information

*Article*

# Automated Trace Clustering Pipeline Synthesis in Process Mining

**Iuliana Malina Grigore [1], Gabriel Marques Tavares [2,3], Matheus Camilo da Silva [1], Paolo Ceravolo [4] and Sylvio Barbon Junior [1,*]**

[1] Dipartimento di Ingegneria e Architettura, Università Degli Studi di Trieste, 34127 Trieste, Italy; iulianamalina.grigore@phd.units.it (I.M.G.); matheus.camilodasilva@phd.units.it (M.C.d.S.)
[2] Chair of Database Systems and Data Mining, Ludwig-Maximilians-Universität München, 80538 Munich, Germany; tavares@dbs.ifi.lmu.de
[3] Munich Center for Machine Learning (MCML), 80539 Munich, Germany
[4] Dipartimento di Informatica, Università Degli Studi di Milano Statale, 20122 Milano, Italy; paolo.ceravolo@unimi.it
* Correspondence: sylvio.barbonjunior@units.it

**Abstract:** Business processes have undergone a significant transformation with the advent of the process-oriented view in organizations. The increasing complexity of business processes and the abundance of event data have driven the development and widespread adoption of process mining techniques. However, the size and noise of event logs pose challenges that require careful analysis. The inclusion of different sets of behaviors within the same business process further complicates data representation, highlighting the continued need for innovative solutions in the evolving field of process mining. Trace clustering is emerging as a solution to improve the interpretation of underlying business processes. Trace clustering offers benefits such as mitigating the impact of outliers, providing valuable insights, reducing data dimensionality, and serving as a preprocessing step in robust pipelines. However, designing an appropriate clustering pipeline can be challenging for non-experts due to the complexity of the process and the number of steps involved. For experts, it can be time-consuming and costly, requiring careful consideration of trade-offs. To address the challenge of pipeline creation, the paper proposes a genetic programming solution for trace clustering pipeline synthesis that optimizes a multi-objective function matching clustering and process quality metrics. The solution is applied to real event logs, and the results demonstrate improved performance in downstream tasks through the identification of sub-logs.

**Keywords:** AutoML; trace clustering; pipeline synthesis; process mining



**Citation:** Grigore, I.M.; Tavares, G.M.; Silva, M.C.d.; Ceravolo, P.; Barbon Junior, S. Automated Trace Clustering Pipeline Synthesis in Process Mining. *Information* **2024**, *15*, 241. https://doi.org/10.3390/info15040241

Academic Editors: Ivan Miguel Pires, Eftim Zdravevski and Petre Lameski

Received: 18 March 2024
Revised: 5 April 2024
Accepted: 18 April 2024
Published: 20 April 2024



## 1. Introduction

Business processes have been central to organizational operations since the emergence of process-oriented views of organizations. With the increase in event data and the intricate nature of business processes, companies need to gain a deeper understanding of how their processes operate in real-world situations. This need has led to the development and growing adoption of process mining techniques as a key catalyst. Process mining (PM), recognized as a significant innovation in business process management [1], is a new discipline that combines process model-driven approaches with data mining. It offers valuable insights into the actual operational dynamics of business processes. Process mining is a critical tool for organizations, bridging the gap between machine learning, data mining, and process modeling and analysis. It enables the extraction of valuable insights from event logs, helping to identify, monitor, and improve business processes [2].

PM works on two basic assumptions: a process model outlines the lifecycle of a single object (i.e., unit of analysis within a business process), and each event relates to exactly one object of a particular type [3]. However, real life event logs are challenging, due to the size

and noise of event logs, which require careful analysis. In addition, the different sets of behaviors within the same business process increase the variability of traces (i.e., a sequence of related events that are recorded in a log file or dataset), posing a data representation challenge. These challenges highlight the continued need for innovative solutions in the evolving field of process mining [4]. An event log is a collection of events that represents a view of a process. Each event has three essential characteristics: case ID, activity, and timestamp. The case ID helps to identify a particular event within the process and links related events. Meanwhile, the timestamp organizes events and helps to identify potential issues. An activity acts as a description or label, providing information about the specific activity or event. This may include the type of operation, the user involved, or the event's outcome. Other attributes, such as cost, resources, and location, can also be added. Trace refers to a sequence of related events that are recorded in a log file or dataset. Traces provide a chronological description of how a particular case or instance of a process unfolded over time.

Trace clustering is a method that aims to tackle the issue of noise present in event logs. It does so by grouping similar traces or sequences of events, which enables the identification of patterns and trends in the data. This simplifies the process of distinguishing meaningful activity from irrelevant or noisy events. Ultimately, trace clustering streamlines the analysis of the overall data.

Trace clustering is a significant aspect of process discovery as it helps in identifying recurrent sequences of activities or events that occur within a process. These sequences are consistent patterns that offer valuable insights into process dynamics. By isolating these patterns, organisations can analyze, optimize, and improve their processes, leading to greater efficiency and compliance. Hence, trace clustering is an essential process mining tool that helps to unearth hidden patterns and derive actionable insights from operational data. The resulting clusters delineate different pathways or process variants, providing the basis for a more profound understanding of process behavior [5].

However, creating a trace clustering solution, called a *pipeline*, requires careful consideration of various steps, including data preprocessing, feature extraction, similarity measurement, clustering algorithm selection, parameter tuning, and cluster evaluation, as mentioned in [4]. Several approaches have been explored in other fields to overcome the issue of pipeline creation. The process of creating data mining pipelines capable of inferring and/or predicting dataset patterns is a complex effort that requires expertise not only in the domain of the problem related to the data [6] but also in the data mining field [7].

A recent solution is pipeline synthesis, which is one of the focuses of the area known as automated machine learning (AutoML) [8]. It aims to develop techniques that can make decisions about pipeline conception in a data-driven, objective, and automated manner—the user simply provides data, and the AutoML system automatically determines the approach that performs best for a particular application [9].

Specifically for trace clustering, researchers in the field of process mining continue to work on addressing effectively handling process variants and concurrency, especially in scenarios where multiple paths or activities can occur simultaneously. Choosing appropriate parameters for clustering algorithms (e.g., distance metrics, number of clusters) can be non-trivial and may require domain expertise. Suboptimal parameter settings can lead to poor clustering quality.

Considering the complexity of designing trace clustering pipelines and the limitations of current approaches, this paper proposes a solution based on genetic programming for trace clustering pipeline synthesis. The solution was tested on four event logs benchmarks, and the results indicate its viability in balancing several indicators. Moreover, an evaluation is provided touching on the relationship between pipeline steps. Finally, the implementation provides a pipeline prototype that can be easily applied by practitioners. The main contributions of this work are:

- A new method for trace clustering pipeline synthesis based on AutoML.

- We present a state-of-the-art review of trace clustering and correlate it with the theoretical basic aspects of PM.
- We introduce the usage of a multi-objective function for trace clustering, based on combining clustering and process discovery metrics to improve clustering quality.

The remainder of this paper is organized as follows: Section 2 discusses research connected with the trace clustering problem. Section 3 presents the theoretical basis for this work, focusing on PM notations and the trace clustering pipeline synthesis. Section 4 follows by presenting the proposed approach for automated pipeline design for clustering event logs. Section 5 presents the results of the experiments developed from the perspectives of PM and data mining. Section 6 discusses the results and presents the current limitations of the solution and the experimental setup. Finally, Section 7 leaves the final remarks and indicates possible future research directions.

## 2. Related Work

The problem of clustering event logs has been discussed at length in the PM literature. In this review, we focus on trace clustering methods, the main goal of which is finding groups of behaviors within a business process [10] emphasizing the pipeline complexity, i.e., from encoding to algorithm selection and hyperparameter tuning. It follows that the identification of sublogs leads to a better understanding of the process for stakeholders and improves output quality for downstream tasks, e.g., process discovery by providing simpler and more comprehensible models [11].

An initial step for clustering traces is deciding the representational scheme. Greco et al. [12] used n-grams to map event data to feature vectors, whereas Song et al. [13] emphasized the importance of trace clustering in identifying process variants in diverse environments. They proposed a generic approach that effectively addresses diversity-related issues by breaking down the log into smaller, more homogeneous subsets of traces. They used trace profiles that characterize cases based on specific features extracted from the log to achieve this. These profiles play a crucial role in assessing the similarity of points to be clustered, providing a systematic method for analyzing process instances and extracting valuable insights for process optimization.

Bose et al. [11] used the string edit distance between traces (represented as strings) to measure their distance. To adjust to the PM domain, the authors propose particular weights depending on the operation type. Bui et al. [14] proposed a new trace clustering algorithm that utilizes trace context. The algorithm comprises two phases: Determining trace context and building clustering. One of the major benefits of this algorithm is its ability to work directly with the trace data without the need for converting them into an intermediate representation. Another significant advantage is that it can automatically detect the optimal number of clusters, which reduces the complexity compared to traditional clustering techniques.

Boltenhagen et al. [15] proposed an improved version of the trace clustering method that enables cluster centroids to have a more complex structure. This enhancement is beneficial when dealing with concurrency and loop constructs in process models. By incorporating concurrency and loop considerations, the clustering method becomes more robust and efficient in analyzing event logs. The outcome is a more abstract representation of trace variants, which simplifies interpretation and reduces redundancy within clusters.

Jablonski et al. [16] introduced a new approach to trace clustering that aims to enhance the quality and conformity of processes. The approach suggests that considering different perspectives is crucial for this purpose. It provides a comprehensive definition of similarity between traces by combining information about performed activities, utilized resources, and data values. However, calculating the optimal weights for this computation is a costly process.

Clustering event logs can be seen as a preprocessing step to identify sublogs, thus improving the quality of downstream tasks, such as process discovery [17]. Other common encoding methods applied are bags of activities [18], dependency spaces [19], and log

footprints [10], for instance. In terms of clustering, a plethora of different methods were already applied, e.g., *k*-means [12,20], hierarchical clustering [11,20], spectral clustering [19], and constrained clustering [21]. Moreover, clustering algorithms are known for being very sensitive to hyperparameter setting, by virtue of this, one should consider that these methods require a significant effort towards correctly adjusting the setup.

The complexity of defining a trace clustering pipeline is huge, which hampers its application in many scenarios. Stakeholders often have the process knowledge; however, specialists might also be overwhelmed by the overabundance of options, leading to a trial-and-error approach where optimization dimensions are not well defined. Given the complexity of defining clustering steps, Tavares et al. [4] proposed a method based on meta-learning to pipelines considering event log behavior as a guide. The authors propose a framework that extracts meta-features (i.e., descriptors) to quantify the behavior of an event log. Then, the defined pipeline space is applied to the event log collection with the aim of finding the best configuration (meta-target). In this way, finding an optimal setup is transformed into a classification problem. A classifier learns the mapping between meta-features and meta-targets and, given a new event log, its meta-features are retrieved and the classifier recommends a pipeline. The approach is the first in the direction of optimizing trace clustering pipelines. However, it falls short in several aspects. First, the pipeline space is highly limited because the computational cost of applying all possible pipelines to the log collection is huge. Moreover, by transforming the pipeline synthetization into a classification problem, the recommendation quality is limited by the classifier performance, which is further impacted in imbalanced scenarios. Finally, the step to choose the best solution is based on positional ranking, which cannot reflect well the distances to the optimal metrics.

Pipeline synthesis can be defined as a process of optimizing elements (i.e., steps) in the combined algorithms and hyperparameters search space. Therefore, it is a difficult problem to model since the search space is highly dimensional and the steps may have hierarchical dependencies, such as hyperparameters that only harmonize if combined with a specific algorithm [22]. In general, AutoML solutions can be divided into two groups. The first group uses local search approaches: starting from a fixed structure of pipeline steps, the search space is divided by first performing a selection of clustering algorithms (or data preprocessors) and then optimizing their parameters [22,23]. In turn, the second group explores the global search approach: it represents pipeline components as data structures and then applies evolutionary computation on the structures to obtain the most optimized dynamic pipeline possible [6,24].

A solution for automated design of trace clustering pipelines based on meta-learning was proposed by Tavares et al. [4]. The authors presented a method to provide pipeline recommendations based on event log behavior. For that, descriptors are extracted and a ranking function defines an optimal pipeline (considering the available search space). The problem is then converted into a classification task, with a traditional predictive model built based on the relationship between event log behavior and optimal solutions. However, the approach falls short as it relies on a considerable collection of event logs to provide good performing recommendations. Moreover, the computational cost is high as every possible pipeline needs to be tested against all event logs.

Therefore, considering the complexity of designing decent trace clustering pipelines and the limitations of automated solutions, we propose a method to find the optimal clustering design, from the encoding algorithm to a particular algorithm and its hyperparameters, based only on a given event log. In this manner, the automation of trace clustering not only provides a readily available prototype but also promotes interpretability and helps conserve resources.

## 3. Preliminaries

This section presents the main foundations for this research, mainly concepts regarding *PM* and *AutoML*. With regard to the definition of traditional event logs, we present

some universes in Definition 1 [25] and then the definition of traditional event logs in Definition 2 [25].

**Definition 1** (**Universe**). *$U_{ev}$ is the universe of events, $U_{act}$ is the universe of activities, $U_{case}$ is the universe of cases, $U_{time}$ is the universe of timestamps, $U_{att} = \{act, case, time, \dots\}$ is the universe of attributes, $U_{val}$ is the universe of values, and $U_{map} = U_{att} \not\rightarrow U_{val}$ is the universe of attribute-value mappings. We assume that $U_{act} \cup U_{case} \cup U_{time} \subseteq U_{val}$, $\perp \notin U_{val}$, and for any $f \in U_{map} : f(act) \in U_{act} \cup \{\perp\}$, f(case) $\in U_{case} \cup \{\perp\}$, and f(time) $\in U_{time} \cup \{\perp\}$.*

**Definition 2** (**Event Log**). *An event log is a tuple $L = (E, \#, \prec)$ consisting of a set of events $E \subseteq U_{ev}$, a mapping $\# \in E \rightarrow U_{map}$, and a strict partial ordering $\prec \subseteq ExE$ on events.*
*For any $e \in E$ and $att \in dom(\#(e))$: $\#_{att}(e) = \#(e)(att)$ is the value of attribute att for event e. For example, $\#\_act(e)$,$\#\_case(e)$, and $\#_{time}(e)$ are the activity, case, and timestamp for an event e. The ordering of events respects time, i.e., if $e_i, e_2 \in E$, $\#_{time}(e_1) \neq \perp$, $\#_{time}(e_2) \neq \perp$, and $\#_{time}(e_1) < \#_{time}(e_2)$, then $e_2 \not\prec e_1$.*

To work with trace clustering, we present the definition of the trace in Definition 3 [25], and then the definition of trace clustering in Definition 4 [15].

**Definition 3** (**Trace**). *A trace $\sigma = < a_1, a_2, \dots, a_n > \in U^*_{act}$ is a sequence of events, with the constraint that, for all $0 < i < j \leq m, time(e_i) \leq time(e_j)$. $L(\sigma)$ is the number of times trace $\sigma$ appears in the event log L.*

**Definition 4** (**Trace Clustering**). *Given a log L, a trace clustering over L is a partition over a (possible proper) subset of the traces in L.*

As in most PM pipelines, trace clustering techniques ingest event logs. Within the scope of our study, we aim to extract the traces from an event log and map them into a numerical feature space by encoding their behavior. Therefore, a function that projects event data into another feature space is required.

**Definition 5** (**Encoding** [26]). *Assuming an event log L, encoding is a function $f_e$ that maps L to a feature space, i.e., $f_e : L \rightarrow \mathcal{R}^n$, where $\mathcal{R}^n$ is an n-dimensional real vector space.*

Encoding is the first step for trace clustering which should then be combined with subsequent steps to form a pipeline.

**Definition 6** (**Trace clustering pipeline synthesis** [9]). *Let $f_e \in \mathcal{F}_e$ be an encoding function in an encoding space, let $\phi = \{\rho 1, \rho 2, \dots, \rho_n\}$ be the space containing all different preprocessors, such that $\forall \rho_1, \rho_2 \in \phi, \rho_1 \circ \rho_2 \in \phi$, let $\mathcal{C} = \{\mathcal{C}_1, \mathcal{C}_2, \dots, \mathcal{C}_n\}$ be a set of clustering algorithms, and let the hyperparameters of each clustering algorithm $C^i$ have a domain $\theta^i \in \Theta$. A trace clustering pipeline is defined by the tuple $({f_e}^i \in \mathcal{F}_e, \rho^i \in \phi, c^i \in \mathcal{C}, \theta^i \in \Theta)$. The trace clustering pipeline synthesis is the problem of finding a pipeline $p \in \mathcal{P}$ that maximizes the performance of an objective function $\pi^*$.*

Hence, automating the design of a trace clustering pipelines involves optimizing a scoring function, typically composed of relevant metrics of interest.

## 4. Materials and Methods

This work presents a novel method that leverages the application of genetic programming (GP) techniques to synthesize trace clustering pipelines specifically tailored for PM based on AutoML. The proposed method employs an evolutionary algorithm to automatically generate and optimize clustering pipelines by evolving a population of potential solutions, as depicted in Figure 1. It combines the optimization power of evolutionary computation with the domain knowledge of PM to generate effective trace clustering pipelines tailored to event log datasets. In this section, we provide a detailed description

of the methodology, its implementation, and the experimental results demonstrating its effectiveness in enhancing quality metrics.

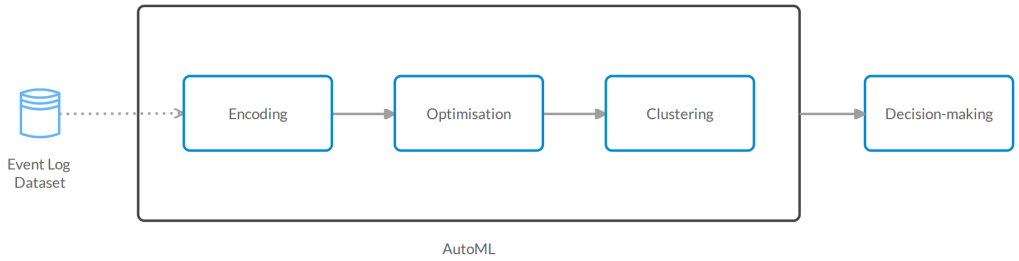


**Figure 1.** Overview of the proposed trace clustering approach.

The synthesis of pipelines for event log data proposed by this work is based on the representation of a trace clustering pipeline's steps as nodes in a tree data structure. The sequences of combinations between different encodings, preprocessing techniques, clustering algorithms, and their hyperparameters constitute the individuals of a diverse population [6]. The optimization process of the proposed method starts with the generation of an initial population size of 25 individuals. This population is iterated for 10 generations with a mutation rate of 0.9 and a crossover rate of 0.1. The initial population is created by randomly generating pipelines, which are then evaluated for performance using a multi-objective (MO) function. Once the quality of the pipelines is assessed, the optimization process continues by generating a new population based on the fittest pipelines, where the best combinations of operators according to the MO function are kept and perpetuated to new individuals through GP techniques, such as crossover and mutation. This process is repeated until a predefined number of generations is optimized and the fittest individual is selected [27]. We expanded mainly on the functionality of the Tree-based Optimization Tool (TPOT) [28], which has an interesting mechanism that also takes into account the length (number of steps) of the pipelines in the optimization process. This assures that the trace clustering pipelines generated by our proposed method not only have a good performance by the MO function but also a simple and understandable pipeline.

### *4.1. Dataset*

We used four real publicly available datasets already employed widely in the literature. BPI12 (https://data.4tu.nl/articles/dataset/BPI_Challenge_2012/12689204, accessed on 17 March 2024) represents a loan application process from a Dutch bank. As BPI12 is a huge event log, we sampled 15% of all events to facilitate computation. Moreover, the goal is to demonstrate the approach's ability to identify behaviors within the log; therefore, we do not envision the sampling as an issue. The next two event logs (BPI13CP (https://data.4tu.nl/datasets/1987a2a6-9f5b-4b14-8d26-ab7056b17929, accessed on 17 March 2024) and BPI13I (https://data.4tu.nl/datasets/0fc5c579-e544-4fab-9143-fab1f5192432, accessed on 17 March 2024)) report the problem management for incidents in a vehicle manufacturer organization. Finally, we included the Helpdesk (https://data.4tu.nl/datasets/94ee26c8-78f6-4387-b32b-f028f2103a2c, accessed on 17 March 2024) event log picturing a process for the ticket management of an Italian software company.

As presented in Table 1, the dataset collection comprehends processes from several different domains. Moreover, the characteristics of the logs vary considerably, e.g., the ratio of variants to several cases ranges from 0.05 to 0.45. We also highlight good variation in several activities and trace lengths. We hypothesize that by having a diverse collection, the experiments force the proposed approach to deal with very different scenarios, measuring its generality.

**Table 1.** Statistics (number of cases, number of events, number of activities, trace length, number of variants) describing the used event logs (Logs).

| Logs | #Cases | #Events | #Activities | Trace Length | #Variants |
|---|---|---|---|---|---|
| BPI12 | 1854 | 39,330 | 24 | 3–130 | 834 |
| BPI13CP | 1487 | 6660 | 4 | 1–35 | 183 |
| BPI13I | 7554 | 65,533 | 4 | 1–123 | 1551 |
| Helpdesk | 4580 | 21,348 | 14 | 2–15 | 226 |

### *4.2. Encoding*

In the context of this work, we modified TPOT to cater to the clustering task in process mining (PM) datasets. Since these data have unique characteristics, we added an encoding functionality to the pipeline's search space. These encoding methods capture various aspects of how event log data are represented. They help in discovering optimal clustering pipelines that are specific to PM datasets. This involves converting sequential data into a numerical format that can be processed and analyzed in subsequent pipeline steps. The encoding technique that maps traces from the event log space into a numerical feature space is a crucial element in the trace clustering pipeline. This numerical representation can then be integrated with downstream data mining algorithms, as discussed in previous work by [29–31]. A recent survey on several encoding techniques from different natures and families demonstrated that there is no unique optimal encoding method for any scenario [32]. Moreover, depending on which dimensions one is interested in optimizing, distinct methods can be considered. This issue is aggravated in environments with multiple objectives, revealing the importance of trading-off between dimensions. Considering the need to provide a robust and diverse algorithm space, we employed four encoding techniques from various families: one-hot, alignment, word2vec, and node2vec. It is important to note that the implemented proposal can have an expanded repertoire of encoding techniques, aligning with the design pattern outlined in the repository.

One-hot encoding is the most common technique used in PM. It works by transforming a variable (with $n$ different values) into an array representing a binary vector with the $i$-th position set to one when a value appears [33]. For event data, one-hot encodes activities as categories where each trace has a corresponding feature vector based on the occurrence and positions of activities in that trace. The size of the feature vector is equal to the vocabulary size of the business process. The alignments are a conformance checking technique that aims at comparing behavior between a model and an event log [34]. More specifically, it aims at directly relating traces to valid execution sequences allowed by the model and, with that, producing several features, such as the associated cost of moves and a fitness value, which are used as feature vector representing traces. Word embeddings have also been used to encode event data [29,31]. We choose word2vec considering its wide use across studies. Word2vec builds upon the concept that words appearing in similar contexts generate similar encodings. For that, it trains a neural network to reconstruct the linguistic context in a document. The embedding comes from the weights of the neural network. For event data, the trace representation comes from the aggregation of each activity's embeddings. Finally, as processes can also be represented as graphs, we employed node2vec [35] aiming to explore the relationships between nodes and their neighborhoods. The node representations come from second-order random walks focusing on a trade-off between breadth and width searches. As in word2vec, trace representation comes from aggregation of activities representations, which, in this scenario, are the nodes.

### *4.3. Clustering*

In addition to the encoding methods afforded mentioned, the optimization search space of the proposed method is composed of multiple preprocess methods, clustering algorithms, and hyperparameters that are implemented in the machine learning library *sklearn* [36]. The preprocessers present in the search space perform a range of different

data transformations which can reveal different aspects of the dataset and help the pipeline to find the clustering groups. They are the following: *MinMaxScaler*, *Normalizer*, and *StandardScaler*. Additionally, there is also the feature selector *VarianceThreshold* and feature decomposers *PCA* and *FastICA* that perform a similar function in the pipeline. Finally, the unsupervised algorithms that compose the search space cover the most relevant types of clustering for the domain problem of this work; this set of algorithms is intended to provide the proposed optimization method with a diversity of options and characteristics for the creation of pipelines [37]. They include the hierarchical clustering algorithm *AgglomerativeClustering*, the density-based clustering algorithm *DBSCAN*, the centroid-based algorithms *MiniBatchKMeans*, and the graph-based *SpectralClustering*. The hyperparameters of the search space are dependent on the clustering algorithms, where not every hyperparameter is available to every clustering algorithm.

### *4.4. Optimization*

The evolutionary process is guided by an MO optimization function that integrates two important metrics: the widely used *silhouette coefficient* and a PM metric known as *the sequence entropy*. By considering both metrics, our proposed method aims to optimize the clustering pipelines based not only on the quality of the resulting clusters but also on their conformity to PM requirements, such as model quality and log complexity.

The silhouette coefficient is a non-supervised metric capable of evaluating the quality of the spatial distribution of clustering results [38]. It provides a measure of how far each data point in a cluster is away from other clusters in the dataset. The coefficient ranges from $-1$ to 1, where values closer to 1 indicate that most of the data points are well clustered, with a significantly higher distance from neighboring clusters in comparison to their cluster. A value near 0 indicates that the data points are in a similar distance between their cluster and another cluster. Finally, a negative value suggests that the data points might have been assigned to the wrong cluster, as their distance to neighboring clusters is smaller than their distance to their cluster. By calculating the silhouette coefficient for each data point and then averaging them across the entire dataset, we can assess the overall quality of a clustering solution.

Sequence entropy is an extension of the entropy metric for event logs that takes into account the sequence of events in a log [39]. It calculates the entropy of the event sequence, considering the relative frequency of different sequences in the log. The sequence with the highest frequency has a higher proportion and contributes more to the calculated entropy. The sequence entropy metric is used to assess the variability and complexity of the event sequence in a process. The metric ranges from 0 to 1, where high sequence entropy indicates greater diversity of sequences and lower predictability in process execution, while low sequence entropy suggests greater regularity and predictability in the event sequences.

Since the metrics proposed for the pipeline optimization have different ranges, our MO function consists of the mean average of the normalized metrics of every individual pipeline in the evaluated generation. Furthermore, we inverted the values of the silhouette coefficient, i.e., the optimization process minimizes the MO function, where pipelines with an MO score closer to 0 have a good fit.

### *4.5. Decision-Making*

The complexity of business processes often results in a significant volume of events in logs, leading to numerous relationships and introducing significant noise. Trace clustering is a valuable tool in mitigating this complexity, but the existence of different behaviors in event logs adds an additional layer of complexity.

Our primary goal is to use a trace clustering pipeline to gain insights into the behavior of event logs, which can support various tasks related to business processes. The challenge of noise in event logs comes from irrelevant or extraneous information that can obscure meaningful patterns or insights within the data. Event logs, which document activities or

events, are critical for monitoring and troubleshooting. However, noise from a variety of sources can prevent effective analysis of these logs.

Mathematically, our goal is to generate an optimal clustering solution, denoted $\pi^*$. Here, $P$ represents a pipeline consisting of a coding function $f_e$, a preprocessor $\rho$, a clustering algorithm $c$, and its hyperparameters $\theta$, which together form the set of steps that yields the pipeline with the lowest average mean of its silhouette coefficient and sequence entropy over the entire set of pipelines generated during the optimization process for an event log $L$. The equation represents this optimization process (1):

$$\pi^* = \underset{\pi \in \text{Comb}(f_e^*, \rho^*, \mathcal{C}^*, \theta^*)}{\arg\min} P\left(\left\{ \underset{f_e^i \in \mathcal{F}_e, \rho^i \in \phi, c^i \in \mathcal{C}, \theta^i \in \Theta}{\arg\min} \left\{ \text{Mean}(S, E) \mid j = 1, \ldots, p \right\} \right\}\right) \quad (1)$$

Here, $\text{Comb}(f_e^*, \rho^*, \mathcal{C}^*, \theta^*)$ represents the set of possible combinations of the optimal encoding function, preprocessor, clustering algorithm, and their respective hyperparameters. The optimization involves minimizing the mean of the silhouette coefficient ($S$) and sequence entropy ($E$) over a range of pipelines generated from different combinations of functions and parameters.

The silhouette coefficient ($S$) and sequence entropy ($E$) are defined as follows:

$$S = \text{sil}_j(f_e^i, \rho^i, c^i, \theta^i(L)) \; E \quad = \text{entropy}_j(f_e^i, \rho^i, c^i, \theta^i(L)) \quad (2)$$

These equations express the silhouette coefficient and sequence entropy calculated for each pipeline $j$ using the corresponding encoding function $f_e^i$, preprocessor $\rho^i$, clustering algorithm $c^i$, and their hyperparameters $\theta^i$ applied to the event log $L$.

## 5. Results

This section presents a discussion based on the results collected. We divide the discussion into two complementary perspectives: process mining (PM) and data mining. From one side, we illustrate how stakeholders can benefit from easily identifying groups of behaviors in an event log. Then, we look at the contribution of designing complex pipelines using genetic programming (GP).

The experiments were conducted on a high-end workstation, a Dell Precision 5820 Tower X-Series, with the following hardware specifications:

- **CPU:** Intel Core i9-10980XE featuring 18 cores, 36 threads, and a maximum clock speed of 4.6 GHz.
- **Memory:** 62.5 GiB of RAM.
- **Storage:** Two SK Hynix PC801 NVMe 1TB drives.
- **GPU:** NVIDIA Quadro T1000 Mobile (TU117GLM).
- **Operating System:** Ubuntu 22.04.2 LTS (Jammy Jellyfish)

The implementation is available for replication purposes (https://github.com/Meta-Group/tpot_pm, accessed on 17 March 2024).

### *5.1. Process Mining Perspective*

We first report results connected with PM to demonstrate the viability of our method. Table 2 presents several process quality measurements before and after the clustering process. For this experiment, we computed the original log fitness, precision, and entropy. Then, the log is submitted to the proposed methodology to find a suitable pipeline for trace clustering. Then, we measure the same metrics and compare the performance improvement or decline. Note that for the clustered metrics, the reported value is the weighted average of all discovered clusters.

It is noticeable that the original event logs already had good fitness performance with three of them reaching over 0.9, and two with almost perfect fitness. When comparing with clustered event logs, two different behaviors arise. First, for BPI12 and BPI13CP, the obtained sublogs had a small fitness variation, remaining similar to the complete logs. For

BPI13I, a significant decrease in fitness value is observed. The log is characterized by having lengthy traces (reaching up to 123 events) with a limited vocabulary (four activities), which makes the identification of sublogs complex since most behavior is similar. For Helpdesk, we envision the opposite phenomenon, i.e., a high increase in fitness values. This behavior is explained by the number of discovered clusters being the same as the number of variants. That is, the trace clustering pipeline identified the event log variants and grouped traces based on which variant they belong to. Therefore, the fitness increases as clusters are more concise, i.e., they have less variability. Regarding precision, BPI12 originally had a very low performance, which was increased substantially by the clustered log. For BPI13CP and Helpdesk, the clustered log presented a very controlled variation when compared with the entire log. Finally, BPI13I, as with fitness, also presented a considerable performance decrease for this metric. Given the clear performance drop for BPI13I, the synthesized pipeline did not favor the quality of generated process models representing the cluster behaviors. On the contrary, for BPI12 and Helpdesk, the discovered pipeline was found to show an increase in the model quality metrics.

**Table 2.** Performance metrics before and after clustering. The sublogs identified by our method are always simpler (less entropic) than the originals. The percentages describe the percentage increase (green) or decrease (red) in comparison with the original event log.

| Log | BPI12 | BPI13CP | BPI13I | Helpdesk |
|---|---|---|---|---|
| Fitness | 0.93 | 0.99 | 0.99 | 0.73 |
| Precision | 0.37 | 0.93 | 0.94 | 0.94 |
| Entropy | 0.44 | 0.31 | 0.40 | 0.25 |
| Number of Clusters | 98 | 25 | 92 | 226 |
| Clustered Fitness (I) | 0.95 (2%) | 0.96 (−3%) | 0.68 (−30%) | 0.99 (26%) |
| Clustered Precision (I) | 0.66 (29%) | 0.86 (−6%) | 0.55 (−39%) | 0.95 (1%) |
| Clustered Entropy (I) | 0.18 (26%) | 0.15 (16%) | 0.14 (27%) | 0.02 (24%) |

When analyzing entropy, the improvement over the original logs is evident (ranging from 16% to 27%). This result indicates that the identified sublogs are inherently coherent. Therefore, the complexity downgrade can enable a better understanding of how an event log is composed of different behaviors. Although smaller event logs tend to have a lower complexity (meaning that sublogs have this tendency), clustering traces do not necessarily lead to an entropy decrease, i.e., random trace clustering would not be consistent with entropy decline. Except for Helpdesk, we observe that the number of discovered clusters is not directly correlated with the number of variants of a given log. Considering the number of variants of the original event logs, we believe the number of clusters discovered is healthy as the pipelines were able to group similar behaviors without limiting clusters to have only a unique variant.

Overall, the results indicate a possible trade-off between the model quality metrics (fitness and precision) and the log complexity (entropy), which is the case for BPI13CP and BPI13I. For both BPI12 and Helpdesk, the clustered event logs improved all metrics concomitantly.

For the next experiment, we aimed to provide a visual representation of the clustered space. For that, we sampled 10% of the Helpdesk event log to improve readability. The synthesized pipeline discovered was composed of the alignments encodings, the *Standard-Scaler* preprocessor, and *MiniBatchKmeans* as the cluster with $k$ as 4. Figure 2 shows the clustered space after dimensionality reduction was applied with PCA. Since most distances were respected (high explained variance), we can rely on the visual representation of traces and clusters. Although the border points cluster association is debatable, the clusters are concise and well separated, as indicated by the obtained silhouette score (0.79). The entropy was also diminished from 0.22 to 0.09. To strengthen the argument of complexity decrease, we also consider the number of nodes and edges of the leveraged directly follow graph. The original log contained 12 nodes and 30 edges. The respective number of nodes and edges for clusters 0, 1, 2, 3 are: 8 and 18, 2 and 1, 10 and 23, 5 and 7. The distribution

of the graph objects shows that each cluster contains different behavior, also highlighted by the number of nodes, which reflect the number of activities. Therefore, this example demonstrates how the complexity (measured from several perspectives) was alleviated by our proposal of automated pipeline synthesis.

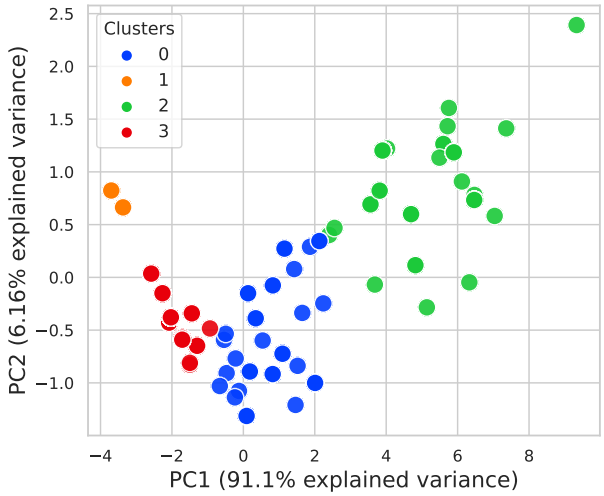


**Figure 2.** Clustered space for a sample of Helpdesk. The synthesized pipeline identified 4 clusters representing different trace behaviors.

The associations between pipeline steps are also worth investigating as the encoding method determines the output quality of subsequent tasks [32]. For this, we submitted each event log ten times to our method. Figure 3 reports the frequency of how each encoding method is combined with a clustering algorithm. Notably, there is a good balance between encoding methods (except for one-hot, which never appears in the final pipeline). Alignments appear more frequently associated with *MiniBatchKMeans* and *DBSCAN*, with the same following for node2vec. Word2vec presented the most heterogeneous behavior, being combined with all possible clusters. Regarding clustering solutions, *MiniBatchKMeans* and *DBSCAN* appear very often, while *SpectralClustering* and *AgglomerativeClustering* are infrequent, demonstrating that the former group matches better the distributions of our event log collection. To generalize this conclusion for the event log space, more logs should be included in the experiments.

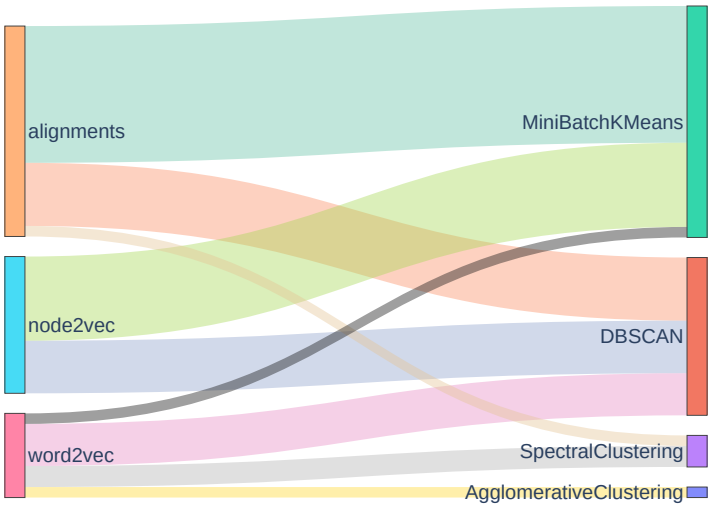


**Figure 3.** Sankey visualization of associations between encoding techniques and clustering algorithms. Each color represents a solution obtained by the combination of a given encoding method and a particular algorithm. The thickness of colored regions refers to the frequency of a combined pair.

### *5.2. Data Mining Perspective*

We present a comprehensive evaluation of the proposed method from a data mining perspective. The evaluation compares the performance of our method against a random search approach, both operating within the same search space and event logs. Random search is a simple and intuitive method used in optimization and hyperparameter tuning. In the context of this work, it involves randomly sampling and combining the steps of clustering pipelines from the defined search space. Random search is a good competitor for optimization tests due to its simplicity, exploration capability, baseline performance, robustness to noise, computational efficiency, and its usefulness in benchmarking and comparative analysis. We analyze and interpret the results obtained by comparing the runs from random search and the proposed method, focusing on key metrics, such as the silhouette coefficient, the sequence entropy, and the execution time. Through this analysis, we aim to provide insights into the effectiveness and efficiency of our method in synthesizing trace clustering pipelines.

The experimentation involved running the random search and proposed approach ten times for each of the four datasets (totaling 40 runs). Each run had a budget of 250 pipelines, resulting in a total of 2500 pipelines generated for each approach per dataset. Overall, a total of 10,000 pipelines were generated and evaluated throughout the experimentation process. For the evaluation, we ranked the best performing ones from the random search, selecting the pipeline with the best performance for each of its 40 runs.

The performance metric chosen was the Euclidean distance between a point representing the optimal values of the silhouette coefficient and the sequence entropy (1 and 0, respectively) and the values obtained by each pipeline. The aim was to assess how close the generated pipelines were to the ideal values. As shown in Figure 4, the results of the evaluation indicated that the proposed method outperformed the random search algorithm in terms of pipeline performance. The pipelines generated by the proposed method exhibited mostly smaller or at least similar Euclidean distances to the ideal values in comparison to random search. Additionally, the evaluation considered the time required for pipeline generation. The proposed method was less time-consuming compared to the random search algorithm. This indicates that the proposed method achieved superior performance while also being more efficient in terms of computational resources.

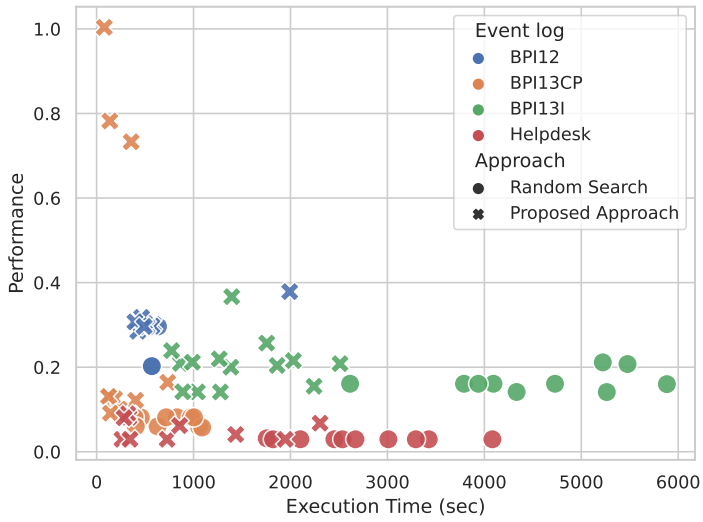


**Figure 4.** Visualization of the performance per execution time of random search and the proposed method.

Overall, the experiment showcased the advantage of the proposed method in terms of reduced computational time, making it a promising approach for generating trace clustering pipelines.

## 6. Discussion and Limitations

The experiments aimed to capture several complimentary qualitative aspects of clustered event logs. From the metrics perspective, we observe a trade off between model quality and log complexity for two logs (BPI13CP and BPI13I). That is, the cost of reducing entropy also implies diminishing the average model's fitness and precision. Contrarily, for BPI12 and Helpdesk, we observe an increase in both facets. The relationship between the balance within these metrics might be influenced by the underlying log behavior. More experiments to generalize these findings are necessary. Overall, the method showed a good capability in reducing complexity throughout the experiments, indicating that trace clustering pipeline synthesis can enable a better understanding of behaviors in event logs. The second part of the experiments demonstrated that random modeling leads to sub-optimal results as the pipeline search space is enormous and finding a suitable sequence of algorithms is not a simple task. In terms of time consumption, the proposed method also delivers consistently.

Furthermore, a few limitations can also be identified. First, the collection of event logs is limited; further investigation needs to be conducted to analyze the proposed method when facing diverse scenarios. Although there exists research in the direction of automating trace clustering pipeline design [4], a direct comparison is not viable for a few reasons. The pipeline search space of these methods is different (our proposal is much more extensive), which hinders a fair evaluation. The meta-learning based method also requires a high number of event logs to build the meta-database. Finally, the MO function is significantly different, meaning that each method optimizes distinct dimensions.

We believe the proposed approach takes one step further in the incorporation of automated pipeline synthesis in PM. The biggest contribution is releasing the burden from specialists who would need to design trace clustering pipelines by trial-and-error. With an automated solution, analysis is facilitated and human time can be freed. Moreover, the approach is similar to zero-shot learning as it does not need to build a knowledge database. One can simply plug the desired event log to find its corresponding pipeline.

## 7. Conclusions

Process mining involves extracting information from event logs, which are often large, noisy, and contain many relationships. This can result in the creation of complex models that are difficult to interpret. To address this problem, trace clustering is used to identify groups with similar behavior. However, selecting appropriate trace clustering parameters can be daunting when dealing with vast amounts of data.

Clustering algorithms are sensitive to hyperparameters, requiring optimization and hyperparameter search spaces. AutoML automates this process by selecting algorithms and hyperparameters. Nevertheless, the construction of trace clustering pipelines remains complex and involves several complicated steps.

To address these challenges, we propose an automated method for synthesizing trace clustering pipelines using genetic programming (GP). Our approach optimizes various dimensions to find an optimal pipeline. The results demonstrate the viability of our solution in real-world scenarios, balancing model quality with sublog complexity. Our analysis spans both process mining (PM) and data mining perspectives, providing valuable insights for business analysis while maintaining rigorous optimization goals.

For future research, further experiments are needed to generalize the presented results, including the inclusion of additional event logs in the pipeline. Additionally, exploring different AutoML-based methods with different multi-objective (MO) optimization functions could provide a more robust approach to pipeline synthesis.

**Author Contributions:** Conceptualization: I.M.G., G.M.T., M.C.d.S., P.C. and S.B.J.; methodology: I.M.G., M.C.d.S. and P.C.; software: S.B.J. and I.M.G.; writing—original draft preparation: G.M.T., M.C.d.S., S.B.J. and I.M.G.; supervision: I.M.G. and P.C. All authors have read and agreed to the published version of the manuscript.

**Funding:** This research received no external funding.

**Institutional Review Board Statement:** Not applicable.

**Informed Consent Statement:** Not applicable.

**Data Availability Statement:** Data are contained within the article.

**Conflicts of Interest:** The authors declare no conflicts of interest.

## References

1. van der Aalst, W.M. Business Process Management: A Comprehensive Survey. *ISRN Softw. Eng.* **2013**, *2013*, 507984. [CrossRef]
2. Martin, N.; Fischer, D.A.; Kerpedzhiev, G.D.; Goel, K.; Leemans, S.J.J.; Röglinger, M.; van der Aalst, W.M.P.; Dumas, M.; La Rosa, M.; Wynn, M.T. Opportunities and Challenges for Process Mining in Organizations: Results of a Delphi Study. *Bus. Inf. Syst. Eng.* **2021**, *63*, 511–527. [CrossRef]
3. van der Aalst, W.M.P.; Carmona, J. *Process Mining Handbook*; Springer: Berlin/Heidelberg, Germany, 2022.
4. Tavares, G.M.; Barbon Junior, S.; Damiani, E.; Ceravolo, P. Selecting Optimal Trace Clustering Pipelines with Meta-learning. In *Proceedings of the Intelligent Systems*; Xavier-Junior, J.C., Rios, R.A., Eds.; Springer International Publishing: Cham, Switzerland, 2022; pp. 150–164.
5. Neubauer, T.R.; Pamponet Sobrinho, G.; Fantinato, M.; Peres, S.M. Visualization for enabling human-in-the-loop in trace clustering-based process mining tasks. In Proceedings of the 2021 IEEE International Conference on Big Data (Big Data), Orlando, FL, USA, 15–18 December 2021; pp. 3548–3556. [CrossRef]
6. Olson, R.S.; Bartley, N.; Urbanowicz, R.J.; Moore, J.H. Evaluation of a Tree-based Pipeline Optimization Tool for Automating Data Science. In Proceedings of the Genetic and Evolutionary Computation Conference 2016, New York, NY, USA, 20–24 July 2016; pp. 485–492. [CrossRef]
7. James, G.; Witten, D.; Hastie, T.; Tibshirani, R. *An Introduction to Statistical Learning*; Springer: Berlin/Heidelberg, Germany, 2013; Volume 112.
8. Tavares, G.M.; Junior, S.B.; Damiani, E. Automating process discovery through meta-learning. In Proceedings of the International Conference on Cooperative Information Systems, Bozen-Bolzano, Italy, 4–7 October 2022; Springer: Berlin/Heidelberg, Germany, 2022; pp. 205–222.
9. Hutter, F.; Kotthoff, L.; Vanschoren, J. *Automated Machine Learning: Methods, Systems, Challenges*; Springer Nature: Berlin/Heidelberg, Germany, 2019.
10. De Koninck, P.; De Weerdt, J.; vanden Broucke, S.K.L.M. Explaining clusterings of process instances. *Data Min. Knowl. Discov.* **2017**, *31*, 774–808. [CrossRef]
11. Bose, R.P.J.C.; van der Aalst, W.M. Context Aware Trace Clustering: Towards Improving Process Mining Results. In Proceedings of the 2009 SIAM International Conference on Data Mining (SDM), Sparks, NV, USA, 30 April–2 May 2009; pp. 401–412. [CrossRef]
12. Greco, G.; Guzzo, A.; Pontieri, L.; Sacca, D. Discovering expressive process models by clustering log traces. *IEEE Trans. Knowl. Data Eng.* **2006**, *18*, 1010–1027. [CrossRef]
13. Song, M.; Günther, C.W.; van der Aalst, W.M.P. Trace Clustering in Process Mining. In *Proceedings of the Business Process Management Workshops*; Ardagna, D., Mecella, M., Yang, J., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 109–120.
14. Bui, H.-N.; Tri-Thanh Nguyen, T.C.N.; Ha, Q.T. A New Trace Clustering Algorithm Based on Context in Process Mining. In *Lecture Notes in Computer Science*; Springer International Publishing: Cham, Switzerland, 2018; Volume 11103. [CrossRef]
15. Boltenhagen, M.; Chatain, T.; Carmona, J. Generalized Alignment-Based Trace Clustering of Process Behavior. In *Lecture Notes in Computer Science*; Springer International Publishing: Cham, Switzerland, 2019; Volume 11522. [CrossRef]
16. Jablonski, S.; Röglinger, M.; Schönig, S.; Wyrtki, K.M. Multi-Perspective clustering of process execution traces. *Enterp. Model. Inf. Syst. Archit. (Emisaj) Int. J. Concept. Model.* **2019**, *14*. [CrossRef]
17. Fani Sani, M.; Boltenhagen, M.; van der Aalst, W. Prototype Selection Using Clustering and Conformance Metrics for Process Discovery. In *Proceedings of the Business Process Management Workshops*; Del Río Ortega, A., Leopold, H., Santoro, F.M., Eds.; Springer International Publishing: Cham, Switzerland, 2020; pp. 281–294.
18. de Medeiros, A.K.A.; Guzzo, A.; Greco, G.; van der Aalst, W.M.P.; Weijters, A.J.M.M.; van Dongen, B.F.; Saccà, D. Process Mining Based on Clustering: A Quest for Precision. In *Proceedings of the Business Process Management Workshops*; ter Hofstede, A., Benatallah, B., Paik, H.Y., Eds.; Springer: Berlin/Heidelberg, Germany, 2008; pp. 17–29.
19. Delias, P.; Doumpos, M.; Grigoroudis, E.; Manolitzas, P.; Matsatsinis, N. Supporting healthcare management decisions via robust clustering of event logs. *Knowl.-Based Syst.* **2015**, *84*, 203–213. [CrossRef]
20. Lakshmi Narayana, N.; Jagadishwari, V. Trace Clustering Techniques for Process Mining. In Proceedings of the 2023 Third International Conference on Advances in Electrical, Computing, Communication and Sustainable Technologies (ICAECT), Bhilai, India, 5–6 January 2023; pp. 1–6. [CrossRef]
21. De Koninck, P.; Nelissen, K.; vanden Broucke, S.; Baesens, B.; Snoeck, M.; De Weerdt, J. Expert-driven trace clustering with instance-level constraints. *Knowl. Inf. Syst.* **2021**, *63*, 1197–1220. [CrossRef]

22. Thornton, C.; Hutter, F.; Hoos, H.H.; Leyton-Brown, K. Auto-WEKA: Combined selection and hyperparameter optimization of classification algorithms. In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 11–14 August 2013.
23. Feurer, M.; Klein, A.; Eggensperger, K.; Springenberg, J.; Blum, M.; Hutter, F. Efficient and robust automated machine learning. *Adv. Neural Inf. Process. Syst.* **2015**, *28*, 2962–2970.
24. Chen, B.; Wu, H.; Mo, W.; Chattopadhyay, I.; Lipson, H. Autostacker: A compositional evolutionary learning system. In Proceedings of the Genetic and Evolutionary Computation Conference, New York, NY, USA, 15–19 July 2018; pp. 402–409.
25. van der Aalst, W.M.P. Process Mining: A 360 Degree Overview. In *Lecture Notes in Business Information Processing*; Springer International Publishing: Cham, Switzerland, 2022; Volume 448.
26. Tavares, G.M.; Barbon Junior, S. Matching business process behavior with encoding techniques via meta-learning: An anomaly detection study. *Comput. Sci. Inf. Syst.* **2023**, *20*, 1207–1233. [CrossRef]
27. ElShawi, R.; Sakr, S. TPE-AutoClust: A Tree-based Pipline Ensemble Framework for Automated Clustering. In Proceedings of the 2022 IEEE International Conference on Data Mining Workshops (ICDMW), Orlando, FL, USA, 28 November–1 December 2022; pp. 1144–1153. [CrossRef]
28. Fu, W.; Olson, R.; Nathan; Jena, G.; PGijsbers; Augspurger, T.; Romano, J.; Saha, P.; Shah, S.; Raschka, S.; et al. EpistasisLab/tpot: V0.11.5. 2020. Available online: https://zenodo.org/records/3872281 (accessed on 17 March 2024).
29. De Koninck, P.; vanden Broucke, S.; De Weerdt, J. act2vec, trace2vec, log2vec, and model2vec: Representation Learning for Business Processes. In *Proceedings of the Business Process Management*; Weske, M., Montali, M., Weber, I., vom Brocke, J., Eds.; Springer International Publishing: Cham, Switzerland, 2018; pp. 305–321.
30. Polato, M.; Sperduti, A.; Burattin, A.; Leoni, M.d. Time and activity sequence prediction of business process instances. *Computing* **2018**, *100*, 1005–1031. [CrossRef]
31. Barbon Junior, S.; Ceravolo, P.; Damiani, E.; Marques Tavares, G. Evaluating Trace Encoding Methods in Process Mining. In *Proceedings of the From Data to Models and Back*; Bowles, J., Broccia, G., Nanni, M., Eds.; Springer International Publishing: Cham, Switzerland, 2021; pp. 174–189.
32. Tavares, G.M.; Oyamada, R.S.; Junior, S.B.; Ceravolo, P. Trace encoding in process mining: A survey and benchmarking. *Eng. Appl. Artif. Intell.* **2023**, *126*, 107028. [CrossRef]
33. Weiss, S.M.; Indurkhya, N.; Zhang, T. *Fundamentals of Predictive Text Mining*, 2nd ed.; Texts in Computer Science; Springer: Berlin/Heidelberg, Germany, 2015. [CrossRef]
34. Rozinat, A.; van der Aalst, W. Conformance checking of processes based on monitoring real behavior. *Inf. Syst.* **2008**, *33*, 64–95. [CrossRef]
35. Grover, A.; Leskovec, J. Node2vec: Scalable Feature Learning for Networks. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, New York, NY, USA, 13–17 August 2016; pp. 855–864. [CrossRef]
36. Buitinck, L.; Louppe, G.; Blondel, M.; Pedregosa, F.; Mueller, A.; Grisel, O.; Niculae, V.; Prettenhofer, P.; Gramfort, A.; Grobler, J.; et al. API design for machine learning software: Experiences from the scikit-learn project. In Proceedings of the ECML PKDD Workshop: Languages for Data Mining and Machine Learning, Prague, Czech Republic, 23–27 September 2013; pp. 108–122.
37. Xu, R.; Wunsch, D. Survey of clustering algorithms. *IEEE Trans. Neural Netw.* **2005**, *16*, 645–678. [CrossRef] [PubMed]
38. Rousseeuw, P.J. Silhouettes: A graphical aid to the interpretation and validation of cluster analysis. *J. Comput. Appl. Math.* **1987**, *20*, 53–65. [CrossRef]
39. Augusto, A.; Mendling, J.; Vidgof, M.; Wurm, B. The connection between process complexity of event sequences and models discovered by process mining. *Inf. Sci.* **2022**, *598*, 196–215. [CrossRef]