Sharing Pearls, Not Pebbles: Privacy-Preserving Federated Stochastic Process Discovery via Partial Model Disclosure

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

Sunday 30th November, 2025

Abstract

Process discovery involves developing algorithms that use event logs to construct process models that describe business processes executed by information systems. Traditional discovery methods use centrally stored event logs. This approach raises privacy and communication concerns in distributed environments when multiple organizations execute the same process and want to discover a global model that represents it. We address these concerns by retaining local event logs within organizations and sharing the local models derived from them with the central server for merging into the global model. To further improve the privacy of this approach, we investigate whether sharing interesting parts of local models can yield high-quality global models. We formulate the extraction of model parts as a maximum-weight connected subgraph problem, enabling organizations to share model parts that describe fractions of their frequent behaviors. Evaluations we conducted on publicly available industrial logs demonstrate that our approach effectively constructs global process models comparable in quality to those obtained from centrally stored event logs, while providing enhanced privacy, reduced communication overhead, and faster runtimes.

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

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 process performance [35]. Process discovery, a core subarea of process mining, concerns the automatic construction of process models from an event log that faithfully describe the behavior of the underlying system [38]. To evaluate the quality of a discovered model

Preprint not peer reviewed
t research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

with respect to the event log, several criteria have been proposed [3]. Recall (also referred to as fitness) measures the extent to which the model can reproduce the behavior observed in the event log. Precision quantifies the extent to which the model avoids allowing behavior that does not appear in the event log. Generalization reflects the degree to which the model avoids overfitting by permitting plausible, unseen behavior. Finally, simplicity assesses the structural parsimony of the model, favoring models that avoid unnecessary constructs.

Discovered models are often represented as Directly-Follows Graphs (DFGs) [36]. A DFG is a directed graph with nodes representing actions, edges capturing the "can follow" ordering constraints between the actions, and numeric weights attached to nodes and edges specifying indicative frequencies of observing the corresponding actions and their adjacent executions in the event log.

The cross-organizational setting poses significant challenges for process discovery because organizations are often reluctant to share raw event data [15, 37]. Federated process discovery (FPD) has been proposed to address limitations of centralized discovery, including concerns about data privacy and security [15, 27], data transfer costs [15, 31], and scalability and efficiency issues [5, 31, 34].

Privacy generally encompasses the right of individuals and organizations to have control over how their data is collected, used, or disclosed to others [40]. Data regulations, such as GDPR, require organizations to protect personal information and consider privacy when processing data [22]. As process mining becomes more widely used across various fields, including healthcare, privacy concerns are growing. This is because process mining relies on event logs that often contain sensitive information, increasing the risk of exposing personal identities and confidential data.

van der Aalst et al. [39] discussed the role of data privacy in process mining and the importance of developing privacy-aware tools. Mannhardt et al. [19] identified privacy challenges in process mining with personalized data and proposed initial guidelines for privacy-preserving mining. Elkoumy et al. [9] highlighted key threats such as re-identification, where anonymized data can be linked back to individuals, and reconstruction, which allows the recreation of original logs from released models. Overall, privacy concerns focus on preventing individual identification from published event data. Elkoumy et al. [9] present three main privacy models for process mining: group-based models, differential privacy models, and encryption-based models. Group-based models have been addressed in several studies [25, 26, 28]. Encryption of the event log can be an initial solution to provide privacy in non-confidential environments before process mining [4]. Differential Privacy (DP) is another approach to preserving privacy by adding noise to sensitive attributes of event logs, making it difficult for attackers to infer private information about individuals [12, 13, 28].

In this paper, we propose a privacy-preserving approach in a federated setting that, unlike previous works, focuses on model-centric analysis. Instead of sharing the entire process model, each organization shares only a part of its model. This partial model emphasizes the most common and representative behaviors found in the local event log. By leaving out rare or exceptional traces, reconstructing the original event log or identifying specific cases becomes much harder. As a result, this selective sharing offers stronger privacy protection.

Figure 1 describes the architecture of the proposed approach, while Table 1 sum-

This research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

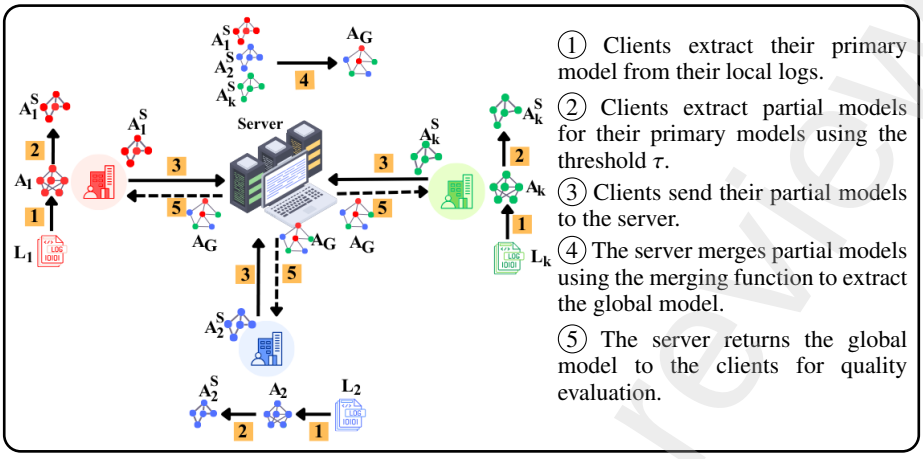


Figure 1: Privacy-preserving federated stochastic process discovery.

Table 1: Symmary of notations.

Symbol	Description
n	The total number of clients
L_i	The local event log of the i-th client
A_i	The local model of the i-th client
A_i^P	The sub-model extracted from A_i
A_i^u	The unfolded model of A_i
A_G	The global model
Server	The entity which aggregates $\{A_1^P, ..., A_n^P\}$ to generate A_G
Er	Entropic relevance, a quality measure
Er_i^g	The Er of the A_G according to L_i
Er_i	The Er of the A_i according to L_i
ϵ	The difference factor
τ	The threshold factor of the local sub-model

marizes the list of symbols used in the paper.

This article is an extended version of our conference paper [42]. The conference paper made the following contributions:

- It introduced a model-merging approach for deterministic frequency finite automata (DFFAs), although it can also be applied to other state-based process representations.
- It analyzed properties of merged DFFAs, including conditions for consistency, similarity, and morphisms between the merged model and the input models.
- It presented the *FedGASPD* and *FedCGASPD* algorithms as federated extensions of *GASPD*, enabling the discovery of DFFA models in distributed settings and the subsequent merging of the highest-quality local models into either a single global DFFA or a collection of DFFAs.
- It reported experiments on industrial event logs, demonstrating that *FedGASPD* and *FedCGASPD* discover models from distributed event data whose quality is comparable to the models produced by *GASPD* using the corresponding centralized logs.

t research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

This article extends the original conference paper in the following ways:

- It introduces the *P²FedGASPD* algorithm, which differs from prior approaches that rely on preprocessing or anonymizing event logs. Instead of sharing raw data or fully anonymized traces, our method preserves privacy by sharing only a partial abstraction of a primary model extracted from local logs. This model-centric strategy shifts the privacy focus from data to model representation and remains compatible with existing privacy-preserving techniques.
- It proposes a novel method for extracting a privacy-preserving partial DFFA model from a primary model by formulating the problem as a maximum-weight connected subgraph. This allows each participant to share only the most frequent and representative behavioral fragments, thereby enhancing privacy while maintaining model quality.
- It extends the experimental evaluation presented in the conference version, demonstrating that the global model produced by *P²FedGASPD* achieves quality comparable to that achieved by sharing complete local models, as in *FedGASPD*.

The remainder of the article proceeds as follows. Section 2 surveys related work on federated process discovery and privacy-preserving approaches within process mining. Then, Section 3 presents preliminaries necessary to understand the subsequent discussions. Section 4 presents the foundation of federated process discovery, detailing the algorithms for model merging, as well as the algorithms *FedGASPD* and *FedCGASPD*. Section 5 discusses *P²FedGASPD* for preserving privacy in federated process discovery. Section 6 reports on the evaluation of the proposed methods, based on open-source implementations and experiments conducted with real-world industrial event logs. Finally, Section 7 concludes the article, outlining directions for future research and summarizing our contributions.

2 Related Work

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

2.1 Federated Process Mining

Federated process mining has emerged as a promising approach. Aalst [37] first introduces the concept,In an effort to enable unified views of inter-organizational workflows by mapping organization event logs into a single federated event log. It discusses two main strategies: sharing filtered event data, and sharing higher-level abstractions such as DFGs to protect confidentiality. However, it leaves the actual merging of these abstractions as an open challenge. Rennert et al. [30] propose an approach for discovering DFGs from federated event logs without requiring handover-of-work or a trusted third party. It uses fully homomorphic encryption to securely compute a global DFG, ensuring that organizations do not share their event logs or leak sensitive information. Expanding on this, [31] 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 aggregating raw

Preprint not peer reviewed
research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

traces centrally to construct process models. While the latter approach involves aggregating traces from multiple devices to a central repository, it often incurs substantial data transfer costs. In contrast, our approach discovers models locally on each device and merges them into a final model, significantly reducing communication overhead. Khan et al. [15] 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 the parties which hold a part of the entire event log. Our approach can be embedded into similar protocols for federate process discovery. Rafiei and van der Aalst [27] propose an abstraction-based approach for privacy-preserving federated process discovery in inter-organizational scenarios, using directly-follows relations to represent event log abstractions. Their approach employs a handover mechanism to capture trace transition between organizations, enabling collaboration while limiting event log exposure. Their approach constructs event log abstractions from trace fragments visible to each organization, which are then aggregated into a unified abstraction representing the entire event log. By contrast, we partition the event log into groups of complete traces—for example, those managed by subsidiaries or business units within an organization—and derive abstractions for each group prior to merging them.

Decomposition and divide-and-conquer strategies are alternatives to centralized process discovery. Carmona et al. [5] 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 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 [34] introduces a generic divide-and-conquer framework that partitions actions, 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. [41] a five-step approach for decomposed and parallel process discovery to enable the efficient analysis of large event logs by breaking them into smaller part. They propose RPSTHD, which uniquely combines the Refined Process Structure Tree (RPST), heuristic mining, and Integer Linear Programming (ILP) to support multi-level decomposition. To decompose an event log, RPSTHD exploits dependency graph of activities derived from the event log into a hierarchical tree of Single-Entry-Single-Exit (SESE) fragments. Each node in this tree represents a SESE fragment, which corresponds to a subset of activities that can be treated as a subgraph. Different from the existing techniques, we work with multiple collections of complete traces rather than their projections, discover models from these collections, each capturing aspects of the overall process based on a data subset, and, finally, merging, rather than assembling, the intermediate models.

2.2 Privacy Preserving Process Mining

Mannhardt et al. [19] analyse the unique privacy challenges appearing with applying process mining to highly personalized, particularly with the advent of wearable sensor technologies. Moreover, they propose a framework and preliminary guidelines for privacy-aware process mining. Elkoumy et al. [9] address privacy concerns in process mining, highlighting key threats such as re-identification and reconstruction.

Preprint not peer reviewed
research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

Burattin et al. [4] introduce a basic methods for event log encryption. However, encryption does not completely eliminate risk of identifying individuals [28]. Michael et al. [21] introduce a novel 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. Fahrenkog-Petersen et al. [12] introduce prefix-based method in which the noise activity is added to prefix of comment trace variants as long as the perturbed prefix is in acceptable distance with the original one. This solution unlike [28] might produce new trace variants that are not in the event log.

Elkoumy et al. [10] introduce a differential privacy approach for event log anonymisation. 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. [33] introduce DP-ELCA, a framework for anonymising event logs with case attributes. It ensures differential privacy for both control flow and case attributes using machine learning-based synthetic data generation. It exploits TraVaS [28] to filters infrequent trace variants to prevent re-identification risks.

PRIPEL [13] applies differential privacy at the level of individual cases, rather than the entire event log. It first then applies differential privacy to alter their frequencies—sometimes removing real variants or introducing entirely new. In the next step, each anonymised 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 prevent direct linkage to the original event times.

Kabierski et al. [14] propose a framework for computing process performance indicators (PPIs) from business process event logs while ensuring individual privacy through differential privacy.

Rafiei and van der Aalst [24] Introduce a method to secure event logs by perturbing the frequency of activities to eliminate frequency-based attacks s to secure event logs. Elkoumy et al. [8] introduce Differentially-Private DFG by adding noise to the weight of edges in the original DFG model.

Lim et al. [16] propose a novel 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 anonymised separately using two differential privacy mechanism; SaCoFa[12] and Laplace mechanism [20].

Rafiei and van der Aalst [25] proposes TLKC-privacy approach in which a greedy algorithm designed to prevent attribute linkage attacks by suppressing infrequent events from traces to ensure privacy through a k-anonymity guarantee. 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 [26] extend the TLKC-privacy approach which supports main process mining perspectives (control-flow, organizational, case, and time), and introduces improved event filtering score to cover different perspectives.

Preprint not peer reviewed
research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

Rafiei and van der Aalst [26] propose a k-anonymity mechanism to anonymise event logs. K-anonymity is to preserve privacy by making sure that each individual's data cannot be distinguished from at least k-1 others. This means that even if someone gains access to the event log, they cannot easily Identify any one person, as each record is hidden within a group of similar records. Rafiei et al. [28] propose TraVaS that exploits partition selection to determine which trace variants to be published. It perturbs the frequency of each trace variant using the k-Truncated Symmetric Geometric Distribution (k-TSGD) mechanism, and finally publishes only those variants whose noisy frequencies higher than a threshold, removing infrequent traces.

Rafiei et al. [29] propose a federated approach in which organisations only share public version of their event log that include interaction events but exclude internal activities.

3 Preliminaries

This section presents concepts necessary to understand the contributions discussed in the subsequent sections. A Deterministic Frequency Finite Automaton (DFFA) is a mathematical model that describes traces and their frequencies.

Definition 3.1 (Deterministic Frequency Finite Automata)
A *Deterministic Frequency Finite Automaton* (DFFA) is a tuple $(Q, \Lambda, q_0, \mathbb{I}, \mathbb{F}, \delta)$, where:

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

⌟

The notation $\delta(q, a, q') = n$ indicates that there is a transition from state q to state q' labeled with a, occurring n times. 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 $\delta(p_3, a, p_4) = 5$. A DFFA is consistent if for each of its states the sum of frequencies of entering and leaving (or terminating at) that state is identical.

Definition 3.2 (Consistency [7])
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 Fig. 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.

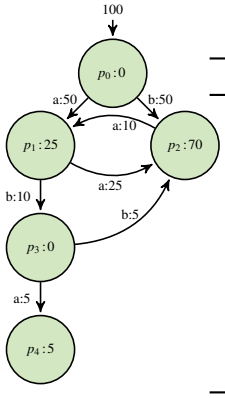
Preprint not peer reviewed
t research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

Definition 3.3 (Similarity [32])
DFFA $A_1 = \left(Q^1, \Lambda^1, q_0^1, \mathbb{I}^1, \mathbb{F}^1, \delta^1\right)$ *simulates* DFFA $A_2 = \left(Q^2, \Lambda, q_0^2, \mathbb{I}^2, \mathbb{F}^2, \delta^2\right)$, 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 $(p, q) \in \mathbb{S}$, if there exists $a \in \Lambda^1$ and $p' \in Q^1$ such that $\delta^1(p, a, p') > 0$, then it holds that there exists $q' \in Q^2$ such that $\delta^2(q, a, q') > 0$ and $(p', q') \in \mathbb{S}$. ⌟

Our approach to federated process discovery is based on *GASPD*, an evolutionary stochastic process discovery algorithm grounded in grammatical inference [2]. *GASPD* uses *ALERGIA* to extract accurate stochastic language models of traces recorded in an input event log. *ALERGIA* operates by first constructing the prefix automaton of the event log and then iteratively reduces it through recursive merging of states that spawn similar sequences of actions into a consistent DFFA Carrasco and Oncina [6].

Algorithm 1 summarizes *GASPD* in pseudocode.



Algorithm 1: *GASPD* [2]

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

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

Figure 2: DFFA A_1

GASPD uses a multi-objective genetic search to discover interesting DFGs from the input event log. It begins by generating a population P of p parameter triplets (Line 2). Each triplet defines a DFG, with one parameter determining which infrequent traces to exclude from the event log and two parameters guiding *ALERGIA* in constructing a DFFA from the remaining frequent traces. Using its native translation mechanism, *GASPD* converts these DFFAs into sound DFGs. In the selection phase (Line 4), the most "fit" population members F are chosen. Fitness is defined by Pareto optimality concerning model size (smaller models are preferred) and entropic relevance of discovered models (lower relevance values are preferred). 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]. Next, in the crossover phase (Line 5), offspring U are generated by applying crossover and mutation operations to k randomly selected parents from F. These operations aim to ensure that offspring inherit the best features of prior generations. Finally, in the replacement phase (Line 6), *GASPD* updates the population by preserving Pareto-optimal members in F and incorporating new, superior members from U.

This research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

4 Federated Process Discovery

This section presents the problem of federated (stochastic) process discovery, a technique for merging DFFAs, the *FedGASPD* algorithm that discovers one DFFA from each input event log and then merges them into the final discovered process model, and the *FedCGASPD* algorithm that refines *FedGASPD* to merge similar models to obtain a range of models of different characteristics.

4.1 Definition

Let X be a set. By $\mathcal{B}(X)$, we denote the set of all finite multisets over X, while, by X^*, we denote the set of all finite sequences over X. 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 $f : \mathcal{L}^* \to \mathcal{M}$.

Let $q : \mathcal{L} \times \mathcal{M} \to [0, 1]$ be a function that measures *quality* $q(L, M)$, $L \in \mathcal{L}$, $M \in \mathcal{M}$ of model M discovered from event log L. For instance, measure q can be precision, recall, simplicity, or generalization [3]. Such a measure usually associates larger values with models that describe the system better. A good federated process discovery approach f should construct a model from multiple event logs that is comparable in quality to the model constructed from the corresponding centralized event log:

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

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 the obtained models into the final process model. Let $m : \mathcal{M} \times \mathcal{M} \to \mathcal{M}$ be a function that merges models. 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 such model can be constructed on a dedicated device at the organization that owns the data and is an abstract, and thus more confidential [37], representation of the corresponding event log. We then merge the models incrementally using function $F_m : \mathcal{M}^* \to \mathcal{M}$:

$$F_m(\langle M_1 \rangle) = M_1 \text{ and } F_m(\langle M_1, M_2, \ldots, M_n \rangle) = F(\langle m(M_1, M_2), \ldots, M_n \rangle)\,.$$

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

Preprint not peer reviewed

t research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

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 $d \in \mathbb{N}$. The first step is to construct the (prefix of the) unfolding of A_2 truncated at the $d + 1$ occurrences of its states [11]. In this prefix automaton, state and transition frequencies are derived from those in A_2 by propagating frequencies along the unfolding branches, using probabilities estimated based on 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 any remaining parts of the prefix. During this embedding process, the frequencies from A_1 and the constructed prefix of A_2 are aggregated. We denote the result of merging A_2 into A_1 using parameter d as $A_1 \lhd_d A_2$.

4.2.1 Unfolding

Algorithm 2 details the unfolding algorithm, which extracts the prefix of a model before merging it with another. This algorithm employs Depth-First Search (DFS) to traverse the model. The process begins by adding the initial state of the model Line 1 to the set of states (Q^u). Next, the incoming frequency of (q_0) is added to $\mathbb{I}^u(q_0)$ Line 9. (R) is defined as a list Line 5 to store each state visited by the algorithm. Since the model may contain cycles, each state can be visited multiple times. Therefore, the algorithm tracks the number of times each state is visited Line 10. Additionally, it appends a visit counter to each state's name, introducing a fresh state to ensure the unfolding forms a tree. To manage this, a map structure (*alias*) is used, which maps the concatenation of the state name and its visit counter to the original state name Lines 28 to 32. To prevent the algorithm from entering an infinite loop, if a traversed path to a state contains a pattern that appears (d) times, the expansion is halted Line 26. For calculating frequencies, the algorithm uses a structure *inFreQ*, which stores the total incoming frequency to each state in the unfolded model. When a transition is unfolded, its frequency is calculated as the product of the transition's frequency in the original model and the incoming frequency of the current unfolded state, divided by the total outgoing frequency (*outFreQ*) of the original state Line 25, Line 36 and Line 37. The final frequency for each unfolded state is calculated in a similar way Line 39. Figure 4 shows the prefix of the unfolding of the DFFA A_2 from Fig. 3 computed for $d = 2$.

4.2.2 Embedding

Algorithm 3 provides details about embedding a model to another one. First, it adds the initial state of the first model A_1 as described in Line 1, since it is the primary model. The sum of the initial frequencies of both models is added to the initial frequency of q_0^1 in the merged model (Line 2). Then, the union of the action sets from both models is added to the action set of (A_G).

This algorithm, similar to the *unfolding* technique, uses DFS to traverse both models. It employs a queue R to store pairs of states (p, q), where p is from the first model and q is from the second model (Line 27). It also defines a list called *explored* to store pairs of states (p, q) to avoid infinite loop (Line 10).

t research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

Algorithm 2: DFFA unfolding method

Input: DFFAs $A = (Q, \Lambda, q_0, \mathbb{I}, \mathbb{F}, \delta)$
Output: DFFA A^u resulting from unfolding A

$Q^u \leftarrow \{q_0\}$;
$\mathbb{I}^u \leftarrow \{(q_0, \mathbb{I}(q_0)\}$;
$\Lambda^u \leftarrow \Lambda$;
$\delta^u \leftarrow \emptyset$;
$R \leftarrow$ new empty queue;
$pathRec \leftarrow (q_0, q_0)$;
$R.push(q_0)$;
$alias \leftarrow \{(q_0, q_0)\}$;
$inFreQ \leftarrow \{(q_0, \mathbb{I}(q_0))\}$;
$statevisitCnt \leftarrow \{(q_0, 1)\}$;
$outFreQ \leftarrow \emptyset$;
for $q \in Q$ **do**
    $totalFreQ \leftarrow 0$;
    for $a \in \Lambda$ **do**
        if $\exists q' \in Q : ((q, a, q'), n) \in \delta, n \in \mathbb{N}^+$ **then**
            $totalFreQ \in totalFreQ + n$;
    $totalFreQ \in totalFreQ + \mathbb{F}(q)$;
    $outFreQ \leftarrow outFreQ \cup (q, totalFreQ)$;
while R is not empty **do**
    $q_{alias} \leftarrow R.pop()$;
    $q \leftarrow alias(q_{alias})$;
    $inFreQ \leftarrow ingFreQ(q_{alias})$;
    for $a \in \Lambda^u$ **do**
        if $\exists q' \in Q : ((q, a, q'), n) \in \delta, n \in \mathbb{N}^+$ **then**
            $trnFreq \leftarrow n \times inFreQ(q_{alias}) \div outFreQ(q)$;
            $trvPath \leftarrow pathRec(q_{alias})$;
            if $detectloop(CONCAT(trvPath, q')) < d$ ***and*** $trnFreq > 0$ **then**
                $Cnt \leftarrow statevisitCnt(q')$;
                $statevisitCnt \leftarrow (statevisitCnt \setminus \{(q', Cnt)\}$;
                $Cnt \leftarrow Cnt + 1$;
                $statevisitCnt \leftarrow statevisitCnt \cup \{(q', Cnt)\}$;
                $q'_{alias} \leftarrow CONCAT(q', Cnt)$;
                $Q^u \leftarrow Q^u \cup q'_{alias}$;
                $alias \leftarrow alias \cup \{(q'_{alias}, q')\}$;
                $path \leftarrow path \cup \{(q'_{alias}, CONCAT(trvPath, q')\}$;
                $\delta \leftarrow \delta \cup \{((q_{alias}, a, q'_{alias}), TrnFreq)\}$;
                $inFreQ \leftarrow inFreQ \cup \{(q'_{alias}, TrnFreq)\}$;
                $R.push(q'_{alias})$;
    $\mathbb{F}^u(q_{alias}) \leftarrow \mathbb{F}^u \cup \{(q_{alias}, inFreQ(q_{alias}) \times \mathbb{F}(q) \div outFreQ(q))\}$;
return $A \leftarrow (Q^u, \Lambda^u, q_0, \mathbb{I}, \mathbb{F}^u, \delta^u)$;

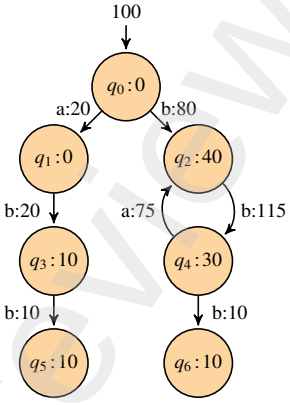


Figure 3: DFFA A_2

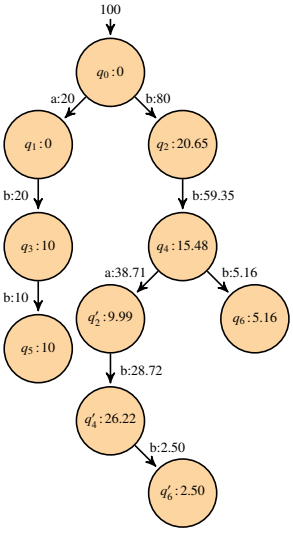


Figure 4: Prefix of DFFA A_2^u ($d = 2$)

Inside the while loop, the algorithm retrieves the pair (p, q) from the queue and adds it to *explored*. Then, based on the current pair (p, q), it explores the next possible transitions in both models. If for an action, there is a transition in both models, the next state p' is added as a successor state of the transition, and the total frequencies of

t research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

the transitions from p to p' and from q to q' are summed and assigned as the transition frequency in the merged model (Line 17, Line 16, Line 21). If, for an action, there is only a transition in the second model Line 19, then q' is added as a successor of state p in the merged model, and its transition frequency is assigned to the transition between (p, q') in the merged model. The pair (p', q') is then added to the queue R if the algorithm has not visited it before (Lines 25 to 27). *MergeDFFA* terminates after it explores the necessary pairs of states from the two input automata starting from the pair of the initial states.

The procedure always terminates because the number of all possible such state pairs is finite. The runtime of the merging process is primarily determined by the size of the prefix of the unfolding of DFFA A_2, which, in the worst case, can be exponential in the size of A_2. When the parameter d is set to a small value, the size of the prefix unfolding is typically only a few times larger than that of A_2. However, increasing d allows the behavior of A_2 to be captured more accurately in the resulting merged automaton.

DFFA A_G in Fig. 5 is the result of embedding A_2 from Fig. 3 into A_1 from Fig. 2 using $d = 2$, that is, $A_G = A \triangleleft_1 A_2$. The khaki-colored states indicate states from A_1 that were reused during the embedding of the prefix of A_2 from Fig. 4 into A_1. The green states correspond to the original states of A_1 not visited during the embedding of the prefix, while the orange states are the states of the prefix that do not "fit" into A_1.

The automaton that results from merging two consistent automata is consistent.

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

Lemma 4.1 holds because the prefix automaton that *MergeDFFA* embeds into A_1 is, by construction, consistent. Furthermore, each time a state from the prefix of the unfolding is embedded into the resulting automaton, the *MergeDFFA* procedure updates both the termination frequency of the merged state and the frequencies of its outgoing transitions to preserve consistency.

The automaton produced by *MergeDFFA* is in the following simulation relations with the input automata.

Lemma 4.2 (Similarity)
If A_1 and A_2 are consistent DFFAs and $A = A_1 \triangleleft_d A_2$, $d \in \mathbb{N}$, 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, $A_2 \preceq A$.* ⌟

Lemma 4.2 holds based on the following observations. Each automaton produced by *MergeDFFA* is constructed by extending the structure of DFFA A_1, thus preserving all of its transitions. Also, *MergeDFFA* ensures that all transitions from the prefix of the unfolding of A_2 are retained in the resulting DFFA. If A_2 is acyclic, the unfolding of A_2 simulates A_2. Finally, the unfolding of an acyclic automaton is isomorphic to the prefix of the unfolding of this automaton for any truncation depth $d > 0$.

Notably, given DFFAs A_1 and A_2 and $d \in \mathbb{N}$, it is not always the case that $A_1 \triangleleft_d A_2 = A_2 \triangleleft_d A_1$; that is, the merging operation is not commutative. This fact suggests that if *MergeDFFA* is used to merge models when solving the federated process discovery problem, different orders of mergings will indeed lead to different discovered models.

Preprint not peer reviewed
t research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

Algorithm 3: DFFA integration

Input: DFFAs $A_1 = \left(Q^1, \Lambda^1, q_0^1, \mathbb{I}^1, \mathbb{F}^1, \delta^1\right)$ and $A_2 = \left(Q^2, \Lambda^2, q_0^2, \mathbb{I}^2, \mathbb{F}^2, \delta^2\right)$

Output: DFFA A resulting from integrating A_2 into A_1

$Q \leftarrow \left\{q_0^1\right\}$;
$\mathbb{I} \leftarrow \{(q_0^1, \mathbb{I}^1(q_0^1) + \mathbb{I}^2(q_0^2))\}$;
$\Lambda \leftarrow \Lambda^1 \cup \Lambda^2$;
$\delta \leftarrow \emptyset$;
$R \leftarrow$ new empty queue;
$R.push((q_0^1, q_0^2))$;
$explored \leftarrow \emptyset$;
while R is not empty **do**
  $(p, q) \leftarrow R.pop()$;
  $explored \leftarrow explored \cup \{(p, q)\}$;
  for $a \in \Lambda$ **do**
    $p' \leftarrow$ NULL;
    $q' \leftarrow$ NULL;
    if $\exists\, p' \in Q^1 : ((p, a, p'), n) \in \delta^1, n \in \mathbb{N}^+$ **then**
      if $\delta(p, a, p') = 0$ **then**
        $\delta \leftarrow \delta \cup \{((p, a, p'), n)\}$;
      $Q \leftarrow Q \cup \{p'\}$;
      $\mathbb{I} \leftarrow \mathbb{I} \cup \{(p', 0)\}$;
    if $\exists\, q' \in Q^2 : ((q, a, q'), n) \in \delta^2, n \in \mathbb{N}^+$ **then**
      if $p' =$ NULL **then**
        $\delta \leftarrow \delta \cup \{((p, a, q'), n)\}$;
        $Q \leftarrow Q \cup \{q'\}\ \mathbb{I} \leftarrow \mathbb{I} \cup \{(q', 0)\}$;
      else
        $\delta \leftarrow (\delta \setminus \{((p, a, p'), n)\}) \cup \{((p, a, p'), n) + \delta^2(q, a, q'))\}$;
    if $p' \neq$ NULL $\vee\ q' \neq$ NULL **then**
      if $(p', q') \notin explored$ **then**
        $R.push(p', q')$
return $A \leftarrow (Q, \Lambda, q_0, \mathbb{I}, \mathbb{F}, \delta)$;

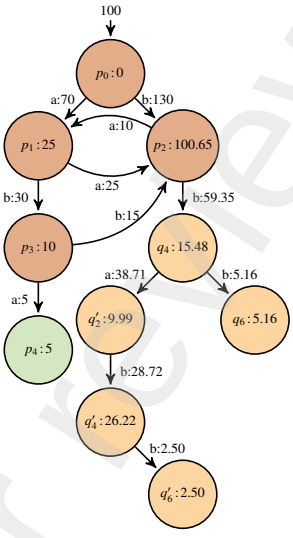


Figure 5: DFFA A_G

4.3 Discovering Single Model

The *FedGASPD* as a federated process discovery exploits a set of parameters to configure the *GASPD* algorithm, along with multiple event logs. These logs originate from the same process, but may be recorded by different organizations or various departments within a single organization. The algorithm works by creating an individual model for each event log and then combining these models to generate a global process model; see Algorithm 4 for more details.

The process of obtaining a model from an input event log begins by applying the *GASPD* algorithm (Algorithm 1). It returns list models that are Pareto-optimal, based on the size and entropic relevance. From this collection of Pareto-optimal models, we select one using the *SELECTDFFA* function. The chosen model is subsequently added to our list of constructed DFFAs, M, by concatenating it with the existing sequence

research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

Algorithm 4: *FedGASPD*

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

Output: A DFFA discovered from L

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

using the operator "$\circ$" (Line 3). Given that the procedure for generating a model can be carried out independently for each input event log, we process all of them simultaneously (Line 2). Different versions of the *SELECTDFFA* function allow for the implementation of various model selection policies. Once all models have been selected, they are merged pairwise through the *MergeDFFA* procedure (Line 4); cf. Section 4.2 for further details on the merging process.

The discovery of Pareto-optimal models and the selection of one optimal model for each input event log can be performed locally on devices at the respective organizations where the input event logs are stored. Once the models are selected, they can be transmitted to a central server, where the merging process takes place.

Note that *GASPD* identifies parameter triplets that trigger *ALERGIA* to construct automata that can be translated into interesting DFGs. The *SELECTDFFA* function, thus, selects one parameter triplet identified by *GASPD* and constructs the corresponding DFFA using *ALERGIA*. After merging all DFFAs into a final model, this model can be trivially translated into a probabilistic automaton using Algorithm 16.1 by de la Higuera [7], which, in turn, can be trivially converted into a DFG model following the procedure outlined in Definition 4.2 by Alkhammash et al. [2]. To ensure brevity of presentation and because the quality of DFFAs directly impacts the quality of the resulting DFGs, here, we design the core techniques to operate with DFFAs.

4.4 Discovering Multiple Models

FedGASPD solves federated process discovery in a straightforward manner: it discovers one model from each input event log and merges them. This approach gives a basic baseline for federated process discovery. However, when the discovered models exhibit diverse characteristics, merging them may result in a loss of distinctiveness and dilute their strong individual features due to the averaging effect. To address this limitation, we propose a variation of *FedGASPD* that avoids merging all models into one. Instead, it groups similar models into clusters and merges the models within each cluster, yielding multiple models that retain similar, strong characteristics of the merged models. We refer to this variant as *FedCGASPD*, which is summarized in Algorithm 5.

After discovering models from input event logs (Lines 2 and 3 in Algorithm 5), they are clustered using the k-means algorithm [18] via the *KMeansCluster* function (Line 4). The *KMeansCluster* function takes as input the list of models, M, and the

This research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

Algorithm 5: *FedCGASPD*

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

Output: A list of DFFAs discovered from L

1 $M \leftarrow \langle\rangle$; $\mathbb{M} \leftarrow \langle\rangle$;
2 **parallelfor** $i \in [1 \,..\, n]$ **do**
3 $\quad M \leftarrow M \circ \langle SELECTDFFA(GASPD(p, m, k, L_i))\rangle$;
4 $C \leftarrow KMeansCluster(M, n)$;
5 **for** $i \in [1 \,..\, n]$ **do**
6 $\quad$ **if** $|C(i)| \geq 0$ **then** $\mathbb{M} \leftarrow \mathbb{M} \circ \langle F_{MergeDFFA}(C(i))\rangle$;
7 **return** $\mathbb{M}$;

desired number of clusters, n, and outputs a list of clusters, C, where each cluster is stored as a list of models. We specify n as the number of desired clusters and initialize n random centroids (initial cluster centers), each represented as a point in the space $[0, 1] \times [0, 1]$. If there are fewer than n meaningful clusters, the k-means algorithm identifies these clusters by iteratively refining the centroids while allowing some centroids to remain unused. 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. Then, the *MergeDFFA* procedure is applied to each non-empty cluster. All models within a cluster are merged, and the resulting merged model is appended to the list of discovered models (Line 6).

5 Privacy in Federated Process Discovery

As discussed in the previous section, the merging approach enables organizations to collaboratively build a global process model without sharing their raw event logs. This method allows the global model to reflect the combined behavior of all local event logs, while reducing the exposure of sensitive information. However, even when only models are shared, privacy risks remain. Sharing a complete local model can still reveal rare or unique behaviors that may lead to reconstruction of traces or reidentification of individuals or sensitive business details. To mitigate this risk, our solution introduces selective model sharing. Instead of sharing the entire local model produced by GASPD, each client systematically extracts and shares only the most frequent and representative behaviors from its model.

5.1 Partial DFFA Model Extraction

A partial DFFA model is extracted from each client's primary model by systematically removing low-frequency behaviors. This approach enables sharing only the most representative and frequent behaviors, enhancing privacy by making it more difficult to

This research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

infer rare, potentially identifying traces. In the following subsections, we first formally define the partial DFFA model, then present formulation of the Maximum Frequency Partial DFFA Model Problem, and finally describe our method for rebalancing the frequencies within the extracted partial DFFA model to ensure consistency property.

Definition 5.1 (Partial DFFA model)
A Partial Deterministic Frequency Finite Automaton (PDFFA) is a DFFA Model extracted from a DFFA. Given a DFFA $A = (Q, \Lambda, q_0, \mathbb{I}, \mathbb{F}, \delta)$ and threshold τ, find a partial model DFFA $A^P = (Q^s, \Lambda^s, q_0, \mathbb{I}^s, \mathbb{F}^s, \delta^s)$ where:

- $Q^s \subseteq Q$;
- $\Lambda^s \subseteq \Lambda$;
- $\mathbb{I}^s \subseteq \mathbb{I}, \mathbb{I}^s(q_0) = \mathbb{I}(q_0)$;
- $\mathbb{F}^s : Q^s \to \mathbb{N}$;
- $\delta^s : Q^s \times \Lambda^s \times Q^s \to \mathbb{N}$; and
- Each state must be reachable from $q_0 : \forall q \in Q^s, \exists\, n > 0, \exists\, q_0, q_1, ..., q_n : \forall\, 0 \leq i \leq n : \exists a \in \Lambda^s, \delta^s(q_i, a, q_{i+1}) > 0 \cap \exists\, a \in \Lambda^s : \delta^s(q_n, a, q) > 0$. ⌟

Problem 1 (Maximum Frequency Partial DFFA Model Problem) *(MFPDFFA)]*
The objective is to maximize the total final and transition frequencies of the selected states and transition edges in the partial DFFA model. MFPDFFA is a generalized variant of the Maximum-Weight Connected Subgraph (MWCS) problem, which has been proven to be NP-hard [17]. In MWCS, a subgraph is selected such that the total weight of the selected vertices in an undirected graph is maximized. In contrast, MFPDFFA involves both nodes and edges being weighted, and the graph is directed. Since MWCS is a special case of our problem, MFPDFFA is also NP-hard. we formulated MFPDFFA as a Mixed Integer Linear Programming (MILP) problem. To begin, the decision variables x_q and $y_{q,a,q'}$ (Eqs. (C1) and (C2))are binary (0 or 1), indicating whether vertices and edges are selected or not. The partial model must include $q0$ (Eq. (C4)). Eq. (C3) enforces that the total number of states and transition edges does not violate the upper bound τ.

$$\text{s.t.} \quad x_q \in \{0,1\} \qquad \forall q \in Q \tag{C1}$$

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

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

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

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

To enforce connectivity from the initial state q_0, we use a flow-based model[1], which introduces flow variables $f_{q,a,q'}$ for each (q, a, q') such that $\delta(q, a, q') > 0$, representing the amount of flow sent along transition edge (q, a, q') to in the partial model to enforce connectivity Eqs. (C5) to (C7). The flow-based model ensures that the selected partial

t research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

model is connected which means states are reachable from the initial state q_0.

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

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

The objective function is a linear sum of transition and final frequencies weighted by these binary variables(Eq. (OF)).

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

To solve this MILP, we employ the com.google.ortools solver, a state-of-the-art optimization library that provides efficient algorithms for solving large-scale integer and linear programming problems.

5.2 Frequency Rebalancing of Partial DFFA Model

After extracting *∂model* from A, the frequencies must be re-balanced to maintain the consistency feature.
Let $\mathcal{P}_q$ denote the final probability of state q, and $\mathcal{P}_{q,a,q'}$ denote the probability of transition from state q to state q' on action a. These probabilities are calculated as follow:

$$\forall q \in Q' : \mathbb{P}_q = \frac{\mathbb{F}(q)}{\sum_{q'\in Q,\, a\in\Lambda} \delta(q,a,q') + \mathbb{F}(q)} \tag{1}$$

$$\forall q, q' \in Q, a \in \Lambda : \mathbb{P}_{q,a,q'} = \frac{\delta(q,a,q')}{\sum_{q''\in Q,\, b\in\Lambda} \delta(q,b,q'') + \mathbb{F}(q)} \tag{2}$$

This normalization ensures that:

$$\forall q \in Q : \sum_{q'\in Q, a\in\Lambda} \mathbb{P}_{q,a,q'} + \mathbb{P}_q = 1 \tag{3}$$

Probabilities in A cannot be directly used in the extracted A^P because some states and transitions have been removed from A. In particular, the condition given in Eq. (3) may no longer hold in A^P. To update the probabilities accordingly, the following equations are used:

$$\forall q, q' \in Q^s,\ a \in \Lambda^s, \mathbb{P}^s_{q,a,q'} = \begin{cases} \mathbb{P}_{q,a,q'} & \delta^s(q,a,q') > 0 \\ 0 & \delta^s(q,a,q') = 0 \end{cases} \tag{4}$$

$$\forall q \in Q^s : \quad \mathbb{P}^s_q = 1 - \sum_{q'\in Q^s, a\in\Lambda^s} \mathbb{P}^s_{q,a,q'} \tag{5}$$

With Eq. (5), the probabilities of removed transitions are added to the final probability of the source state of those transitions. The definition of consistency (Definition 3.2)

t research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

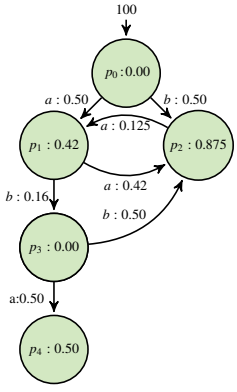


Figure 6: DFFA A_1 after converting frequencies to probability distributions

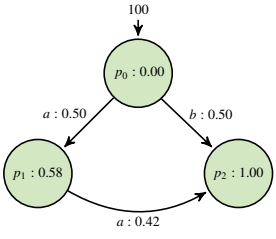


Figure 7: Normalised DFFA A_1^P

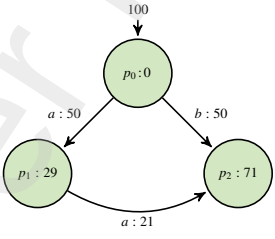


Figure 8: Re-balanced DFFA A_1^P

and Eq. (5) lead to a system of linear equations, which includes one equation for each state and each transition, respectively (Eqs. (6) and (7)). According to Eq. (8), the sum of all final frequencies is equal to the sum of all initial frequencies.

$$\forall q \in Q^s : \sum_{q' \in Q^s, a \in \Lambda^s} \delta^s(q', a, q) + \mathbb{I}^s(q) = \sum_{q' \in Q^s, a \in \Lambda^s} \delta^s(q, a, q') + \mathbb{F}^s(q) \tag{6}$$

$$\forall q, q' \in Q^s, a \in \Lambda^s : \delta^s(q, a, q') = \mathbb{P}^s_{q,a,q'} \times (\sum_{q'' \in Q^s,\, a \in \Lambda^s} \delta^s_{q'',a,q} + \mathbb{I}^s(q)) \tag{7}$$

$$\sum_{q \in Q^s} \mathbb{I}^s(q) = \sum_{q \in Q^s} \mathbb{F}^s(q) \tag{8}$$

The system of equations above forms a linear programming problem that can be solved using standard optimization techniques. The solution provides re-balanced transition and final frequency for the A^P.

5.3 Privacy Preserving Federated Stochastic Process Discovery

The detailed *P²FedGASPD* algorithm is presented in Algorithms Algorithms 6 and 7. *P²FedGASPD* extracts the global model A_G in two phases: the server phase and the client phase.

Server phase:

t research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

The server is responsible for merging the partial models sent by the clients. First, the server initializes its parameters (Lines 1 and 2). Then, it iteratively calls the *Update-Client* function for each client (Line 4) and sends the updated A_G and ϵ. In this step, the server requests all clients to send their updated partial models (Lines 4 and 5). After gathering all partial models, the server merges them using Algorithm 3 (Line 6). The loop of extracting and merging partial models continues until the server does not receive any updated partial models.

Client phase:

Each client starts by initializing its parameters τ, the primary model A, the partial model A^P, and sets its status to *Initialisation* (Lines 1 to 4). When *UpdateClient* function is called by the server for the first time, the client checks its status. If it is in the *Initialisation* state, it extracts the primary model A using Algorithm 1 and changes its status to *Start* (Lines 14 and 15). On subsequent calls, the client checks its status, and if it is not *Finish*, it first extracts and updates the A^P by calling *RetrivePartial-Model* (Lines 16 and 18). Then, the client compares $Er_{(A_g,L)}$ with $Er_{(A,L)}$ to determine if the quality of the global model is comparable to the primary local model A (Line 19). If so, the client changes its status to *Finish* (Line 20), meaning it will no longer update its partial model A^P. In this case, the client concludes that the current A^P sent to the server is sufficient to represent its local event log L. Otherwise, the client extracts a new A^P with more information by incrementally increasing τ (from 1 up to the size of the primary model A).

Algorithm 6: $P^2FedGASPD$ Server

Input: a non-empty list of clients $C = \langle C_1, \ldots, C_n \rangle$
Output: A_G

1 $M \leftarrow \emptyset$
2 $A_G \leftarrow$ NIL
3 **while** *there is an updated model* **do**
4 **parallelfor** $i \in [1 \,..\, n]$ **do**
5 $M \leftarrow M \circ C_i - > UpdateClient(A_G, \epsilon)$
6 $A_G \leftarrow F_{MergeDFFA}(M)$
7 **return** A_G;

Preprint not peer reviewed
t research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

Algorithm 7: P^2*FedGASPD* Client

Input: p, m, k, L

$\tau \leftarrow 1$
$A \leftarrow$ NIL
$A^P \leftarrow$ NIL
$state \leftarrow Initialisation$
RetrivePrimaryModel(p, m, k, L)**:**
$A \leftarrow SELECTDFFA(GASPD(p, m, k, L))$
$state \leftarrow Start$
return A
RetrivePartial-Model(A, τ)**:**
$A^P \leftarrow MFPDFFA(A, \tau)$
return A^P;
UpdateClient(A_G, ϵ)**:**
if $state = Initialisation$ **then**
    $A \leftarrow RetrivePrimaryModel(p, m, k, L)$
    $state \leftarrow Start$
else if $state \neq Finish$ **then**
    $A^P \leftarrow RetrivePartial\text{-}Model(A, \tau)$
    $\tau \leftarrow \tau + 1$
    if $Er_{(A,L)} - Er_{(A_G,L)} < \epsilon$ **then**
        $state \leftarrow Finish$
return A^P;

6 Evaluation

This section presents the evaluation of our federated process discovery approach. The evaluation focuses on addressing the following questions:

- Do federated process discovery methods that rely on complete model sharing (*FedGASPD* and *FedCGASPD*) produce models of comparable quality to those obtained through centralized discovery (*GASPD*)?
- Does federated process discovery based on partial model sharing (P^2*FedGASPD*) achieve model quality comparable to federated approaches that share complete models (*FedGASPD* and *FedCGASPD*)?

We next describe the implementation of our approach and the datasets used to evaluate it, and present the experimental results that answer these questions. 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 (15.7GB of effective memory).

Preprint not peer reviewed
t research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

6.1 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 2 summarizes the characteristics of the nine event logs used in the evaluation.

Table 2: Characteristics of the event logs used in our experiments.

Event log	#Actions	#Traces	#Distinct traces	Max. trace length	Avg. trace length
BPI-2012	24	13,087	4,366	175	20.03
BPI-2013	4	7,554	1,511	123	8.67
BPI-2017	26	31,509	15,930	180	38.15
BPI-2018	34	2,861	2,498	680	61.58
BPI-2019	42	251,734	11,973	990	06.33
BPI-2020-1	34	6,449	753	27	11.18
BPI-2020-2	19	6,886	89	20	05.34
Sepsis Cases	16	1,050	846	185	14.48
Hospital Billing	18	100,00	1,020	217	4.51

6.2 Complete Model Sharing

For each event log, *GASPD*, *FedGASPD*, and *FedCGASPD* were initialized the initial population size, p, and the number of generations, m, to 50. To construct human-readable models, we controlled the parameter responsible for the exclusion of infrequent traces. Consequently, we constructed models of at most 1,000 nodes and arcs, with many models substantially smaller than this target size.

To simulate cross-organizational environments, each event log was split randomly among a requested number of computational nodes, ensuring that each node has approximately the same portion of the event log traces. We implemented the *SELECTDFFA* function from Algorithms 4 and 5, by selecting one DFFA from a Pareto-frontier of DFFAs that minimizes this objective function: $(1 - size^*) + (1 - relevance^*)/2$, where $size^*$ and $relevance^*$ are the normalized size and entropic relevance of the model, with normalization performed as described in Section 4.4. This way, we select a model that balances the two competing objectives equally.

Table 3 presents the execution times of *GASPD* and its federated alternatives. Note that *FedGASPD* and *FedCGASPD* are substantially faster than *GASPD*. For example, for the nasa-cev event log, the federated discovery techniques are up to 26 times faster than *GASPD* and, at a minimum, 5 times faster, refer to BPI-2013 event log. This observation can be explained by the fact that *FedGASPD* and *FedCGASPD* perform

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

research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

Table 3: Execution times of *GASPD*, *FedGASPD*, and *FedCGASPD* algorithms

Event log	Execution time(in seconds)						
	GASPD	*FedGASPD*			*FedCGASPD*		
		2 nodes	4 nodes	8 nodes	2 nodes	4 nodes	8 nodes
BPI-2012	416	178	83	61	166	81	58
BPI-2013	84	35	26	11	43	22	17
BPI-2017	3,120	1,459	514	342	1,401	499	367
BPI-2018	2,898	1,325	165	76	1,317	173	78
BPI-2019	2,288	2,216	471	251	2,224	480	260
Request for Payment	1402	685	331	229	676	342	236
PrepaidTravelCost	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

computationally demanding *GASPD* genetics on many smaller event logs in parallel and then apply a computationally efficient merging of the resulting models.

Figure 9 shows the characteristics of the Pareto-optimal models discovered from the nine event logs by the three algorithms for the different numbers of computation nodes, while Tables 4 and 5 summarize the ranges of the sizes and entropic relevance values of the discovered models. These results further support our contention that federated approaches yield useful new models compared to the centralized solution. When comparing the two federated approaches, *FedCGASPD* results more often in smaller models than *FedGASPD*, as it tends to merge models of similar characteristics and, hence, preserve the distinctive sizes of the various groups of similar models. *FedGASPD*, however, tends to discover models of lower entropic relevance, hence describing the traces and their frequencies from the input event logs more accurately. This observation confirms the trade-off between the size and quality of the discovered models [23] and suggests that the two proposed methods prioritize different aspects of this trade-off.

We conclude that *FedGASPD* and *FedCGASPD* discover models of comparable characteristics as *GASPD*. Table 6 presents Pareto dominance counts for the various techniques. Note that the centralized solution, *GASPD*, does not consistently dominate across different datasets. Specifically, it delivers a good share of Pareto-optimal solutions for BPI-2017, BPI-2020-2, Sepsis Cases, and Hospital Billing event logs, but not for other event logs, where *FedCGASPD* discovers the lion's share of Pareto-optimal models.

6.3 Partial Model Sharing

Next, we focus on federated process discovery based on partial model sharing. In the first scenario (asynchronous sharing), we compare P^2*FedGASPD* with *FedGASPD* by incrementally increasing the size of the partial DFFA model that each node shares with the central server until the difference in entropic relevance between the global and local models falls below a threshold (ϵ). In the second scenario (synchronous sharing), all nodes synchronously share a fixed percentage of their local models with a central server, gradually increasing the percentage until they share their entire model. This approach allows us to observe how the quality of the global model improves as more

Preprint not peer reviewed
t research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

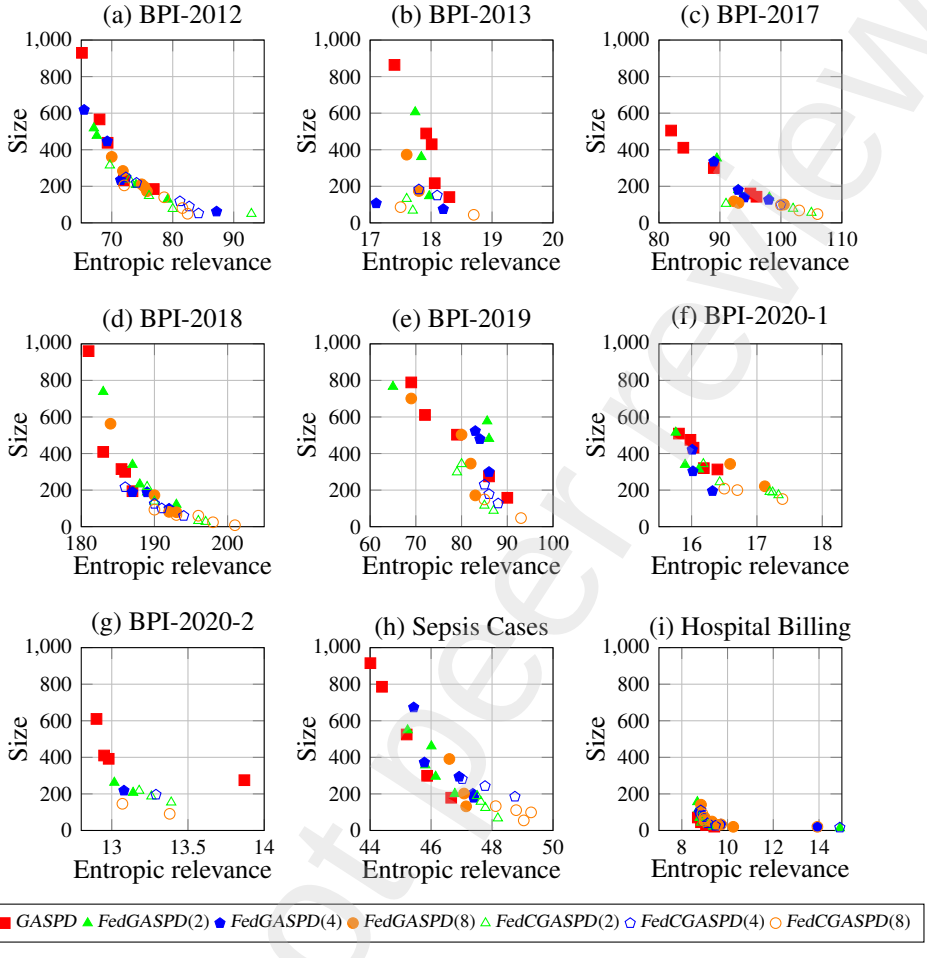


Figure 9: Size and entropic relevance of DFFAs discovered by *GASPD*, *FedGASPD*, and *FedCGASPD*; smaller sizes and entropic relevance values signify better models. The numbers in the parenthesis signify the numbers of computation nodes.

information is collectively contributed.

6.3.1 Asynchronous Sharing

In this scenario, we limit the model size to less than 200 and set $\epsilon = 0.5$, meaning that if the difference in entropic relevance between the global and local models is less than 0.5, the client stops sharing additional local models.

Figure 10 shows the characteristics of the Pareto-optimal models discovered from the nine event logs by *P*^{2}*FedGASPD* and *FedGASPD* for the different numbers of computation nodes (clients), while Table 7 compares the qualities of the Pareto-optimal models overall. Each column in the table indicates by how many solutions the number of Pareto-optimal models of *P*^{2}*FedGASPD* exceeds that of *FedGASPD*; for ex-

research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

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

Event Log	Size											
	GASPD		*FedGASPD*						*FedCGASPD*					
			2 nodes		4 nodes		8 nodes		2 nodes		4 nodes		8 nodes	
	min	max	min	max	min	max	min	max	min	max	min	max	min	max
BPI-2012	185	930	128	856	173	361	52	249	62	446	50	315	11	205
BPI-2013	486	760	143	715	174	373	130	291	107	311	67	218	47	190
BPI-2017	143	460	166	507	100	184	98	121	125	7500	56	154	48	121
BPI-2018	196	403	120	339	81	282	59	125	98	318	27	218	8	66
BPI-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
PrepaidTravelCost	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

Table 5: Relevance of DFFAs discovered by *GASPD*, *FedGASPD*, and *FedCGASPD*

Event Log	Entropic relevance													
	GASPD		*FedGASPD*						*FedCGASPD*					
			2 nodes		4 nodes		8 nodes		2 nodes		4 nodes		8 nodes	
	min	max	min	max	min	max	min	max	min	max	min	max	min	max
BPI-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-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-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-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-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
PrepaidTravelCost	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	46.6	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	9.16	08.83	14.91	08.9	13.93

Table 6: Pareto dominance counts

Event log	*GASPD*	*FedGASPD*			*FedCGASPD*			Total score
		2 nodes	4 nodes	8 nodes	2 nodes	4 nodes	8 nodes	
BPI-2012	2 (10.52%)	**4 (21.05%)**	2 (10.52%)	0 (00.00%)	3 (15.78%)	0 (00.00%)	**4 (21.05%)**	19 (100.00%)
BPI-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-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-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-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%)
PrepaidTravelCost	**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%)

ample, +2 means that *P*^{2}*FedGASPD* has two more Pareto-optimal models remaining after comparison than *FedGASPD*. Table 8 shows the average percentage of each local model shared by the nodes. For instance, 76.80 means that each node or client, on average, shares 76.80 % of its local model with the server until the quality of the global model falls within the ϵ threshold relative to the local model.

Our results indicate that the *P*^{2}*FedGASPD* approach can discover more accurate and more compact models than *FedGASPD*. This improvement arises from sharing

t research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

Table 7: Pareto dominance counts comparison.

Event log	*FedGASPD* vs *P²FedGASPD*		
	2 nodes	4 nodes	8 nodes
BPI-2012	+2	+3	+5
BPI-2013	+4	+5	+5
BPI-2017	-1	+3	+1
BPI-2018	-1	+4	+1
BPI-2019	-2	+4	+2
Request for Payment	0	+3	+4
PrepaidTravelCost	+3	+1	+5
Sepsis Cases	+1	+5	+7
Hospital Billing	+8	+3	+8

Table 8: Average local model sharing.

Event log	Sharing percentage (%) *P²FedGASPD*		
	2 nodes	4 nodes	8 nodes
BPI-2012	76.80	71.12	61.14
BPI-2013	56.25	42.50	27.75
BPI-2017	97.16	94.66	92.33
BPI-2018	86.33	70.00	53.14
BPI-2019	70.66	68.50	61.88
Request for Payment	81.83	77.54	60.90
PrepaidTravelCost	95.37	94.88	92.00
Sepsis Cases	58.75	50.33	44.00
Hospital Billing	80.40	75.20	65.88

only partial models that capture the core and most representative behaviors of each client, thereby avoiding the propagation of exceptional or idiosyncratic behaviors. If a behavior is rare on one client but frequent on another, the latter reinforces it during the federated merging process. Only behaviors that are infrequent across all clients fail to appear in the merged global model. This collective reinforcement of common behavior explains why *P²FedGASPD* produces smaller yet more accurate global models.

6.3.2 Synchronous Sharing

In the second scenario, all nodes 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 models are 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 nodes share 100% of their local models. By measuring the quality difference at each stage, we assess how the global model improves as more information becomes available.

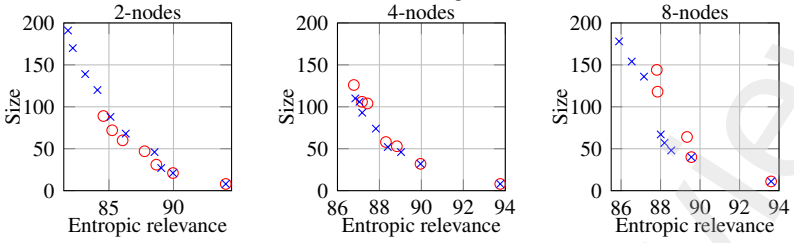
As expected, once the sharing level reaches 100%, the quality of the resulting global model aligns with that obtained under full model sharing. Figure 11 shows the difference in the average entropic relevance between partial and complete sharing. Negative values indicate that partial sharing achieves lower entropic relevance and, therefore, higher model quality. For example, in Fig. 11-c (left plot), for the 4-node scenario, sharing 85% of each local model yields a global model whose average entropic relevance is approximately half that of the model obtained under full sharing. Figure 11 also reports the size of the global model at different sharing levels, showing how partial sharing leads to more compact models. We conclude that sharing approximately 60% of local models is sufficient to construct high-quality, small global models.

7 Conclusions and Future Work

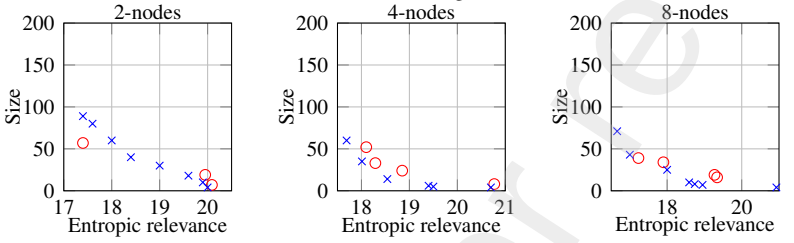
This paper presents the *FedGASPD* and *FedCGASPD* algorithms for stochastic process discovery in distributed environments. These algorithms are built upon the robust foundation of *GASPD*, a stochastic process discovery algorithm that operates over cen-

Preprint not peer reviewed
This research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

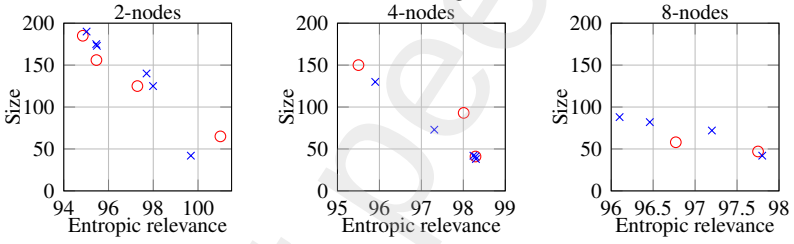
(a) BPI-Challenge-2012



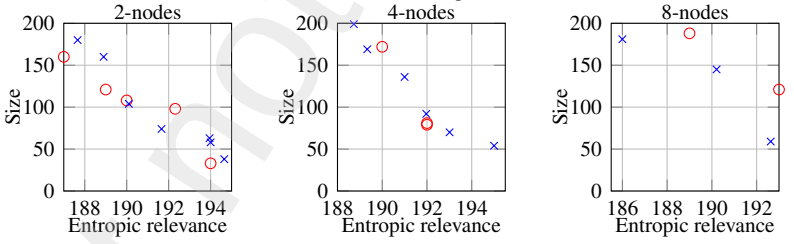
(b) BPI-Challenge-2013



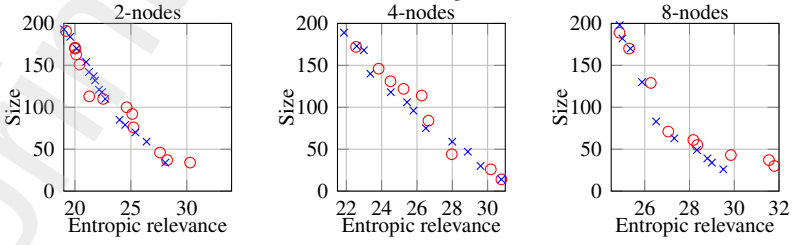
(d) BPI-Challenge-2017



(e) BPI-Challenge-2018



(f) BPI-Challenge-2019



Preprint not peer reviewed
t research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

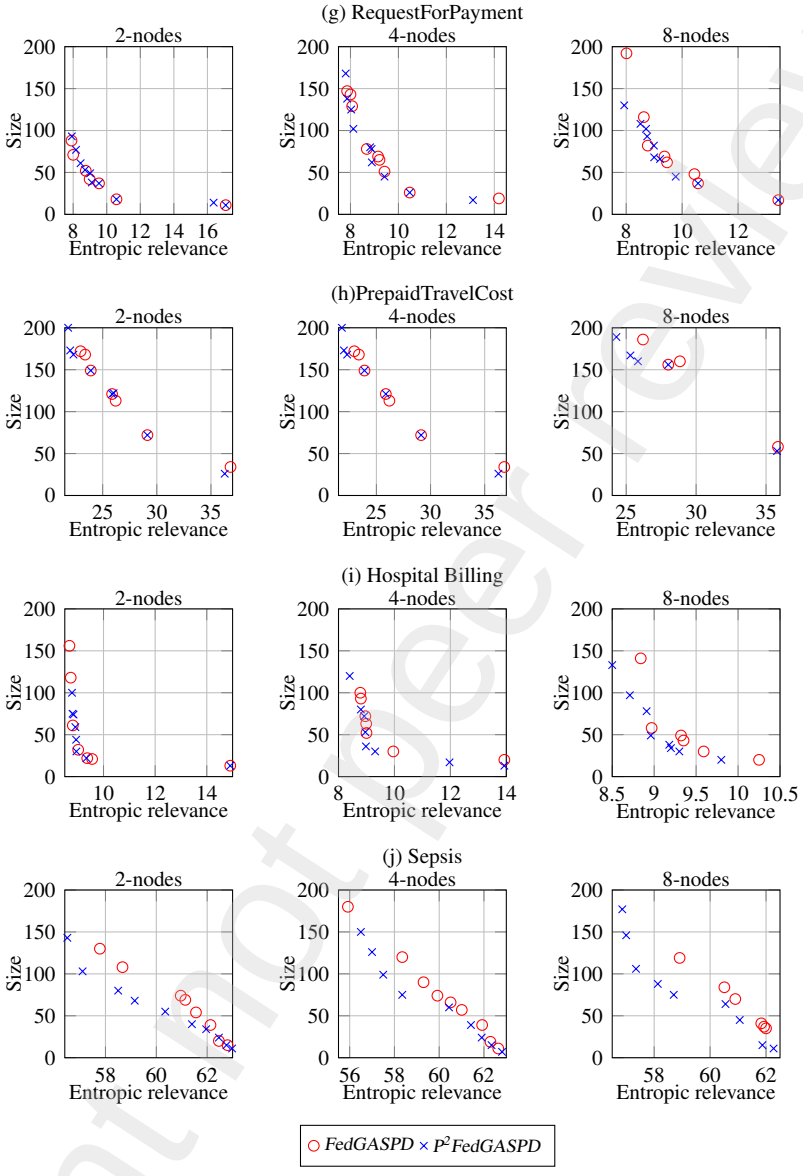


Figure 10: Size and entropic relevance of DFFAs discovered by *FedGASPD* and *P²FedGASPD*; smaller sizes and entropic relevance values signify better models.

tralized event logs. The new algorithms discover process models from several local event logs, possibly scattered across different organizations, and aim to preserve the autonomy and privacy of each party and to decrease data communication requirements and the overall model discovery time. Our experiments with real-world event logs demonstrate the effectiveness of *FedGASPD* and *FedCGASPD*, providing scalable al-

research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

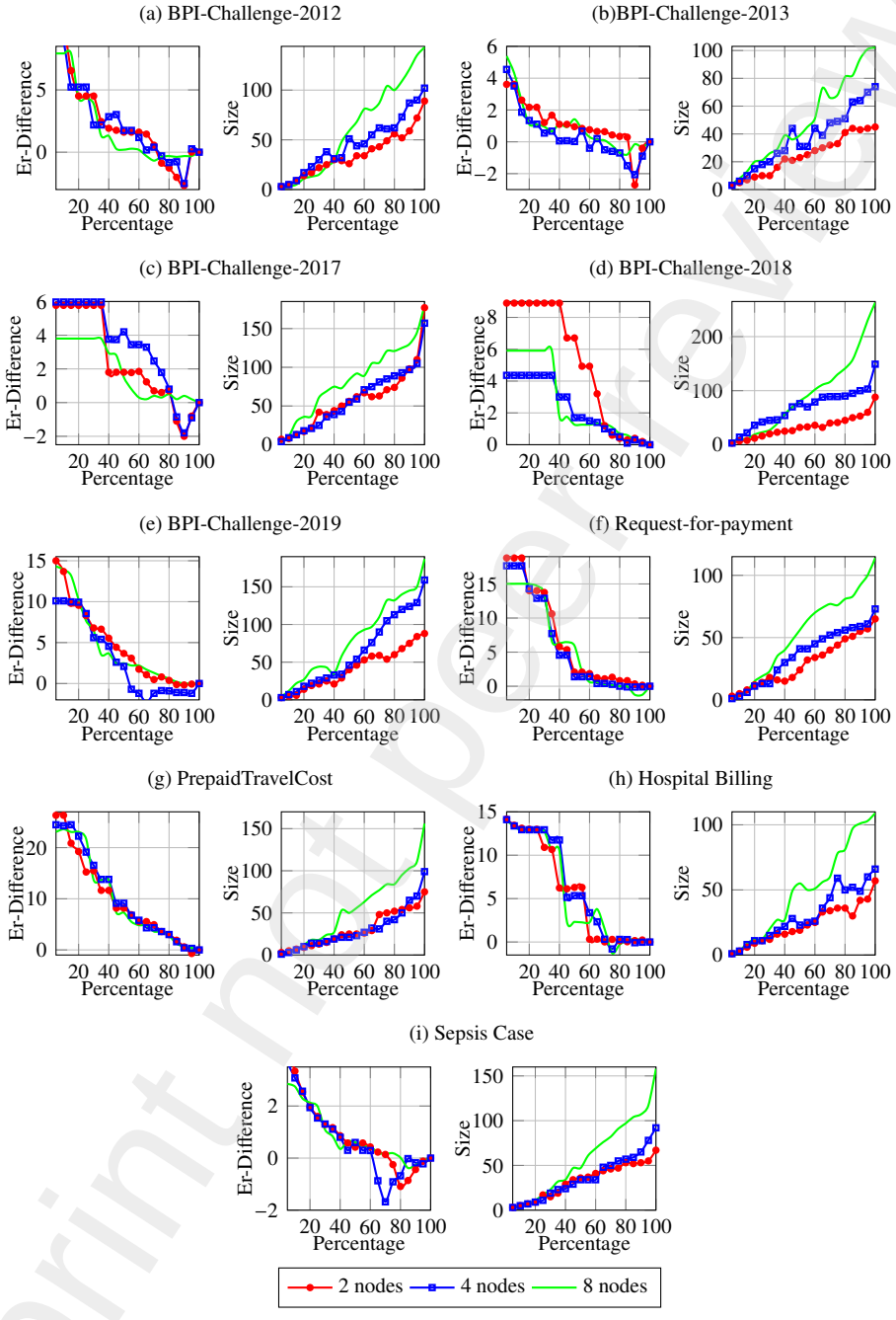


Figure 11: Impact of increased local model sharing on aggregated model quality.

Preprint not peer reviewed
research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

ternatives for process discovery in distributed environments. The algorithms discover DFFAs that can be translated, using the native translation of *GASPD*, into sound DFGs.

While federated approaches improve privacy by avoiding the sharing of raw data, sharing complete local models can still expose rare or sensitive behaviors. To address this, we introduce *P²FedGASPD*, a privacy-preserving approach that shifts the focus from data-centric to model-centric privacy. *P²FedGASPD* enables each client to share only a portion of its local model, systematically excluding infrequent or unique behaviors that could pose an attack risk. This partial model selection is formulated as a maximum-weight connected subgraph problem, ensuring that only the most frequent and representative behaviors are included. Experimental results show that, in many cases, *P²FedGASPD* can achieve a global model comparable in size and entropic relevance to one produced by sharing entire local models. This is made possible by its partial model-sharing mechanism, which not only preserves important behaviors but also filters out infrequent ones, helping protect privacy.

By combining federated approaches such as *FedGASPD* with model-centric privacy mechanisms such as *P²FedGASPD*, organizations can collaboratively discover promising quality process models while reducing privacy risks. In future work, we aim to formally define attack models and quantify the risk of disclosing partial models. Additionally, we will study how sharing partial models from different clients may affect the risk of attacks on others.

Although the presented results suggest that the models constructed by *FedGASPD* and *FedCGASPD* are comparable in quality to those discovered by *GASPD*, several avenues exist to further enhance the findings. First, a systematic exploration of the impact of different orders of merging DFFAs discovered from local event logs on the quality of the constructed global DFFAs and, consequently, DFGs, could provide valuable insights. Similarly, investigating clustering techniques that improve the performance of *FedCGASPD* may lead to significant advancements. Additionally, optimizing the discovered models for other quality criteria than size and relevance could further refine their applicability. Finally, exploring alternative policies for selecting superior local models represents an interesting direction for future research.

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, and Alistair Moffat. Stochastic directly-follows process discovery using grammatical inference. In *CAiSE*, volume 14663 of *LNCS*, pages 87–103. Springer, 2024.

[3] 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.

[4] 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.

This research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

[5] 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.
[6] 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.
[7] Colin de la Higuera. *Grammatical Inference*. Cambridge University Press, Cambridge, 2010.
[8] Gamal Elkoumy, Alisa Pankova, and Marlon Dumas. Privacy-preserving directly-follows graphs: Balancing risk and utility in process mining. *Cryptography and Security*, 2020. URL arXivpreprintarXiv:2012.01119.
[9] 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.
[10] Gamal Elkoumy, Alisa Pankovab, and Marlon Dumasa. Differentially private release of event logs for process mining. *Information Systems*, 115, 2023.
[11] Javier Esparza and Keijo Heljanko. *Unfoldings: A Partial-Order Approach to Model Checking*. EATCS Monographs in Theoretical Computer Science. Springer, 2008.
[12] Stephan A. Fahrenkog-Petersen, Martin Kabierski, Fabian Rosel, Han van der Aa, and Matthias Weidlich. Sacofa: Semantics-aware control-flow anonymization for process mining. In *3rd International Conference on Process Mining*, pages 72–79. IEEE, 2021.
[13] 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.
[14] Martin Kabierski, Stephan A. Fahrenkrog-Petersen, and Matthias Weidlich. Hiding in the forest: Privacy-preserving process performance indicators. *Information Systems*, 112, 2023.
[15] 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.
[16] 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.
[17] 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.
[18] 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.
[19] 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.
[20] 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.
[21] 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.
[22] European parliament and the council of the european union. regulation. Techni-

This research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

cal report, 2016. URL https://eur-lex.europa.eu/legal-content/EN/TXT/?uri=CELEX%3A32016R0679.

[23] 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.

[24] 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.

[25] 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.

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

[27] 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.

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

[29] Majid Rafiei, Mahsa Pourbafrani, and W.M.P. van der Aalst. Federated conformance checking. *Information Systems*, 131, 2025.

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

[31] 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.

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

[33] 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.

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

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

[36] 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.

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

[38] 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.

[39] 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, Christian Günther, Antonella Guzzo, Paul Harmon, Arthur ter Hofstede, John Hoogland, Jon Espen Ingvaldsen, Koki Kato, Rudolf Kuhn, Akhil Kumar, Marcello La Rosa, Fab-

This research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra

rizio 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.

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

[41] 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.

[42] Hootan Zhian, Rajkumar Buyya, and Artem Polyvyanyy. Federated stochastic process discovery using grammatical inference. In *CAiSE*, volume 15702 of *LNCS*, pages 76–93. Springer, 2025.

Preprint not peer reviewed
t research paper has not been peer reviewed. Electronic copy available at: https://ssrn.com/abstra