrow MFPDFFA(A, \tau)$; $A^P \leftarrow Rebalance\left(A, A^P\right)$;
  return $(A^P, true)$;
if $ER(A_G, L) - ER(A, L) \leq \epsilon$ **then**
  return $(A^P, false)$;
if $\tau \geq size(A)$ **then**
  return $(A^P, false)$;
$\tau \leftarrow \tau + 1$;
$A^P \leftarrow MFPDFFA(A, \tau)$; $A^P \leftarrow Rebalance\left(A, A^P\right)$;
return $(A^P, true)$;

5 Evaluation

This section evaluates our federated stochastic process discovery approaches, focusing on model quality, selective model sharing, and trace reconstruction ambiguity. Specifically, we address the following research questions:

- Do federated approaches based on complete model sharing (*FedGASPD* and *FedCGASPD*) discover models of comparable quality to those obtained through centralized discovery (*GASPD*)?
- Does partial model sharing (P^2*FedGASPD*) achieve global model quality comparable to complete model sharing (*FedGASPD*)?
- How does partial model sharing affect trace reconstruction ambiguity, and what trade-offs arise between model quality, disclosed model size, and privacy?

We first describe the implementation and datasets used in our experiments (Section 5.1). Next, we introduce a trace reconstruction attack model and define Trace Reconstruction Complexity (TRC) to assess privacy (Section 5.2). We then evaluate complete model sharing by comparing *FedGASPD* and *FedCGASPD* with centralized *GASPD* (Section 5.3). Finally, we evaluate partial model sharing (Section 5.4) in two scenarios: quality-guided sharing, where clients progressively disclose more information until the global model meets a quality requirement, and incremental sharing, where the proportion of disclosed model information is systematically increased to assess its effect on global model quality.

In all experiments, we set the initial population size $p = 50$, the number of generations $m = 50$, and the number of parents to generate offspring $k = 50$. For all federated

Journal Pre-proof

approaches, we used an unfolding bound of $d = 2$. In all experiments, local models were merged in the order they were provided, without applying any ordering strategy.

5.1 Implementation and Datasets

We implemented the *GASPD*, *FedGASPD*, *FedCGASPD*, and *P²FedGASPD* algorithms and made the implementation publicly available.[1] Using this implementation, we evaluated the performance of the algorithms on nine real-world event logs sourced from the IEEE Task Force on Process Mining.[2] For the experiments, we selected event logs that cover a broad range of characteristics, including the number of distinct activities, the number of unique traces, and the distribution of trace lengths. Such diversity is important for assessing the robustness and generalizability of our algorithms across different real-world scenarios. Table 1 summarizes the characteristics of the nine event logs used in the evaluation.

All experiments were executed on a machine equipped with a 13th Gen Intel® Core™ i7-1355U processor running at 1.70 GHz and 16 GB of RAM (15.7 GB of effective memory).

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

Event log	#Actions	#Traces	#Distinct traces	Max. trace length	Avg. trace length
BPI challenge 2012	24	13,087	4,366	175	20.03
BPI challenge 2013	4	7,554	1,511	123	8.67
BPI challenge 2017	26	31,509	15,930	180	38.15
BPI challenge 2018	34	2,861	2,498	680	61.58
BPI challenge 2019	42	251,734	11,973	990	6.33
Request for payment	34	6,449	753	27	11.18
Prepaid travel costs	19	6,886	89	20	5.34
Sepsis cases	16	1,050	846	185	14.48
Hospital billing	18	100,000	1,020	217	4.51

5.2 Privacy Assessment via Trace Reconstruction

In our federated process discovery approaches, clients do not share their event logs. Instead, they transmit complete or partial DFFAs to a central server for model merging. Although event logs remain local, the shared models may still expose behavioral information about the underlying process executions.

We consider an attacker who has access to a disclosed model but not to the underlying event log or case-specific background knowledge. In particular, the attacker does not know which traces permitted by the model actually occurred in the original event log. Although DFFAs contain frequency information, our attack model considers only the behavior permitted by the disclosed model, disregarding frequency values.

Definition 5.1 (Trace Reconstruction Attack)
Let M be a DFFA disclosed by a client, and let lang(M) denote its language. The

[1] https://github.com/HzhianUnimelb/PrivacyFedProcessMining_Sources
[2] https://www.tf-pm.org/resources/logs

Journal Pre-proof

attacker attempts to identify which traces in lang(M) are present in the event log of the client. Since the disclosed model may permit behavior that was not observed, not every candidate trace necessarily belongs to the original event log. ⌟

To assess reconstruction ambiguity under this attack model, we use precision, refer to Section 3.2. Precision measures the extent to which a model avoids permitting behavior beyond that observed in an event log. Intuitively, the additional behavior permitted by a disclosed model introduces candidate traces that may not have occurred in the original event log, making it more difficult to distinguish observed traces from unobserved ones. We define Trace Reconstruction Complexity (TRC) as the complement of precision.

Let L be the event log of a client and M its disclosed model. TRC is defined as:

$$TRC(L, M) = 1 - \text{precision}(L, M)\,.$$

TRC measures the structural ambiguity introduced by behavior permitted by the disclosed model beyond the original event log. When the model permits no behavior beyond the observed traces, precision is one and TRC is zero. As the model permits additional unobserved behavior, precision decreases, resulting in higher TRC values. Thus, higher TRC values indicate greater behavioral ambiguity, which may make it more difficult for the attacker to identify which model traces were actually observed.

TRC is a structural measure rather than a probability of successful trace reconstruction. It is computed using the original event log, which is available to the evaluator but not to the attacker. TRC does not quantify the privacy of individual cases, nor does it capture information that might be inferred from the disclosed frequency values. These limitations delimit the scope of the proposed privacy assessment.

5.3 Complete Model Sharing

To construct human-readable models, we controlled the parameter that excludes infrequent traces. Consequently, we constructed models of at most 1,000 states and transitions combined, with many models substantially smaller than this target size.

To simulate cross-organizational environments, we randomly split each event log among a requested number of clients, ensuring each client receives approximately the same portion of the event log traces. We implemented the *SELECTDFFA* function from Algorithms 2 and 3 by selecting one DFFA from a Pareto frontier of DFFAs that minimizes the objective function $(size^* + relevance^*)/2$, where $size^*$ and $relevance^*$ are the normalized size and entropic relevance of the model, respectively, with normalization performed as described in Section 4.4. This way, we select a model that balances the two competing objectives equally.

Table 2 presents the execution times of *GASPD* and its federated alternatives. Both *FedGASPD* and *FedCGASPD* reduce execution times across all evaluated event logs, although the improvements depend on the dataset and number of clients. For example, on the BPI challenge 2018 log, the federated approaches achieve speedups of up to approximately 38 times, while on the BPI challenge 2013 log, they achieve speedups ranging from approximately 2 to 8 times. These improvements arise primarily from executing *GASPD* in parallel on smaller local event logs, followed by model merging.

Journal Pre-proof

Table 2: Execution times of *GASPD*, *FedGASPD*, and *FedCGASPD* algorithms.

Event log	Execution time (in seconds)						
	GASPD	*FedGASPD*			*FedCGASPD*		
		2 clients	4 clients	8 clients	2 clients	4 clients	8 clients
BPI challenge 2012	416	178	83	61	166	81	58
BPI challenge 2013	84	35	26	11	43	22	17
BPI challenge 2017	3,120	1,459	514	342	1,401	499	367
BPI challenge 2018	2,898	1,325	165	76	1,317	173	78
BPI challenge 2019	2,288	2,216	471	251	2,224	480	260
Request for payment	1,402	685	331	229	676	342	236
Prepaid travel costs	175	94	57	46	92	53	42
Sepsis cases	421	234	64	31	237	60	36
Hospital billing	61	32	20	16	35	26	18

Table 3: Size of DFFAs discovered by *GASPD*, *FedGASPD*, and *FedCGASPD*.

Event Log	Size											
	GASPD		*FedGASPD*					*FedCGASPD*				
			2 clients		4 clients		8 clients	2 clients		4 clients		8 clients
	min	max	min	max	min	max	min	max	min	max	min	max	min	max
BPI challenge 2012	185	930	128	856	173	361	52	249	62	446	50	315	11	205
BPI challenge 2013	486	760	143	715	174	373	130	291	107	311	67	218	47	190
BPI challenge 2017	143	460	166	507	100	184	98	121	125	750	56	154	48	121
BPI challenge 2018	196	403	120	339	81	282	59	125	98	318	27	218	8	66
BPI challenge 2019	158	808	481	766	171	503	127	230	229	632	116	572	47	218
Request for payment	313	509	314	515	221	465	187	369	195	421	173	321	151	235
Prepaid travel costs	275	629	234	459	247	362	196	283	213	345	153	266	90	241
Sepsis cases	179	786	201	549	179	673	132	391	132	189	149	314	43	133
Hospital billing	21	71	13	156	20	160	20	141	32	70	16	110	20	92

Figure 9 shows the characteristics of the Pareto-optimal models discovered from the nine event logs by the three algorithms for different numbers of clients, while Tables 3 and 4 summarize the ranges of model sizes and entropic relevance values. These results further support our contention that federated approaches can discover models with characteristics comparable to those obtained through centralized discovery. When comparing the two federated approaches, *FedCGASPD* more often produces smaller models than *FedGASPD*, as it clusters models with similar size and entropic relevance characteristics before merging them. *FedGASPD*, however, tends to discover models with lower entropic relevance, indicating a better description of the observed traces and their frequencies. This observation illustrates the trade-off between model size and quality [29] and suggests that the methods favor different aspects of this trade-off.

We conclude that *FedGASPD* and *FedCGASPD* discover models with characteristics comparable to those produced by *GASPD*. Table 5 presents Pareto dominance counts for the various techniques. The results show that the centralized approach, *GASPD*, does not consistently dominate across different datasets. Specifically, it contributes a substantial share of Pareto-optimal solutions for BPI challenge 2017, Prepaid travel costs, Sepsis cases, and Hospital billing, whereas *FedCGASPD* contributes the largest share of Pareto-optimal solutions for the remaining event logs.

Overall, the results show that *FedGASPD* and *FedCGASPD* can discover models with size and entropic relevance comparable to those produced by centralized *GASPD*,

Journal Pre-proof

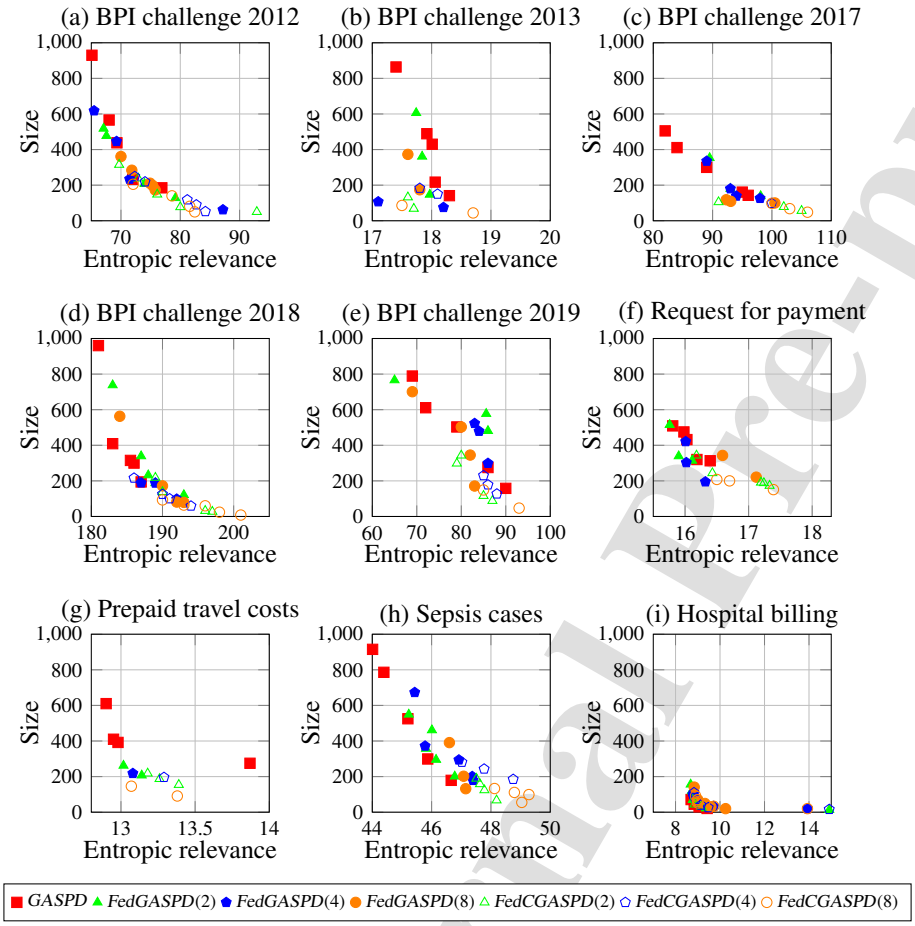


Figure 9: Size and entropic relevance of DFFAs discovered by *GASPD*, *FedGASPD*, and *FedCGASPD*; smaller sizes and entropic relevance values signify better models. Numbers in parentheses indicate the number of clients

while substantially reducing execution time. *FedGASPD* generally achieves lower relevance than *FedCGASPD*, whereas *FedCGASPD* often discovers smaller models. These findings suggest that federated discovery provides a competitive alternative to centralized discovery, with different trade-offs between model size and quality.

5.4 Partial Model Sharing

Next, we evaluate federated process discovery based on partial model sharing. In the first scenario (quality-guided sharing), we compare *P*^{2}*FedGASPD* with *FedGASPD* by incrementally increasing the size of the partial DFFA each client shares with the central server until the difference in entropic relevance between the global and local

Journal Pre-proof

Table 4: Relevance of DFFAs discovered by *GASPD*, *FedGASPD*, and *FedCGASPD*.

Event Log	Entropic relevance													
	GASPD		*FedGASPD*						*FedCGASPD*					
	min	max	2 clients		4 clients		8 clients		2 clients		4 clients		8 clients	
			min	max	min	max	min	max	min	max	min	max	min	max
BPI challenge 2012	65.11	76.89	64.84	79.13	70.00	75.17	72.32	84.23	65.47	87.19	72.91	92.90	72.05	98.00
BPI challenge 2013	17.40	18.10	17.72	18.25	17.65	18.46	17.82	18.71	16.33	18.29	17.70	18.83	17.51	20.77
BPI challenge 2017	82.01	93.34	89.79	95.42	92.63	103.04	100.88	105.45	89.69	100.18	93.88	106.57	101.31	110.33
BPI challenge 2018	185.26	187.42	187.77	193.08	191.87	194.16	190.44	197.13	189.25	193.34	192.33	198.58	193.01	205.07
BPI challenge 2019	61.01	87.27	64.77	87.39	69.51	88.59	70.65	89.66	76.40	88.69	81.56	92.22	83.39	94.73
Request for payment	15.80	16.39	15.75	16.30	16.59	17.13	16.30	17.61	16.00	16.75	16.18	17.33	16.70	18.35
Prepaid travel costs	12.95	13.78	13.07	13.47	13.09	13.85	13.40	13.53	13.06	13.25	13.17	13.44	13.40	13.75
Sepsis cases	44.01	46.66	45.23	46.67	45.24	47.45	46.60	47.51	47.45	48.32	47.01	48.85	48.12	50.03
Hospital billing	08.70	09.42	08.68	14.91	08.78	13.93	08.84	10.25	08.78	09.16	08.83	14.91	08.90	13.93

Table 5: Pareto dominance counts.

Event log	*GASPD*	*FedGASPD*			*FedCGASPD*			Total score
		2 clients	4 clients	8 clients	2 clients	4 clients	8 clients	
BPI challenge 2012	2 (13.33%)	**4 (26.66%)**	2 (13.33%)	0 (00.00%)	3 (20.00%)	0 (00.00%)	**4 (26.66%)**	15 (100.00%)
BPI challenge 2013	0 (00.00%)	0 (00.00%)	1 (25.00%)	0 (00.00%)	1 (25.00%)	0 (00.00%)	**2 (50.00%)**	04 (100.00%)
BPI challenge 2017	**3 (30.00%)**	0 (00.00%)	0 (00.00%)	0 (00.00%)	**3 (30.00%)**	1 (10.00%)	**3 (30.00%)**	10 (100.00%)
BPI challenge 2018	3 (15.00%)	1 (05.00%)	3 (15.00%)	3 (15.00%)	3 (15.00%)	3 (15.00%)	**4 (20.00%)**	20 (100.00%)
BPI challenge 2019	2 (20.00%)	1 (10.00%)	0 (00.00%)	2 (20.00%)	**3 (30.00%)**	0 (00.00%)	2 (20.00%)	10 (100.00%)
Request for payment	1 (10.00%)	2 (20.00%)	2 (20.00%)	0 (00.00%)	**3 (30.00%)**	1 (10.00%)	1 (10.00%)	10 (100.00%)
Prepaid travel costs	**4 (57.14%)**	1 (14.28%)	0 (00.00%)	0 (00.00%)	0 (00.00%)	0 (00.00%)	2 (28.58%)	07 (100.00%)
Sepsis cases	**5 (38.46%)**	2 (15.38%)	1 (07.69%)	1 (07.69%)	2 (15.38%)	0 (00.00%)	2 (15.38%)	13 (100.00%)
Hospital billing	**4 (28.57%)**	**4 (28.57%)**	0 (00.00%)	1 (07.14%)	3 (21.42%)	2 (14.28%)	0 (00.00%)	14 (100.00%)

models falls below a threshold (ϵ). In the second scenario (incremental sharing), all clients share the same percentage of their local models with the central server, gradually increasing this percentage until they share their complete models. This scenario lets us examine how the quality of the global model changes as clients contribute more information.

5.4.1 Quality-Guided Sharing

In this scenario, we limit the model size to at most 200 and set $\epsilon = 0.5$. A client does not increase its sharing budget if the entropic relevance of the global model exceeds that of its local model by no more than 0.5. Otherwise, the client continues increasing its sharing budget until the quality requirement is satisfied or the maximum permitted size is reached.

Figure 10 shows the characteristics of the Pareto-optimal models discovered from the nine event logs by *P*^{2}*FedGASPD* and *FedGASPD* for different numbers of clients, while Table 6 compares their Pareto dominance results. Each column in the table indicates the difference in the number of Pareto-optimal models obtained by *P*^{2}*FedGASPD* and *FedGASPD*; for example, +2 means *P*^{2}*FedGASPD* has two more Pareto-optimal models than *FedGASPD*.

Table 8 shows the average percentage of each local model shared by the clients. For instance, 76.80% means that clients share, on average, 76.80% of their local models

Journal Pre-proof

when the procedure terminates, either because their quality requirements are satisfied or because no further information can be disclosed. *P²FedGASPD* often discovers models with lower entropic relevance and smaller size than *FedGASPD*.

As shown in Table 8, the average proportion of each local model shared decreases as the number of clients increases across all event logs. However, the amount of sharing varies substantially between datasets. For example, for eight clients, BPI challenge 2013 requires sharing only 27.75% of each local model on average, whereas BPI challenge 2017 and Prepaid travel costs require sharing more than 92%. These results suggest that the potential reduction in model disclosure depends on the characteristics of the underlying process. This phenomenon may also arise from sharing only partial models that capture the core and most representative behaviors of each client, thereby limiting the propagation of exceptional or infrequent behaviors. If a behavior is rare on one client but frequent on another, it may still be retained through the contribution of the latter. Behaviors that are infrequent across all clients are less likely to appear in the merged global model. This collective preservation of common behavior may contribute to the smaller and lower-relevance global models observed for *P²FedGASPD*.

The TRC results are summarized in Fig. 11 and Table 7. Figure 11 presents the relationship between model size and TRC for *P²FedGASPD* and *FedGASPD*. As discussed in Section 5.2, model size and TRC capture different properties. Therefore, reducing the size of a shared model does not necessarily increase or decrease TRC. TRC depends on the additional behavior permitted by the disclosed model relative to the original event log. The comparison in Fig. 11 shows that *P²FedGASPD* often achieves TRC values comparable to or higher than those of *FedGASPD* with smaller models across datasets and client configurations. This shows that achieving higher TRC does not necessarily require increasing the size of the shared model. *P²FedGASPD* prioritizes frequently observed behavior while excluding some less frequent branches. This limits the disclosure of uncommon execution paths while potentially maintaining greater reconstruction ambiguity under the proposed attack model.

The results in Table 7 further support this observation. The table compares the Pareto-optimal models of *FedGASPD* and *P²FedGASPD* based on model size and TRC. *P²FedGASPD* has more non-dominated solutions than *FedGASPD* for all datasets and client configurations. For example, for Sepsis cases, *P²FedGASPD* has 9, 10, and 7 more non-dominated solutions for 2, 4, and 8 clients, respectively. Similarly, for BPI challenge 2019, it has 5, 9, and 5 more non-dominated solutions for 2, 4, and 8 clients, respectively. These results indicate that *P²FedGASPD* contributes more non-dominated solutions with respect to model size and TRC across the evaluated configurations, demonstrating its potential to discover compact models with high reconstruction ambiguity under the proposed attack model.

5.4.2 Incremental Sharing

In the second scenario, all clients share the same proportion of their local models with the central server, beginning at 5% and increasing in increments of 5% until the entire local model is shared. At each sharing level, a global model is constructed from the partially shared information, and its quality is compared against the global model obtained when all clients share 100% of their local models. By measuring the quality

Journal Pre-proof

Table 6: Pareto dominance counts comparison based on size and entropic relevance.

Event log	P^2FedGASPD − FedGASPD		
	2 clients	4 clients	8 clients
BPI challenge 2012	+2	+3	+5
BPI challenge 2013	+4	+5	+5
BPI challenge 2017	-1	+3	+1
BPI challenge 2018	-1	+4	+1
BPI challenge 2019	-2	+4	+2
Request for payment	0	+3	+4
Prepaid travel costs	+3	+1	+5
Sepsis cases	+1	+5	+7
Hospital billing	+8	+3	+8

Table 7: Pareto dominance counts comparison based on size and TRC.

Event log	P^2FedGASPD − FedGASPD		
	2 clients	4 clients	8 clients
BPI challenge 2012	+5	+5	+6
BPI challenge 2013	+4	+4	+3
BPI challenge 2017	+6	+4	+8
BPI challenge 2018	+5	+5	+5
BPI challenge 2019	+5	+9	+5
Request for payment	+8	+4	+7
Prepaid travel costs	+6	+6	+7
Sepsis cases	+9	+10	+7
Hospital billing	+6	+2	+9

Table 8: Average local model sharing.

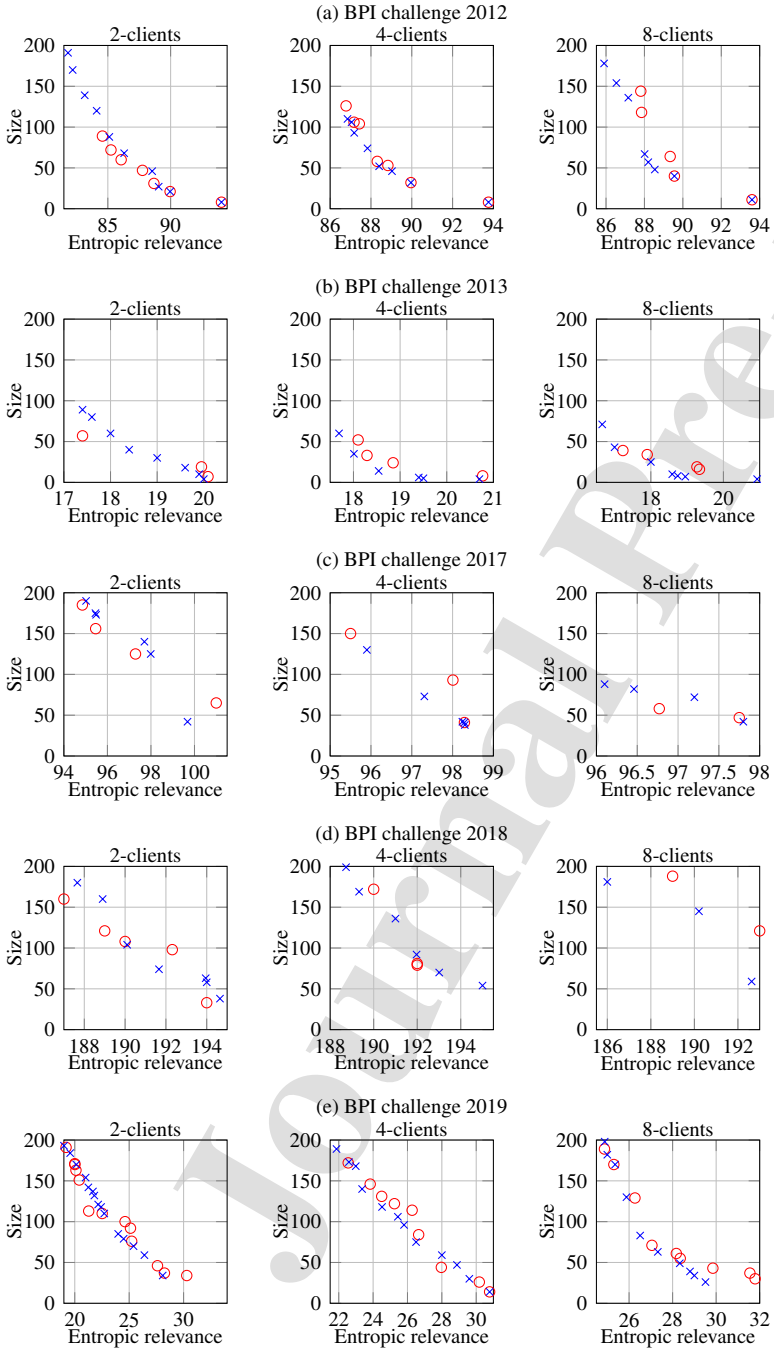
Event log	Sharing percentage (%)		
	P^2FedGASPD		
	2 clients	4 clients	8 clients
BPI challenge 2012	76.80	71.12	61.14
BPI challenge 2013	56.25	42.50	27.75
BPI challenge 2017	97.16	94.66	92.33
BPI challenge 2018	86.33	70.00	53.14
BPI challenge 2019	70.66	68.50	61.88
Request for payment	81.83	77.54	60.90
Prepaid travel costs	95.37	94.88	92.00
Sepsis cases	58.75	50.33	44.00
Hospital billing	80.40	75.20	65.88

difference at each stage, we assess how global model quality changes as more information becomes available.

As expected, once the sharing level reaches 100%, the resulting global model quality matches that obtained under full model sharing. Figure 12 shows the difference in the average entropic relevance between the global models obtained under partial and complete sharing, denoted ΔER, together with the size of the global model. Negative ΔER values indicate that the global model obtained under partial sharing achieves lower entropic relevance than the one obtained under complete sharing and, therefore, higher model quality. For example, in Figure 12(c) (the ΔER plot), for the 4-clients scenario, sharing 85% of each local model yields a global model whose average entropic relevance is lower than that of the global model obtained under full sharing ($\Delta\text{ER} \approx -2$). Figure 12 also reports the size of the global model at different sharing levels, showing how partial sharing leads to more compact models.

Overall, *P^2FedGASPD* can reduce the amount of shared model information while maintaining global model quality comparable to that of *FedGASPD*. By prioritizing frequently observed behavior, selective model sharing can limit the propagation of infrequent behaviors into the merged model, often resulting in smaller global models. The TRC results further indicate that *P^2FedGASPD* can achieve favorable trade-offs between disclosed model size and structural reconstruction ambiguity under the proposed attack model. These findings demonstrate the potential benefits of selective model sharing, although the resulting model quality and reconstruction ambiguity de-

Journal Pre-proof



Journal Pre-proof

Journal Pre-proof

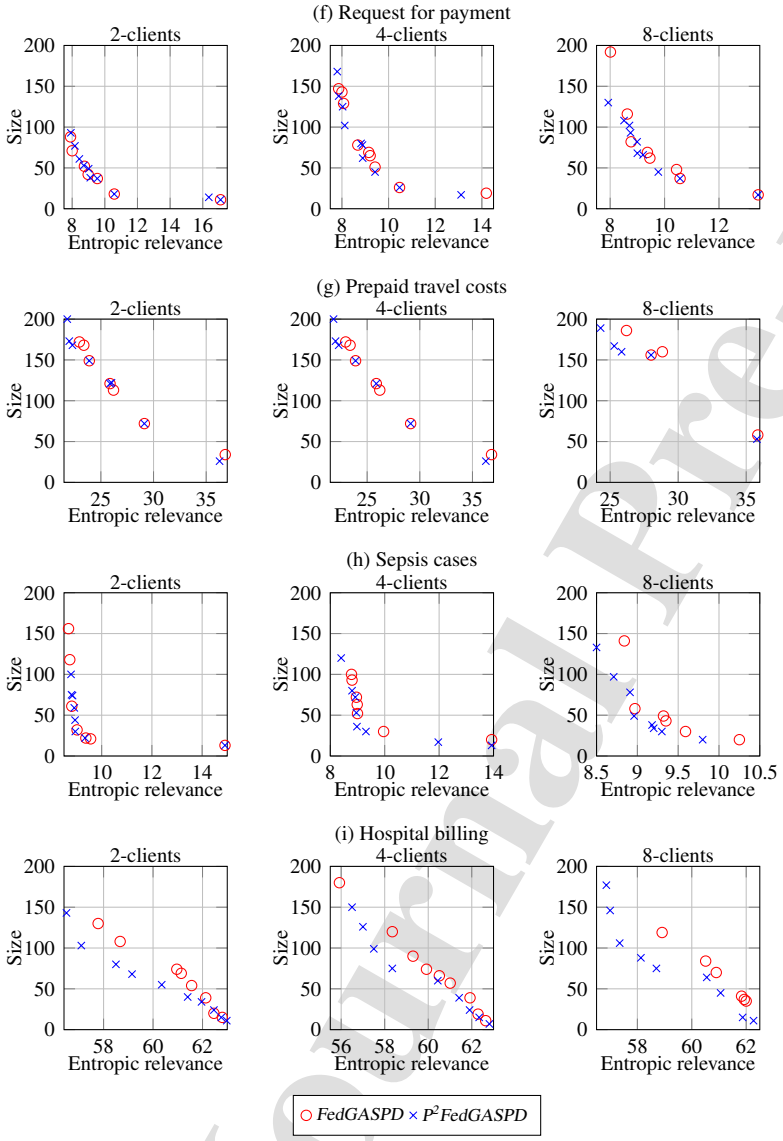
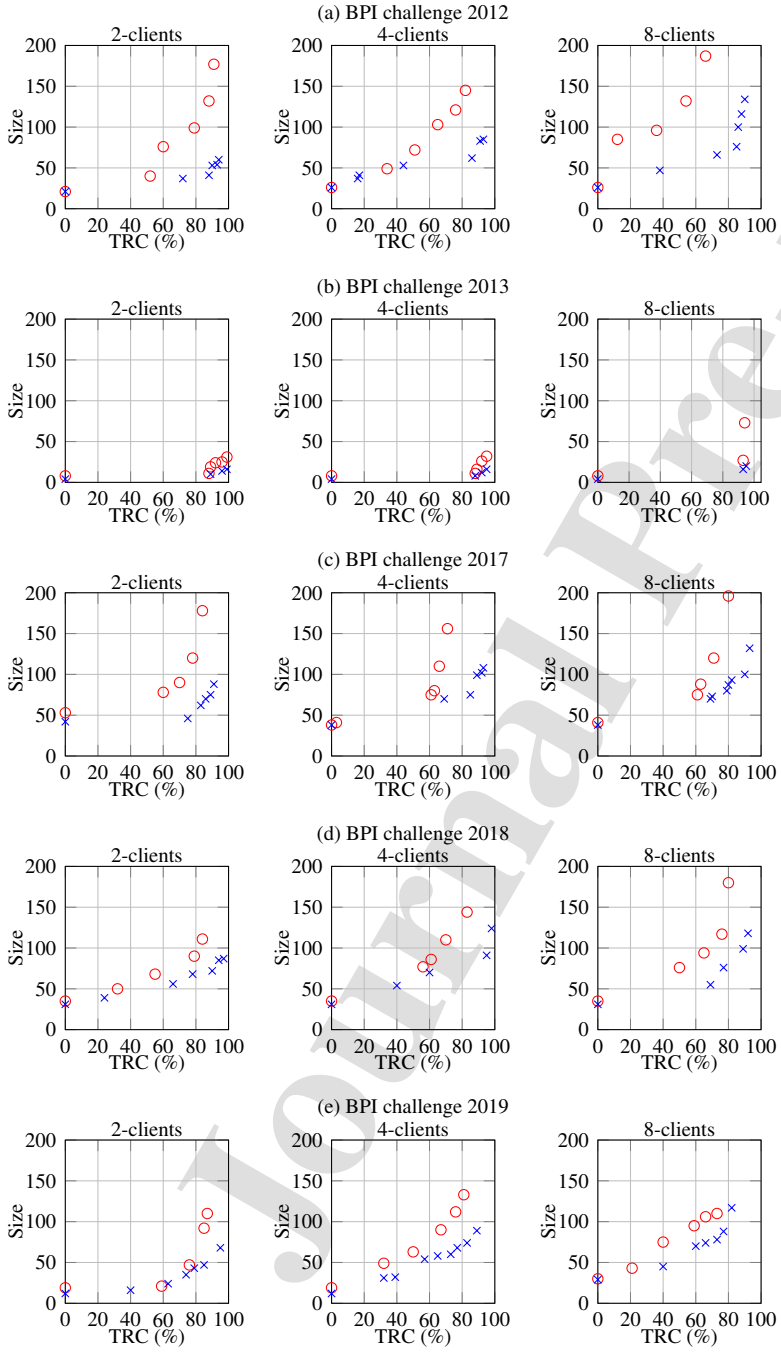


Figure 10: Size and entropic relevance of DFFAs discovered by *FedGASPD* and *P²FedGASPD*; smaller model sizes and lower entropic relevance values are preferred.

pend on the dataset and sharing configuration.

Journal Pre-proof



Journal Pre-proof

Journal Pre-proof

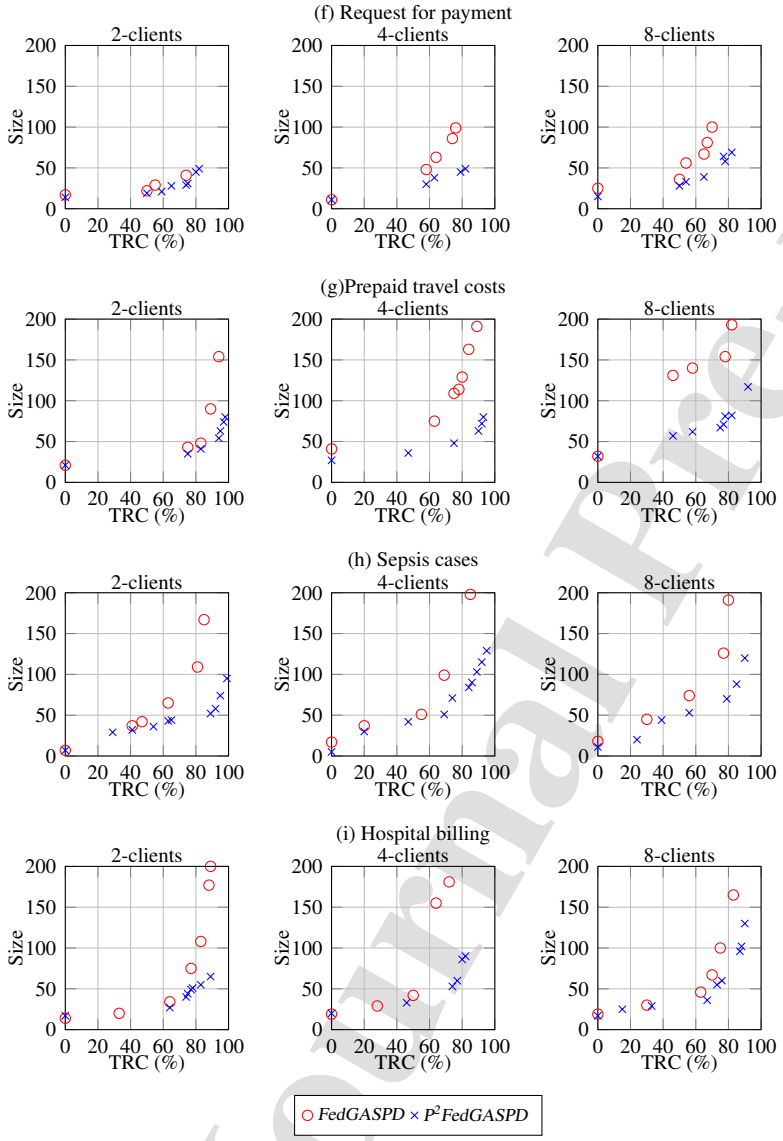


Figure 11: Size and TRC of local DFFAs disclosed to the server by *FedGASPD* and *P²FedGASPD*, with TRC evaluated against the corresponding client event logs. TRC values are expressed as percentages; smaller model sizes and higher TRC values are preferred under the proposed attack model.

Journal Pre-proof

Journal Pre-proof

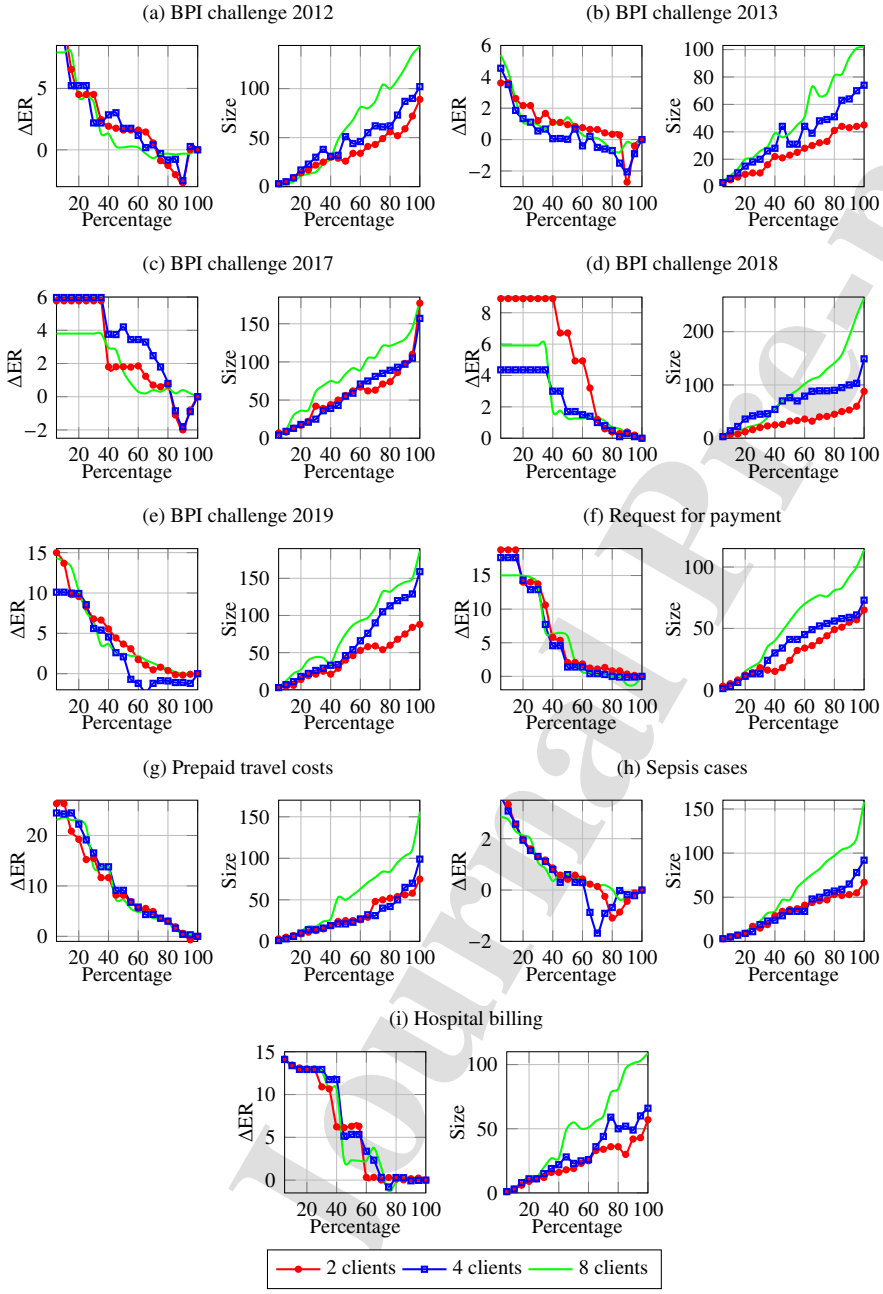


Figure 12: Impact of increased local model sharing on merged model quality.

Journal Pre-proof

6 Conclusions

This paper presents *FedGASPD* and *FedCGASPD*, two algorithms for federated stochastic process discovery. These algorithms build upon *GASPD*, a stochastic process discovery algorithm that operates on centralized event logs. The proposed algorithms discover process models from local event logs distributed across different organizations, preserving their data autonomy and avoiding the transfer of raw event data. They employ a model merging technique that preserves frequency consistency and relevant behavioral properties, producing global DFFAs that can be translated into DFGs using the native translation procedure of *GASPD*. Experiments on real-world event logs demonstrate that *FedGASPD* and *FedCGASPD* discover models with quality comparable to centralized discovery while substantially reducing execution times in many configurations. These results establish model-based federated discovery as an effective alternative to centralized stochastic process discovery.

While federated discovery avoids sharing raw event data, sharing complete local models can still expose rare or sensitive behaviors. To address this limitation, we introduce *P²FedGASPD*, a privacy-aware approach that extends federated discovery with selective model sharing. *P²FedGASPD* lets each client disclose only a portion of its local model, prioritizing frequent and representative behaviors while limiting disclosure of infrequent ones. Partial model selection is formulated as a size-constrained graph optimization problem related to maximum-weight connected subgraph selection. The selected models are rebalanced to preserve frequency consistency, and the amount of shared information is adjusted iteratively based on global model quality. We also introduce Trace Reconstruction Complexity (TRC), a precision-based measure that quantifies structural reconstruction ambiguity under a specified trace reconstruction attack model. Experiments demonstrate that *P²FedGASPD* can substantially reduce the amount of shared model information while maintaining global model quality comparable to complete model sharing. Moreover, the results show that selective model sharing can achieve favorable trade-offs between disclosed model size and reconstruction ambiguity.

By combining federated stochastic process discovery with selective model sharing, our work demonstrates that organizations can collaboratively discover high-quality process models while controlling the amount of behavioral information they disclose. The proposed approach represents a step towards privacy-aware process discovery in which disclosure is guided by the quality requirements of the resulting global model. Future work will investigate additional privacy dimensions and more powerful attack models, including re-identification risks and attacks that exploit information accumulated across successive disclosures. One can also study how partial models shared by different clients may be combined to infer information about individual organizations.

Several directions remain for further improving the proposed discovery algorithms. First, systematically investigating how different merging orders affect the quality of global DFFAs and their corresponding DFGs may reveal opportunities to improve model quality. Second, exploring alternative clustering techniques could improve the effectiveness of *FedCGASPD*. Third, extending model optimization beyond size and entropic relevance to other quality criteria could broaden the applicability of the proposed approaches. Finally, investigating alternative policies for selecting local models

Journal Pre-proof

and determining which model information to share could further improve the trade-offs between global model quality and information disclosure.

References

[1] Deepesh Agarwal, Julio-Cesar Silva Araujo, Christelle Caillouet, Frederic Cazals, David Coudert, and Stephane Pérennes. Connectivity inference in mass spectrometry based structure determination. In *Algorithms – ESA 2013*, volume 8125 of *LNCS*, pages 289–300. 21st ESA, 2013.

[2] Hanan Alkhammash, Artem Polyvyanyy, Alistair Moffat, and Luciano García-Bañuelos. Entropic relevance: A mechanism for measuring stochastic process models discovered from event data. *Information Systems*, 107:101922, 2022.

[3] Hanan Alkhammash, Artem Polyvyanyy, and Alistair Moffat. Stochastic directly-follows process discovery using grammatical inference. In *CAiSE*, volume 14663 of *LNCS*, pages 87–103. Springer, 2024.

[4] Joos C. A. M. Buijs, Boudewijn F. van Dongen, and W.M.P. van der Aalst. Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity. *International Journal of Cooperative Information Systems*, 23(01):1440001, 2014.

[5] Andrea Burattin, Mauro Conti, and Daniele Turato. Toward an anonymous process mining. In *3rd International Conference on Future Internet of Things and Cloud*, pages 58–63. IEEE, 2015.

[6] Josep Carmona, Jordi Cortadella, and Michael Kishinevsky. Divide-and-conquer strategies for process mining. In *BPM*, volume 5701 of *LNCS*, pages 327–343. Springer, 2009.

[7] Rafael C. Carrasco and Jose Oncina. Learning stochastic regular grammars by means of a state merging method. In *Grammatical Inference and Applications*, volume 862 of *LNCS*, pages 139–152. Springer, 1994.

[8] Colin de la Higuera. *Grammatical Inference*. Cambridge University Press, Cambridge, 2010.

[9] Gamal Elkoumy, Alisa Pankova, and Marlon Dumas. Privacy-preserving directly-follows graphs: Balancing risk and utility in process mining. *CoRR*, abs/2012.01119, 2020.

[10] Gamal Elkoumy, Stephan A. Fahrenkrog-Petersen, Mohammadreza Fani Sani, Agnes Kochmider, Saskia Nuñez von Voigt, Majid Rafiei, and Leopold von Waldthausen. Privacy and confidentiality in process mining - threats and research challenges. *ACM Transactions on Management Information Systems*, 13:1–17, 2021.

[11] Gamal Elkoumy, Alisa Pankova, and Marlon Dumas. Differentially private release of event logs for process mining. *Information Systems*, 115, 2023.

[12] Javier Esparza and Keijo Heljanko. *Unfoldings: A Partial-Order Approach to Model Checking*. EATCS Monographs in Theoretical Computer Science. Springer, 2008.

[13] European Parliament and Council of the European Union. Regulation (EU) 2016/679 of the European Parliament and of the Council of 27 April 2016 on the protection of natural persons with regard to the processing of personal data and on the free movement of such data, and repealing Directive 95/46/EC (General Data Protection Regulation). *Official Journal of the European Union*, L 119:1–88, 2016. URL https://eur-lex.europa.eu/eli/reg/2016/679/oj.

[14] Stephan A. Fahrenkrog-Petersen, Han van der Aa, and Matthias Weidlich. PRIPEL: Privacy-preserving event log publishing including contextual information. In *Business Process Management*, LNCS, pages 111–128. Springer, 2020.

[15] Stephan A. Fahrenkrog-Petersen, Martin Kabierski, Fabian Rosel, Han van der Aa, and

Journal Pre-proof

Matthias Weidlich. SaCoFa: Semantics-aware control-flow anonymization for process mining. In *3rd International Conference on Process Mining*, pages 72–79. IEEE, 2021.

[16] Martin Kabierski, Stephan A. Fahrenkrog-Petersen, and Matthias Weidlich. Hiding in the forest: Privacy-preserving process performance indicators. *Information Systems*, 112, 2023.

[17] Asjad Khan, Aditya Ghose, and Hoa Khanh Dam. Cross-silo process mining with federated learning. In *ICSOC*, volume 13121 of *LNCS*, pages 612–626. Springer, 2021.

[18] Sander J. J. Leemans, Xixi Lu, David Chapela-Campa, Claudio Di Ciccio, Izack Cohen, Benoît Depaire, Stephan A. Fahrenkrog-Petersen, Avigdor Gal, Massimiliano de Leoni, Henrik Leopold, Orlenys López-Pintado, Felix Mannhardt, Fabrizio Maria Maggi, Niels Martin, Marco Montali, Marco Pegoraro, Artem Polyvyanyy, Arik Senderovich, Sam Verboven, Akio Watanabe, Matthias Weidlich, and Jan Martijn E. M. van der Werf. Stochastic process mining: Characteristics and challenges. *IEEE Trans. Knowl. Data Eng.*, 38(10): 7061–7079, 2026.

[19] Jungeun Lim, Stephan A. Fahrenkrog-Petersen, Xixi Lu, Jan Mendling, and Minseok Song. The impact of event data partitioning on privacy-aware process discovery. In *International Conference on Business Process Management*, volume 565 of *LNBIP*, pages 7–22. Springer, 2025.

[20] Alexander A. Loboda, Maxim N. Artyomov, and Alexey A. Sergushichev. Solving generalized maximum-weight connected subgraph problem for network enrichment analysis. In *Algorithms in Bioinformatics*, volume 9838 of *LNCS*, pages 210–221. Springer, 2016.

[21] J. B. MacQueen. Some methods for classification and analysis of multivariate observations. In *Berkeley Symposium on Mathematical Statistics and Probability*, volume 1, pages 281–297. University of California Press, 1967.

[22] Felix Mannhardt, Sobah Abbas Petersen, and Manuel Fradinho Oliveira. Privacy challenges for process mining in human-centered industrial environments. In *14th International Conference on Intelligent Environments (IE)*, pages 64–71. IEEE, 2018.

[23] Felix Mannhardt, Agnes Koschmider, Nathalie Baracaldo, Matthias Weidlich, and Judith Michael. Privacy preserving process mining: Differential privacy for event logs. *Business & Information Systems Engineering*, pages 595–614, 2019.

[24] Judith Michael, Agnes Koschmider, Felix Mannhardt, Nathalie Baracaldo, and Bernhard Rumpe. User-centered and privacy-driven process mining system design for IoT. In *Information Systems Engineering in Responsible Information Systems*, volume 350 of *LNBIP*, pages 194–206. Springer, 2019.

[25] Laurent Perron and Vincent Furnon. OR-tools. URL https://developers.google.com/optimization/.

[26] Anastasiia Pika, Moe T. Wynn, Stephanus Budiono, Arthur H. M. ter Hofstede, Christian Rennert, Julian Albers, W.M.P. van der Aalst, and Hajo A. Reijers. Towards privacy-preserving process mining in healthcare. In *Business Process Management Workshops*, volume 362 of *LNCS*, pages 483–495. Springer, 2020.

[27] Artem Polyvyanyy. Notes of a process scientist: Process discovery. In *Mining a Scientist's Process*, volume 16480 of *LNCS*, pages 368–383. Springer, 2026.

[28] Artem Polyvyanyy, Jussi Vanhatalo, and Hagen Völzer. Simplified computation and generalization of the refined process structure tree. In *WS-FM*, volume 6551 of *LNCS*, pages 25–41. Springer, 2010.

[29] Artem Polyvyanyy, Alistair Moffat, and Luciano Garcia-Banuelos. An entropic relevance measure for stochastic conformance checking in process mining. In *ICPM*, pages 97–104. IEEE, 2020.

[30] Artem Polyvyanyy, Andreas Solti, Matthias Weidlich, Claudio Di Ciccio, and Jan Mendling. Monotone precision and recall measures for comparing executions and specifi-

Journal Pre-proof

cations of dynamic systems. *ACM Trans. Softw. Eng. Methodol.*, 29(3):17:1–17:41, 2020.

[31] Artem Polyvyanyy, Alistair Moffat, and Luciano García-Bañuelos. Bootstrapping generalization of process models discovered from event data. In *CAiSE*, volume 13295 of *LNCS*, pages 36–54. Springer, 2022.

[32] Majid Rafiei and W.M.P. van der Aalst. Mining roles from event logs while preserving privacy. In *Business Process Management Workshops*, volume 362 of *LNBIP*, pages 676–689. Springer, 2019.

[33] Majid Rafiei and W.M.P. van der Aalst. TLKC-privacy model for process mining. In *Research Challenges in Information Science*, volume 385 of *LNBIP*, pages 398–416, 2020.

[34] Majid Rafiei and W.M.P. van der Aalst. Group-based privacy preservation techniques for process mining. In *Data & Knowledge Engineering*, volume 134, 2021.

[35] Majid Rafiei and W.M.P. van der Aalst. An abstraction-based approach for privacy-aware federated process mining. *IEEE Access*, 11:33697–33714, 2023. ISSN 2169-3536. doi: 10.1109/ACCESS.2023.3263673.

[36] Majid Rafiei, Frederik Wangelik, and W.M.P. van der Aalst. TraVaS: Differentially private trace variant selection for process mining. In *Process Mining Workshops*, volume 468 of *LNBIP*, pages 114–126. Springer, 2022.

[37] Majid Rafiei, Mahsa Pourbafrani, and W.M.P. van der Aalst. Federated conformance checking. *Information Systems*, 29, 2025.

[38] Christian Rennert, Julian Albers, Sander J. J. Leemans, and W.M.P. van der Aalst. Your secret is safe with me: Federated directly-follows graph discovery. In *7th International Conference on Process Mining (ICPM)*. IEEE, 2025.

[39] Javier Rojo, Jose Garcia-Alonso, Javier Berrocal, Juan Hernández, Juan Manuel Murillo, and Carlos Canal. SOWCompact: A federated process mining method for social workflows. *Information Sciences*, 595:18–37, 2022.

[40] Davide Sangiorgi. On the origins of bisimulation and coinduction. *ACM Trans. Program. Lang. Syst.*, 31(4):15:1–15:41, 2009.

[41] Anja F. Syring, Niek Tax, and Wil M. P. van der Aalst. Evaluating conformance measures in process mining using conformance propositions. *Trans. Petri Nets Other Model. Concurr.*, 14:192–221, 2019.

[42] Hannes Ueck, Robert Andrews, Moe T. Wynn, and Sander J. J. Leemans. Differentially private event logs with case attributes. In *Process Mining Workshops*, volume 533 of *LNBIP*, pages 240–252. Springer, 2024.

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

[44] W.M.P. van der Aalst. *Process Mining—Data Science in Action*. Springer Berlin Heidelberg, 2nd edition, 2016.

[45] W.M.P. van der Aalst. A practitioner's guide to process mining: Limitations of the directly-follows graph. *Procedia Computer Science*, 164:321–328, 2019.

[46] W.M.P. van der Aalst. Federated process mining: Exploiting event data across organizational boundaries. In *SMDS*, pages 1–7. IEEE, 2021.

[47] W.M.P. van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *Transactions on Knowledge and Data Engineering*, 16 (9):1128–1142, 2004.

[48] W.M.P. van der Aalst, Arya Adriansyah, Ana Karla Alves de Medeiros, Franco Arcieri, Thomas Baier, Tobias Blickle, Jagadeesh Chandra Bose, Peter van den Brand, Ronald Brandtjen, Joos Buijs, Andrea Burattin, Josep Carmona, Malu Castellanos, Jan Claes, Jonathan Cook, Nicola Costantini, Francisco Curbera, Ernesto Damiani, Massimiliano de Leoni, Pavlos Delias, Boudewijn F. van Dongen, Marlon Dumas, Schahram Dustdar, Dirk Fahland, Diogo R. Ferreira, Walid Gaaloul, Frank van Geffen, Sukriti Goel,

Journal Pre-proof

Christian Günther, Antonella Guzzo, Paul Harmon, Arthur ter Hofstede, John Hoogland, Jon Espen Ingvaldsen, Koki Kato, Rudolf Kuhn, Akhil Kumar, Marcello La Rosa, Fabrizio Maggi, Donato Malerba, Ronny S. Mans, Alberto Manuel, Martin McCreesh, Paola Mello, Jan Mendling, Marco Montali, Hamid R. Motahari-Nezhad, Michael zur Muehlen, Jorge Munoz-Gama, Luigi Pontieri, Joel Ribeiro, Anne Rozinat, Hugo Seguel Pérez, Ricardo Seguel Pérez, Marcos Sepúlveda, Jim Sinur, Pnina Soffer, Minseok Song, Alessandro Sperduti, Giovanni Stilo, Casper Stoel, Keith Swenson, Maurizio Talamo, Wei Tan, Chris Turner, Jan Vanthienen, George Varvaressos, Eric Verbeek, Marc Verdonk, Roberto Vigo, Jianmin Wang, Barbara Weber, Matthias Weidlich, Ton Weijters, Lijie Wen, Michael Westergaard, and Moe Wynn. Process mining manifesto. In *Business Process Management Workshops*, volume 99 of *LNCS*, pages 169–194. Springer, 2011.

[49] Alan F. Westin. Privacy and freedom. *Washington and Lee Law Review*, 1968.

[50] Zhiqiang Yan, Bo Sun, Yu Chen, Lijie Wen, Lei Hu, Jianmin Wang, Mingji Yang, and Lu Wang. Decomposed and parallel process discovery: A framework and application. *Future Gener. Comput. Syst.*, 98:392–405, 2019.

[51] Hootan Zhian. Federated stochastic process discovery: Definition, benefits, and challenges. In *BPM Doctoral Consortium*, 2025.

[52] Hootan Zhian, Rajkumar Buyya, and Artem Polyvyanyy. Federated stochastic process discovery using grammatical inference. In *Proceedings of the 37th International Conference on Advanced Information Systems Engineering*, volume 15702 of *LNCS*, pages 76–93. Springer, 2025.

[53] Hootan Zhian, Rajkumar Buyya, and Artem Polyvyanyy. Multi-objective metaheuristics for effective and efficient stochastic process discovery. In *Business Process Management*, volume 16044 of *LNCS*, pages 469–486, 2025.

[54] Hootan Zhian, Rajkumar Buyya, and Artem Polyvyanyy. Collaborative model optimization in federated stochastic process discovery. In *Business Process Management*, volume 16936 of *LNCS*, pages 611–629, 2026.

Journal Pre-proof

Declaration of Interest Statement

☒ The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

☐ The author is an Editorial Board Member/Editor-in-Chief/Associate Editor/Guest Editor for this journal and was not involved in the editorial review or the decision to publish this article.

☐ The authors declare the following financial interests/personal relationships which may be considered as potential competing interests:

Journal Pre-proof