Process Trace Clustering: A Heterogeneous Information Network Approach

Phuong Nguyen[1], Aleksander Slominski[2], Vinod Muthusamy[2], Vatche Ishakian[2], and Klara Nahrstedt[1]

[1]University of Illinois at Urbana-Champaign, [2]IBM T.J. Watson Research Center

Abstract

Process mining is the task of extracting information from event logs, such as ones generated from workflow management or enterprise resource planning systems, in order to discover models of the underlying processes, organizations, and products. As the event logs often contain a variety of process executions, the discovered models can be complex and difficult to comprehend. Trace clustering helps solve this problem by splitting the event logs into smaller subsets and applying process discovery algorithms on each subset, resulting in per-subset discovered processes that are less complex and more accurate. However, the state-of-the-art clustering techniques are limited: the similarity measures are not process-aware and they do not scale well to high-dimensional event logs. In this paper, we propose a conceptualization of process's event logs as a heterogeneous information network, in order to capture the rich semantic meaning, and thereby derive better process-specific features. In addition, we propose SeqPathSim, a meta path-based similarity measure that considers node sequences in the heterogeneous graph and results in better clustering. We also introduce a new dimension reduction method that combines event similarity with regularization by process model structure to deal with event logs of high dimensionality. The experimental results show that our proposed approach outperforms state-of-the-art trace clustering approaches in both accuracy and structural complexity metrics.

1 Introduction

Motivation: With the advancement in information technology enterprises have opted to automate their daily operations. Real-world *business processes* including airline ticket reservation, online shopping, and administrative procedures such as procurement and hiring are invoked electronically using information systems. Figure 1 shows a simple *process model*—the graphical representation of a process—to manage a loan application. The nodes correspond to activities, or *events*, and the edges define the control flow. Each execution instance, or *trace*, of a process is defined as an ordered list of events executed from the beginning to the end. The information systems responsible for executing instances of a process generate a lot of data in the form of *event logs*, or a collection of *process traces*[1]. Figure 2 shows an example of a process trace that consists of a sequence of events (each line is an event) and represents an instance of process in Figure 1.

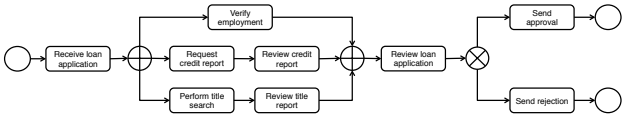


Figure 1: An example loan application's process model

```
TraceID,Timestamp,Resource,Activity
Trace-10,10/16/2014 10:12:20,Resource21,Record loan application
Trace-10,10/18/2014 09:30:00,Resource10,Request credit report
Trace-10,10/25/2014 15:10:16,Resource21,Review credit report
Trace-10,10/28/2014 14:15:28,Resource21,Send rejection
```

Figure 2: Traces based on the loan application process model

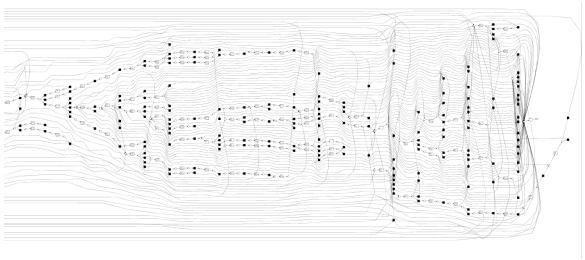


Figure 3: A complex mined process model

Problem Description: In many situations the process model as defined in Figure 1 may not be available or may be obsolete. *Process mining* algorithms [16, 17] can then be used to derive the process model from the event logs. Mining for process model is also important to obtain insights into how the process is being executed in real-life. In particular, the aim of process mining is to *discover* or understand an organization's processes, *monitor* to validate whether the implemented process conforms to the model, and to *improve* the process by redesigning it to avoid observed bottlenecks.

Mining a process model from all observed traces often results in spaghetti-like models [23] as shown in Figure 3, which illustrates part of the real-world process for building permit application. These models are too complicated for humans to comprehend because of their complexity and inaccuracy, and too cumbursome to modify or optimize. A common solution is to *cluster* the traces [8] into coherent sets of traces, and represent each set by a process model.

[1]We use *event logs* and *process traces*, or simply *traces*, interchangeably.

As traces are clustered largely based on the notion of similarity, a good trace similarity measure results in better quality clusters. Thus, existing solutions to process trace clustering mainly focus on mapping traces to appropriate data structures [8, 11], and deriving new similarity measures [2, 1] that can be used by off-the-shelf clustering algorithms [7]. These approaches lack certain aspects that we believe are essential to having good quality clusters.

First, existing approaches fail to capture the rich semantic relationships between events from the event logs. For example, events that share an underlying role or belong to the same group of events, should still have a certain level of similarity. Events that are executed/invoked by the same resource (or person) are also likely to be similar. Since additional information (e.g., organization, roles, products information, etc.) about traces is typically available in real-world process traces, it would be desirable to to capture the extra semantics inferred from such information.

Second, existing approaches, especially the edit distance-based approaches [1] are not scalable. Computing the pair-wise similarity of a large set of traces each of which contains many events is very expensive, since the complexity of edit distance-based measures is quadratic with the length of traces. Efforts to apply standard dimension reduction techniques to process mining are limited to vector space model-based approaches, whose similarity calculation is not as expensive [12].

Scope and Contributions: In this paper, we propose a new process trace clustering approach that tries to resolve both of the above issues. Particularly, for the *semantic gap issue*, we propose a new data representation for process traces based on extensible heterogeneous information networks [13] and capture the rich process semantics in the nodes and edges of the network. With this representation, users can select intuitive *meta-paths* between nodes in the network to model different semantic relationships between process traces. While the selected meta-paths can be used directly to calculate the similarity between traces using existing `PathSim` measure [14], `PathSim` does not capture the sequential similarity between sequences of events in traces. Therefore, we propose `SeqPathSim`, a similarity measure that combines both the peer similarity captured by `PathSim` and the sequential similarity between traces captured by edit-distance based approaches. To overcome performance issues of edit distance-based approaches, we propose a new dimension reduction method that is tailored for process traces. Particularly, we model the dimension reduction as an optimization problem and propose an objective function that maximizes both topical similarity and process model-based relationships between events of the same dimension. Since such an optimization problem is NP-hard, we propose a greedy approximation algorithm to solve the problem. Experimental results using real-world datasets that span multiple business domains show that our proposed approach outperforms state-of-the-art process trace clustering approaches both in terms of effectiveness and efficiency.

Paper Overview: The remaining of the paper is organized as follow. We review relevant related work in Section 2, and in Section 3, we present some background and basic concepts about heterogeneous information networks. We formally define the trace clustering problem in Section 4. In Section 6, we present our modeling of process traces as an extendable heterogeneous information network. We describe our proposed `SeqPathSim` measure in Section 6 and our dimension reduction method in Section 7. The experimental results are presented in Section 8. We conclude in Section 9 with closing remarks and future research directions.

2 Related Work

Process mining [16, 17] is the task of extracting knowledge from event logs in order to discover, monitor, and improve processes. In terms of discovery, a number of techniques have been developed to discover process models from event logs, such as the alpha miner [20], heuristic miner [19], and fuzzy miner [9]. There are also different notations for representing process models, such as Petri nets [18], UML activity diagrams, and BPMN [21]. In this paper, we use Petri nets to model processes and use the heuristic miner to discover process models. For monitoring, conformance fitness is used to answer the question "Do the model and the log conform to each other?". We use the conformance checking method developed by Rozinat *et al.* [10] that is implemented within the ProM framework [22]. There have been variety of metrics proposed to evaluate the complexity of process models [1, 3], and we adopt the structural complexity metrics for Petri nets [1].

Process trace clustering research has focused on developing new data representations for traces, and deriving new similarity measures between traces that can be used with off-the-shelf clustering algorithms. One of the most common trace representations is the vector space (or bag-of-events) model [8, 11], where each trace is represented as a vector and each dimension corresponds to an event type. Similarity between traces is measured using typical measures such as Euclidean distance or Cosine similarity. To better capture the order of execution of events, Greco *et. al* [8] use a k-*gram* representation (i.e., a sequence of k consecutive events is used as one dimension in a vector space). This approach, however, does not scale well as the number of dimensions increases exponentially with k. Other approaches [2, 1] propose using sequence-based similarity measures such as Hamming distance and edit-distance. In Particular, Bose *et al.* [1] derive edit operation costs based on a 3-gram context of events. Weerdt *et al.* [3] consider the semantic gap issue between traces clustering and generic clustering algorithms and propose an active learning-based clustering approach to

bridge the gap. We focus on the data representation of traces and propose a new similarity measure, SeqPathSim, that improves on previous approaches by combining the rich semantic of peer similarity with the sequential distance metric by edit-distance.

Mining heterogeneous information networks (HIN) [13] is a research direction that has gained a lot of attention recently for its ability to capture the rich semantics of structural types of nodes and edges in the network. Sun *et al.* propose PathSim [14], a meta path-based similarity measure that captures the similarity semantics among peer objects in HIN. Based on PathSim, a user-guided clustering approach [15] is also proposed to cluster objects in an HIN. In this paper, we propose another similarity measure, i.e., SeqPathSim, between *sequences of objects* in an HIN. We leverage HINs as the data representation for process traces and derive the similarity between traces that can be used with any existing clustering algorithm.

Dimension reduction techniques have been suggested to address the curse of high dimensional datasets [4]. There are also efforts [12] to use dimension reduction techniques to improve the performance of process trace clustering, but those efforts mainly focus on vector space model representation. We propose a new dimension reduction approach for sequence-based representation of traces that combines an off-the-shelf dimension reduction technique with regularization by the process execution model to improve the performance results for process traces.

3 Preliminaries: Heterogeneous Information Network

A heterogeneous information network [13] is an information network (or graph) with multiple types of nodes (vertices) and multiple types of links (edges).

DEFINITION 3.1. ***Heterogeneous Information Network (HIN)****: An HIN is a directed graph* $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ *with a node type mapping function* $\phi : \mathbf{V} \rightarrow \mathcal{A}$*, where* $\mathcal{A}$ $(|\mathcal{A}| > 1)$ *is the set of node types, and an edge type mapping function* $\psi : \mathbf{E} \rightarrow \mathcal{R}$*, where* $\mathcal{R}$ $(|\mathcal{R}| > 1)$ *is the set of edge types.*

An example HIN is the bibliographic network that contains multiple types of nodes, such as papers (P), venues (C), and authors (A), and multiple types of edges, such as submissions (between P and C), and citations (between P and P).

Multiple paths may exist between two nodes in HINs. A *meta-path*, described by a sequence of relations in the HIN that connects two types of nodes is used to capture the underlining semantic of each path. For example, APA represents the co-author relationship between authors, or ACP represents the paper submission relationship.

To measure similarity between nodes in HINs, Sun *et al.* [14] propose PathSim (Definition 3.2), a similarity measure that takes advantage of the rich semantic structure in the network and captures the true peer similarity between nodes in HINs.

DEFINITION 3.2. PathSim*: Given a symmetric meta-path* $\mathcal{P}$*, PathSim between two objects of the same type* x *and* y *via meta-path* $\mathcal{P}$*, denoted as* $\sigma_{\mathcal{P}}(x, y)$*, is defined as:*

$$\sigma_{\mathcal{P}}(x, y) = \frac{2 \times |\Gamma_{\mathcal{P}}(x, y)|}{|\Gamma_{\mathcal{P}}(x, x)| + |\Gamma_{\mathcal{P}}(y, y)|} \tag{3.1}$$

where $\Gamma_{\mathcal{P}}(x, y)$ *is the set of paths from* x *to* y *via meta-path* $\mathcal{P}$*.*

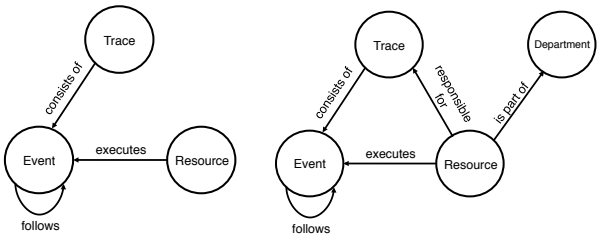
4 Problem Definition

Consider a set of *process traces* $\mathbb{T}$. Each trace $t \in \mathbb{T}$ consists of a finite sequence of *events* $t = (e_1, e_2, ..., e_n), e_i \in \mathbb{E}, 1 \leq i \leq n, n > 0$, where $\mathbb{E}$ is the set of all event types. The number of events per trace n may be different from trace to trace. For each event e_i in a trace t, there is an associated *resource* $r_j \in \mathbb{R}$ that generates or executes the event, with $\mathbb{R}$ being the set of all resources.

As highlighted in Section 1, discovering process models using the entire set of process traces, may result in spaghetti-like models [16]. Therefore, in this paper, we study the trace clustering problem that divides $\mathbb{T}$ into non-overlapping subsets of clusters $\{\mathbb{T}_i\}$, so that each cluster represents an underlying process model with higher accuracy.

Unlike classic data clustering problems [7] where the objective is to either maximize the precision and recall (in cases where ground-truth labels are available), or minimize the intra-cluster and maximize the inter-cluster distances (when ground-truth labels are not available), the objective of trace clustering is to group traces that share similar execution patterns, thus enabling discovery of process models with *high degrees of fitness*. Specifically, the process model's fitness quantifies the extent to which the discovered model can accurately reproduce the recorded traces. In addition, a good trace clustering result should also generate clusters whose process models are simple and compact. More precisely, they should exhibit *low degrees of structural complexity*.

Therefore, in this paper, we use the two metrics that are widely used in other process trace clustering work [1, 2, 3, 11]: *weighted average fitness*, denoted as AveFitness, and *weighted average structural complexity*, denoted as AvgComplexity. Particularly, for each cluster in a clustering result, a process model is generated (e.g., using the Heuristic mining algorithm [19]) and then converted to the Petri-Net model [18] for conformance analysis. The conformance fitness score of a discovered process model is the fraction of traces that can be fully replayed on that model. A process model has a perfect fitness score if all the recorded traces can be replayed by the model. The weighted average conformance fitness over a set of k clusters $\{\mathbb{T}_i\}$ of traces is defined as $\texttt{AvgFitness} = \frac{\sum_{i=1}^{k} |\mathbb{T}_i| \cdot \texttt{Fitness}(\mathbb{T}_i)}{\sum_{i=1}^{k} |\mathbb{T}_i|}$, where $\texttt{Fitness}(\mathbb{T}_i)$ is the fitness score of a cluster of traces $\mathbb{T}_i$. A higher fitness score implies a more accurate process model for a given cluster.



(a) Basic HIN model (b) Extended HIN model

Figure 4: HIN models for process traces.

The structural complexity is measured based on the complexity of the graphical representation of a process model. Specifically, given a process model represented as a Petri net, the complexity is measured by counting the number of control-flows, AND-joins and splits, and XOR-joins and splits that appear in the process model. Similar to `AvgFitness`, `AvgComplexity` is the weighted average of the complexity metrics based on the cluster sizes. A lower structural complexity score implies a simpler, more compact model, that is potentially better understandable by humans.

Given these clustering result quality metrics, the process trace clustering problem is formally defined as follows:

DEFINITION 4.1. ***Process trace clustering:*** *Let $\mathbb{T}$ be a set of process traces, $\mathbb{E}$ a set of events, and $\mathbb{R}$ as set of resources. Find a k-partition $\{\mathbb{T}_i\}$ of $\mathbb{T}$ ($k \geq 2$): $|\{\mathbb{T}_i\}| = k, \mathbb{T}_i \cap \mathbb{T}_j = \emptyset, \forall\ 1 \leq i, j \leq k$ that maximizes the average fitness* `AvgFitness`$(\{\mathbb{T}_i\})$*, and minimizes the average structural complexity* `AvgComplexity`$(\{\mathbb{T}_i\})$.

Like other clustering problems, the effectiveness of trace clustering largely depend on how one defines the notion of *similarity* between traces. Therefore, in this work, we focus on deriving a similarity function `sim` to measure the similarity between two traces. This similarity function can be used with off-the-shelf clustering algorithms [7] to produce results of high fitness and low structural complexity.

5 Modeling Process Traces as HIN

Motivated by the effectiveness of HINs to capture the peer similarity between nodes in other domains [13, 14], we propose the modeling of process traces as a heterogeneous graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ (Figure 4a) with the set of nodes $\mathbf{V} = \mathbb{T} \cup \mathbb{E} \cup \mathbb{R}$ that consists of three node types: trace, event, and resource. The set of edges $\mathbf{E}$ outline the different types of interactions between different node types. We define the following edge types (i.e., $\mathcal{R}$):

- `consist-of`: An event is a part of a trace.
- `follow-up`: An event follows another event in a trace.
- `execute`: An event is executed or generated by a resource.

As we show in Section 8, the above node types and edge relations are chosen to be the most basic types and relations, so that they are generic enough to capture a wide variety of traces from different domains. Nevertheless, one can easily augment the HIN model with additional types of nodes and edges targeting a specific business process domain. For example, an extended HIN model, shown in Figure 4b, includes an additional node type "$Department$", and two edge types: `is-part-of`, which captures the relationship between a resource and the department it belongs to, and `responsible-for`, which indicates the resource is responsible for a trace.

Meta-paths: Given the HIN model described above, we define the following meta-paths:

- TET: Meta-path between two traces that share common event(s).
- $TEET$: Meta-path between two traces that consist of consecutive events.
- $TERET$: Meta-path between two traces that consist of events executed by the same resource.

6 Meta-Path based Similarity Measures

PathSim-based Similarity Measure: By modeling process traces as an HIN, their similarity can be calculated using the similarity measure between trace-typed nodes in the HIN. In particular, we consider `PathSim` [14] similarity based on the meta-paths described in Section 5.

It has been shown that the linear combination of multiple meta-paths results in better outcomes than that of an individual meta-path [14]. Thus, in this paper, we combine the `PathSim` similarities obtained by individual meta-paths using the following linear formula:

$$\sigma^*(x, y) = \sum_{\mathcal{P}_i} w_i \times \sigma_{\mathcal{P}_i}(x, y) \tag{6.2}$$

where $\sigma_{\mathcal{P}_i}(x, y)$ is the `PathSim`-based similarity between two traces x and y via meta-path $\mathcal{P}_i$, and w_i is the weight associated with meta-path $\mathcal{P}_i$. Finding the optimal set of weights $\{w_i\}$ is still an open problem and is orthogonal to our work. Similar to previous work [14, 15], we assume that meta-paths are selected with user guidance.

After calculating the `PathSim` similarity between every pair of traces using formula 6.2, we can apply any off-the-shelf clustering algorithm (*e.g.*, hierarchical clustering) to cluster the input process traces.

SeqPathSim-based Similarity Measure: Modeling process traces as an HIN helps capture the rich semantics of structural types of nodes and edges in the network. HINs however do not maintain the sequential order of events in each process trace. As a result, `PathSim` cannot measure the similarity between traces that share similar event execution orderings. For example, the `PathSim` based on the $TEET$ meta-path represents only the sequential relationship between two consecutive events. Since traces consist of a sequence of

multiple events, traces sharing the same sequential execution should typically have higher similarity scores than traces that do not. Therefore, it is essential to have a new similarity measure that captures the similarity between two sequences of events (*i.e.*, two traces) in an HIN.

The edit distance [5] quantifies how similar two sequences are by counting the minimum number of operations required to transform one sequence into the other. Edit distance has shown its effectiveness in measuring similarity between sequence-like data traces in multiple domains, such as text mining, process mining, and bioinformatics.

Motivated by the need to leverage the similarity between two sequential executions of events in process traces, we propose `SeqPathSim`: a new similarity measure for HINs that combines the rich semantic relationships between nodes captured by `PathSim` with the sequential similarity captured by edit distance. In particular, `SeqPathSim` uses generic edit-distance [5].

The performance of generic edit distance depends significantly on how one defines the costs of editing operations (*i.e.*, replace, delete, and insert). For example, using unit cost, *i.e.*, Levenshtein's distance [6], has been shown to be effective in string similarity tasks. Particularly, we consider two types of editing costs: *insertion/deletion cost* (the cost to insert or delete an event before or after another event), and *replacement* cost (the cost to replace an event by another event). For insertion/deletion cost, we use the `PathSim` based similarity via an EE meta-path (the path between an event that follows another event), since this meta-path captures how likely an event is executed before or after another event. For replacement cost, we use a combination of the `PathSim` based similarity via ERE (representing two events that are executed by the same resource) and ETE (representing events that are part of the same trace), since these meta-paths capture indirect similarities of two events.

Similar to generic edit-distance, `SeqPathSim` between two traces $x = (a_1, a_2, ..., a_m)$ and $y = (b_1, b_2, ..., b_n)$ ($a_i, b_j \in \mathbb{E}, 1 \leq i \leq m, 1 \leq j \leq n$), denoted as $\upsilon_{mn}(x, y)$, or $\upsilon_{m,n}$ for short, is defined by the following recursive formula (Equation 6.3).

$$\upsilon_{m,n} = \begin{cases} \upsilon_{m-1,n-1} & \text{for } a_m = b_n \\ min_\upsilon & \text{for } a_m \neq b_n \end{cases} \tag{6.3}$$

with:

$$min_\upsilon = \begin{cases} \upsilon_{m-1,n} + \sigma_{\mathcal{P}_{EE}}(a_m, b_n) \\ \upsilon_{m,n-1} + \sigma_{\mathcal{P}_{EE}}(a_m, b_n) \\ \upsilon_{m-1,n-1} + \sigma_{\mathcal{P}_{ERE}, \mathcal{P}_{ETE}}(a_m, b_n) \end{cases}$$

7 Optimizing SeqPathSim for High-dimensionality

`SeqPathSim` leverages both the rich semantic relationships between nodes captured by `PathSim` and the sequential similarity by edit-distance. It also inherits the performance drawbacks of edit distance-based measures. Recall that the complexity of generic edit-distance is $O(m * n)$, where m and n are the lengths of two sequences being compared. Moreover, we need to calculate the similarities between every pair of traces. Clustering real-world traces that are often of high dimensionality (*e.g.*, up to hundreds of events per process trace resulting in high values of m and n) using `SeqPathSim` is therefore computationally expensive.

Figure 5: Dimension-reduced loan application's process model

Despite the high number of dimensions, we observe that comparing process traces does not require the traces to be represented at the fine-grained level of events. As an example, consider the loan application process in Figure 1. At a higher level of abstraction, the process essentially consists of three main steps: receiving the application, reviewing the application, and informing with a decision. Therefore, the process model in Figure 1 can be abstracted using only three dimensions as shown in Figure 5. Under the new representation (each new dimension becomes a new event type in the HIN), it is still possible to compare and differentiate between process traces, such as traces of loan applications under review versus those that have already reached the inform decision step. Most importantly, the performance of `SeqPathSim` with the new dimensions will be improved due to the decrease in dimensionality.

Trace representation for dimension reduction: Before applying dimension reduction techniques to process traces, it is important to have an appropriate data representation for traces. The most common data representation for dimension reduction is based on the vector space model, in which each trace t is represented as a vector $t = (s_1, s_2, ..., s_{|\mathbb{E}|})$. The value of each dimension s_i is associated with a type of event $e_i \in \mathbb{E}$ and equals the frequency of the event e_i in the trace t: $s_i = f_{e_i,t}$. This representation, captures the "local" importance of each event type to a trace (via $f_{e_i,t}$), but does not capture the "specificity" of each event type across all the traces. Taking the process model in Figure 1 as an example, since the event *"Receive loan application"* appear in almost all traces (being an entry point to the process), it becomes less important as a differentiator between traces (*i.e.*, it has low specificity).

Motivated from TF IDF-based document representation in text mining [24], we propose a new data representation for process traces that captures both the local importance of each event and its specificity to a trace. In addition to a trace's event frequency, we consider the popularity of each event across all traces: $n_{e_i} = |\{t \in \mathbb{T}, e_i \in t\}|$. Intuitively, a large n_{e_i} implies that event e_i is popular and thus exhibits low specificity for a trace. As a result, the value of each dimension in a trace's vector s_i is based

on the combination of the trace's event frequency $f_{e_i,t}$ (representing local importance) and inverse event popularity (representing specificity). We propose a new calculation of s_i following a logarithmic formula as follows:

$$s_i = \begin{cases} (1 + log(f_{e_i,t})) \times log(\frac{|\mathbb{T}|}{n_{e_i}}) & \text{if } e_i \in t \\ 0 & \text{otherwise} \end{cases} \quad (7.4)$$

Having represented process traces as vectors, the set of input traces $\mathbb{T}$ can be represented as a large matrix $\mathbf{M}$ of size $|\mathbb{T}| \times |\mathbb{E}|$, where each element $\mathbf{M}_{ij}$ ($1 \leq i \leq |\mathbb{T}|, 1 \leq j \leq |\mathbb{E}|$) is the value associated with event type e_j in the i-th trace. Next we will show how to perform dimension reduction on the matrix $\mathbf{M}$ of input traces.

Process model-regularized trace dimension reduction: Given the matrix representation $\mathbf{M}$, we can apply off-the-shelf dimension reduction techniques [4] on $\mathbf{M}$, such as non-negative matrix factorization (NMF), principle component analysis (PCA), or singular value decomposition (SVD). The results of those techniques often include a matrix $\mathbf{M}'$ of size $|\mathbb{T}| \times \kappa$ (where $\kappa \ll |\mathbb{E}|$ is the number of new dimensions) that represents the original traces projected onto the new dimensions; and a matrix $\mathbf{W}$ of size $|\mathbb{E}| \times \kappa$ that represents the mapping from the old to the new dimensions (*i.e.*, each row in $\mathbf{W}$ is appropriate to a distribution vector of an event over the new dimensions).

The results of the existing techniques, however, cannot be used directly for an edit distance-based approach like `SeqPathSim`. Specifically, while `SeqPathSim` requires the input traces to be in the form of sequences of events in the new dimensions, the above results only give us the "soft" mappings from the input events to the new dimensions (*i.e.*, in the form of matrix $\mathbf{W}$). Therefore, there is a need to transform $\mathbf{W}$ into a "hard" assignment of the original events to the new dimensions. Formally, if we represent κ new dimensions as a set of κ clusters of events $\mathbb{C} = \{\mathcal{C}_i\}, 1 \leq i \leq \kappa$, then we need to derive a one-to-one mapping function $\rho : \mathbb{E} \rightarrow \mathbb{C}$ that maps each event $e \in \mathbb{E}$ to a cluster in $\mathbb{C}$. The objective of the mapping function ρ is to maximize the collective similarities between pairs of events that belong to the same cluster.

This problem can be represented as an optimization problem with the following objective function:

$$\arg\max_{\rho} \sum_{\rho(e_j)=\rho(e_k)} sim(e_j, e_k) \quad (7.5)$$

where $sim(e_j, e_k)$ is the similarity between e_j and e_k in the new dimensions and can be computed with existing measures such as Cosine similarity or Euclidean distance-based similarity.

Deriving the "hard" assignment solely based on the result of the existing dimension reduction techniques, however, ignores the important information about the relationships between events in the process model. A process execution model can be obtained by doing a projection of the process traces' heterogeneous graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ on the set of event nodes $\mathbb{E}$, denoted as $\mathbf{G}_{\mathbb{E}} = (\mathbf{V}_{\mathbb{E}}, \mathbf{E}_{\mathbb{E}})$. Because the process model captures the follow-up relationships between events (*i.e.*, the weights of edges in the process model represent the number of times an event follows another event in a trace), it provides strong indications of which events to assign to a cluster. For example, events that frequently follow each other are likely to be in the same cluster. Therefore, we propose adding another component, denoted as Δ, to the objective function in Equation 7.5 to account for the regularization based on the process model. Particularly, Δ is used to maximize the collective similarities between pairs of events that follow one or the other in the process execution model. The new objective function for finding an optimal mapping ρ is as follow:

$$\arg\max_{\rho}[(1-\lambda) \times \sum_{\rho(e_j)=\rho(e_k)} sim(e_j, e_k) + \lambda \times \Delta] \quad (7.6)$$

with:

$$\Delta = \sum_{(e_j,e_k) \in E_{\mathbb{E}}} w(e_j, e_k) \times sim(e_j, e_k)$$

where $w(e_j, e_k)$ is the weight of the edge between e_j and e_k in process model $\mathbf{G}_{\mathbb{E}}$, and λ is a user specified parameter to tune the preference between the statistical similarity on new the dimensions (the first component) and the regularization based on the process model (the second component).

The optimization problem in Equation 7.6 is a variant of the set partitioning problem and finding a feasible solution for such a problem is NP-hard. Therefore, we design a greedy algorithm: instead of optimizing the global objective defined in Equation 7.6, we propose a local objective function defined in Equation 7.7, where an event e_j is assigned to the cluster (*i.e.*, new dimension) that contains the event closest to e_j.

$$\rho(e_j) = \rho(e^*) \text{ w.r.t. } e^* = \arg\max_{e_k \in \mathbb{E}} sim'(e_j, e_k) \quad (7.7)$$

with:

$$sim'(e_j, e_k) = (1-\lambda) \times sim(e_j, e_k) + \lambda \times \sigma_{\mathcal{P}_{EE}}(e_j, e_k)$$

where $\sigma_{\mathcal{P}_{EE}}(e_j, e_k)$ is the `PathSim`-based similarity between e_j and e_k via meta-path EE, which is used to account for the sequential relationship between two events in the process model (*i.e.*, $\sigma_{\mathcal{P}_{EE}}(e_j, e_k)$ can be considered as the local regularization term, similar to the role of Δ in Equation 7.6); $sim'(e_j, e_k)$ is the new similarity measure between events that combines both statistical similarity (*i.e.*, $sim(e_j, e_k)$) and sequential similarity (*i.e.*, $\sigma_{\mathcal{P}_{EE}}(e_j, e_k)$).

Algorithm 1 Process Model-Regularized Traces Dimension Reduction

procedure GREEDYPROCESSDIMRED($\mathbf{W}$, κ, $\mathbb{E}$, λ)
  // Calculate $\sigma_{\mathcal{P}_{EE}}$
  for each pair $(e_j, e_k) \in \mathbb{E}$ **do**
    Calculate $\sigma_{\mathcal{P}_{EE}}(e_j, e_k)$ (by Definition 3.2)
  // Calculate pairwise similarities
  for each pair $(e_j, e_k) \in \mathbb{E}$ **do**
    $\mathcal{S}_{jk} = sim'(e_j, e_k)$ (by Equation 7.7)
  // Perform hierarchical clustering
  $\mathcal{H} =$ `hierarchical_clustering`$(\mathcal{S})$
  // Flatten the hierarchy $\mathcal{H}$
  $\mathbb{C} =$ `flatten_hierarchy`$(\mathcal{H}, \kappa)$
  return $\mathbb{C}$

Dataset	Traces	Events	Event types
BPIC'13	7554	65533	13
RECEIPT	1434	8577	27
BANK	2000	116839	113

Table 1: Datasets

Our greedy approximation algorithm outlined in Algorithm 1 uses a bottom-up strategy—similar to that of agglomerative clustering algorithms—to assign original events to clusters (*i.e.*, the new dimensions). First, `PathSim`-based similarity $\sigma_{\mathcal{P}_{EE}}$ and a similarity matrix $\mathcal{S}$ are calculated between all events in $\mathbb{E}$ using Equation 7.7. Next, the algorithm treats each event as a singleton cluster and attempts to successively merge, or agglomerate, pairs of events that are closest to each other until all clusters have been merged into a single cluster that contains all events. This step creates a hierarchy $\mathcal{H}$, where each leaf node is an event and the root is the single cluster of the last merge. The final step is to cut the hierarchy at some point to obtain the desirable number of clusters κ. While there are a number of criteria [24] that can be used to decide the cutting point on the hierarchy, we use a simple, but effective approach that is based on finding a minimum similarity threshold so that the distance between any two events in the same cluster is no more than the threshold, and no more than κ clusters are formed.

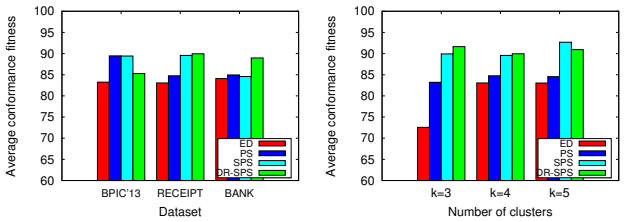
8 Experimental Evaluation

In this section, we demonstrate the efficacy and the efficiency of our methods using multiple real-world and synthetic datasets spanning different business process domains.[2] Our experiments were conducted on an Intel Core i7 machine with 16 GB of memory running Windows 7.

Datasets: Table 1 lists detailed properties about the datasets, which are publicly available and range from a relatively small to a large number of dimensions.

The BPIC'13 dataset consists of logs representing Volvo's IT incident and problem management process.

[2]All datasets are publically available at `https://data.3tu.nl/repository/collection:event_logs`



(a) Comparison with $k = 4$ (b) RECEIPT dataset

Figure 6: Conformance fitness analysis comparison

The RECEIPT dataset consists of logs representing the record of execution of the receiving phase of the building permit application process in an anonymous municipality.

The BANK dataset consists of synthetically generated logs that represent a large bank transaction process.

Evaluation Metrics: As described in Section 4, in this paper we evaluate the process trace clustering results using process-specific metrics: weighted average conformance fitness (`AvgFitness`), and weighted average structure complexity (`AvgComplexity`).

Given a clustering result, a process model is generated for each cluster using the Heuristic mining algorithm [19] and then converted to the Petri-Net model [18] for conformance analysis. Given the Petri-net model, we use two publicly available plugins from the ProM framework [22] for fitness and structural complexity analysis: The Conformance Checker Plugin is used to measure the fitness of the generated process models and the Petri-Net Complexity Analysis Plugin is used to analyze the structural complexity of the process models. After fitness and complexity scores are calculated for each cluster, the final scores are calculated as the average score over all clusters, weighted by clusters' sizes, as described in Section 4.

Trace Similarity: We evaluate the performance of the following approaches:

- `ED`: Our baseline approach is the context-aware edit distance-based clustering [1] where the costs of editing operations are derived from a trigram of consecutive events. We use the implementation of `ED` included in the ProM framework.
- `PS`: `PathSim`-based approach as described in Section 6.
- `SPS`: `SeqPathSim` algorithm as described in Section 6.
- `DR-SPS`: `SPS` with dimension reduction method as described in Section 7 with $\lambda = 0.6$.

Meta-paths: The meta-paths we use for trace-to-trace similarity in `PS` are TET, $TEET$, and $TERET$ with weights 0.2, 0.5, and 0.3 respectively (these weights are chosen based on user guidance and careful tuning).

For all approaches, hierarchical clustering [7] is used as the clustering algorithm in the final step, after the similarities

Metric	BPIC'13				BANK			
	ED	PS	SPS	DR-SPS	ED	PS	SPS	DR-SPS
Control Flows	19.42	18.53	**18.42**	20.01	**223.57**	262.56	282.85	271.90
AND Joins/Splits	4.85	**4.23**	5.09	6.90	**36.23**	38.65	42.65	37.53
XOR Joins/Splits	6.74	6.63	**6.01**	6.56	**72.21**	81.45	87.32	89.23

Table 2: Structural complexity comparison between approaches on BPIC'13 and BANK datasets with $k = 4$.

Metric	K=3				K=4				K=5			
	ED	PS	SPS	DR-SPS	ED	PS	SPS	DR-SPS	ED	PS	SPS	DR-SPS
Control Flows	166.84	134.61	20.00	**16.39**	101.06	86.65	21.84	**15.74**	100.16	80.77	20.54	**13.26**
AND Joins/Splits	15.89	1.71	1.86	**1.69**	6.27	7.14	1.92	**1.49**	6.25	1.57	1.84	**0.30**
XOR Joins/Splits	21.35	7.15	4.42	**2.11**	14.32	14.33	4.07	**2.32**	14.04	5.61	3.66	**1.43**

Table 3: Structural complexity comparison on RECEIPT dataset across different k

between every pair of traces have been calculated.

8.1 Conformance fitness comparison Figure 6a shows the weighted average conformance fitness results across different datasets where the number of clusters $k = 4$. Figure 6b shows the weighted average conformance fitness results for the RECEIPT dataset while varying the number of clusters k. Similar results were observed with the other datasets but were omitted due to space constraints.

The figures show that PS, the only non-edit distance approach in our evaluation, performs quite well compared with other edit distance-based approaches, and also clearly outperforms the baseline ED. This verifies the effectiveness of using PathSim similarity to capture the similarity between traces. DR-SPS, although being applied to the traces after dimension reduction, still performs well compared with SPS without dimension reduction (even better in some cases). This is interesting, considering the primary purpose of dimension reduction is to improve the efficiency not the effectiveness. This result can be explained by the fact that DR-SPS can intelligently group the traces that consist of different events but in the same new dimension (and thus highly correlated) into the same cluster. We evaluate the efficiency gained by dimension reduction in the following subsections.

All of our proposed approaches outperform the baseline edit distance-based approach ED. The results help confirm the effectiveness of modeling process traces as an HIN and using PathSim as the similarity measure between traces and events. It also helps verify the benefit of combining the process semantic relationships captured by PathSim with the sequential relationships captured by edit-distance in a unified measurement (*i.e.*, SPS and DR-SPS).

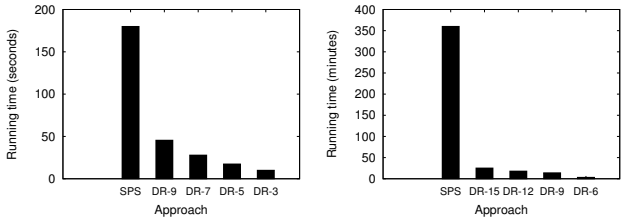
8.2 Structural complexity comparison Table 2 shows the weighted average complexity scores across different structural metrics (XOR joins/splits, AND joins/splits, and control flows) for BPIC'13 and BANK datasets, where the number of clusters $k = 4$. Table 3 shows similar results for the RECEIPT dataset with varying number of clusters k.

Overall, the results highlight that our proposed approaches generally outperform the baseline ED by producing clusters of simpler process models. The out-performance is particularly clear in the RECEIPT (across different number of clusters) and BPIC'13 datasets. For the BANK dataset, although ED produces clusters with the least complex process models, the difference is not significant and our proposed approaches outperform the baseline over other evaluation metrics (i.e., conformance fitness and efficiency).

8.3 Dimension reduction comparison In the next set of experiments, we focus on evaluating the effectiveness and efficiency of using DR-SPS and SPS. Recall that DR-SPS uses dimension reduction methods while SPS does not.

In terms of efficiency, Figure 7 shows the running time comparisons between DR-SPS (where DR-*n* denotes DR-SPS with *n* dimensions) and SPS on the RECEIPT and BANK datasets. As expected, DR-SPS significantly outperforms SPS with up to 9x speed-up on the RECEIPT dataset (which has 27 dimensions), and 100x speed-up on the BANK dataset (which has 113 dimensions).

In terms of effectiveness, Table 4 shows the fitness and structural complexity comparisons between DR-SPS (with varying numbers of dimensions) and SPS (whose results are displayed in parentheses). The results show that DR-SPS outperforms SPS in most of the cases (indicated in bold) in both fitness and structural complexity metrics. This result is interesting and somewhat surprising as it is generally expected that dimension reduction for edit distance-based approaches only benefits the efficiency at the expense of effectiveness. In fact, the results show that our DR-SPS approach with dimension reduction achieves both objectives.



(a) RECEIPT dataset (b) BANK dataset

Figure 7: Running time with and without dimension reduction.

Metric (SPS)	RECEIPT				Metric (SPS)	BANK			
	DR-3	DR-5	DR-7	DR-9		DR-6	DR-9	DR-12	DR-15
Fitness (89.56)	**91.98**	**89.95**	89.29	**90.82**	Fitness (84.58)	**88.97**	**86.57**	**87.12**	**85.52**
Control Flows (21.94)	200.28	**15.74**	**12.64**	**12.19**	Control Flows (282.86)	**271.90**	287.76	325.59	293.55
AND Joins/Splits (1.92)	21.97	**1.5**	**0.62**	**0.85**	AND Joins/Splits (42.65)	**37.53**	**41.39**	45.92	**42.54**
XOR Joins/Splits (4.07)	20.36	**2.33**	**1.95**	**1.38**	XOR Joins/Splits (87.32)	89.23	**83.15**	98.48	92.05

Table 4: Effectiveness of using dimension reduction on RECEIPT and BANK datasets with varying dimensions.

9 Conclusions

In this paper, we modeled process traces as a heterogeneous information network (HIN), and defined a similarity metric that not only uses the rich semantic relationship between nodes, but also the sequential execution of events in each trace. Our dimension reduction technique allows us to overcome the computational scalability drawbacks of edit-distance algorithms while remarkably still providing better quality results. A thorough experimental evaluation using real-world traces spanning multiple business process domains validate and show the efficacy of our approach.

Our on-going research is pursued along two dimensions. First, we intend to extend this approach to other domains, and in particular apply our algorithms on network logs to find clusters of anomalous behavior such as malicious attacks. Second, we seek to take advantage of the resources available in a cloud computing platform to provide business users an interactive tool they can use to explore and understand their processes. The challenge here is to facilitate going from a business problem to a solution without overwhelming users with too many choices or hard-to-interpret results. Taking the advantage of the performance gains and simplifications in mined processes, the ultimate goal is to have a tool that can be applied to a diverse set of datasets and quickly guide users to results that they can act on and then validate their decisions in a continuous manner.

References

[1] RP Jagadeesh Chandra Bose and Wil MP van der Aalst. Context aware trace clustering: Towards improving process mining results. In *SDM*, 2009.

[2] RP Jagadeesh Chandra Bose and Wil MP van der Aalst. Trace clustering based on conserved patterns: Towards achieving better process models. In *Business Process Management Workshops*,. Springer, 2010.

[3] Jochen De Weerdt, Jan Vanthienen, Bart Baesens, et al. Active trace clustering for improved process discovery. *Knowledge and Data Engineering, IEEE Transactions on*, 25(12):2708–2720, 2013.

[4] Imola K Fodor. A survey of dimension reduction techniques, 2002.

[5] Navarro, Gonzalo. A guided tour to approximate string matching. *ACM computing surveys* 2001.

[6] Levenshtein, Vladimir I. "Binary codes capable of correcting deletions, insertions, and reversals." *In Soviet physics doklady*, vol. 10, no. 8, pp. 707-710. 1966.

[7] Xu, Rui, and Donald Wunsch. Survey of clustering algorithms. *Neural Networks, IEEE Transactions* 2005.

[8] Greco, Giuseppe, et al. Discovering expressive process models by clustering log traces. *Knowledge and Data Engineering, IEEE Transactions on* 18.8 (2006): 1010-1027.

[9] Christian W Günther and Wil MP Van Der Aalst. Fuzzy mining–adaptive process simplification based on multi-perspective metrics. In *Business Process Management*, 2007.

[10] Anne Rozinat and Wil MP van der Aalst. Conformance checking of processes based on monitoring real behavior. *Information Systems*, 33(1):64–95, 2008.

[11] Minseok Song, Christian W Günther, and Wil MP Van der Aalst. Trace clustering in process mining. In *Business Process Management Workshops*, 2009.

[12] Song, Minseok, et al. A comparative study of dimensionality reduction techniques to enhance trace clustering performances. *Expert Systems with Applications* 40.9 (2013): 3722-3737.

[13] Yizhou Sun and Jiawei Han. Mining heterogeneous information networks: a structural analysis approach. *ACM SIGKDD Explorations Newsletter*, 14(2):20–28, 2013.

[14] Sun, Yizhou, et al. Pathsim: Meta path-based top-k similarity search in heterogeneous information networks. *Proceedings of the VLDB Endowment*, 4(7), 2011.

[15] Sun, Yizhou, et al. Integrating meta-path selection with user-guided object clustering in heterogeneous information networks. In *18th ACM SIGKDD*, 2012.

[16] Wil Van Der Aalst. *Process mining: discovery, conformance and enhancement of business processes*. Springer Science & Business Media, 2011.

[17] Wil Van der Aalst, Ton Weijters, and Laura Maruster. Workflow mining: Discovering process models from event logs. *Knowledge and Data Engineering, IEEE Transactions on*, 2004.

[18] Wil MP Van der Aalst. The application of petri nets to workflow management. *Journal of circuits, systems, and computers*, 8(01):21–66, 1998.

[19] AJMM Weijters, Wil MP van Der Aalst, and AK Alves De Medeiros. Process mining with the heuristics miner-algorithm. *Technische Universiteit Eindhoven, Tech. Rep. WP*, 166:1–34, 2006.

[20] Lijie Wen, Wil MP van der Aalst, Jianmin Wang, and Jiaguang Sun. Mining process models with non-free-choice constructs. *Data Mining and Knowledge Discovery*, 15(2):145–180, 2007.

[21] Stephen A White. Introduction to BPMN. *IBM Cooperation*, 2004.

[22] van Dongen, Boudewijn F., et al. The ProM framework: A new era in process mining tool support. In *Applications and Theory of Petri Nets*, 2005.

[23] De Medeiros, Ana Karla Alves, et al. Process mining based on clustering: A quest for precision. *Business Process Management Workshops*, 2008.

[24] Manning, Christopher D., Prabhakar Raghavan, and Hinrich Schtze. Introduction to information retrieval. Vol. 1. Cambridge: Cambridge university press, 2008.