Enhancing Approximate Conformance Checking Accuracy with Hierarchical Clustering Model Behaviour Sampling

Yilin Lyu
School of Computing and
Information System
The University of Melbourne
Parkville, Victoria 3010, Australia
Email: yillyu1@student.unimelb.edu.au

***Abstract*—Conformance checking techniques evaluate how well a process model aligns with an actual event log. Existing methods, which are based on optimal trace alignment, are computationally intensive. To improve efficiency, a model sampling method has been proposed to construct a subset of model behaviour that represents the entire model. However, current model sampling techniques often lack sufficient model representativeness, limiting their potential to achieve optimal approximation accuracy. This paper proposes new model behaviour sampling approaches using hierarchical clustering to compute an approximation closer to the exact result. This paper also refines the existing upper bound algorithm for better approximation. Our experiments on six real-world event logs demonstrate that our method improves approximation accuracy compared to state-of-the-art model sampling methods.**

Keywords—Approximate conformance checking Model behaviour sampling Hierarchical clustering Process mining

I. INTRODUCTION

Conformance checking is a set of process mining functionalities aimed at identifying deviations between the actual behaviour of the event log ("as-is") and the modeled behaviour of the process model ("to-be"). It facilitates further applications, such as model repair, anomaly detection, and algorithm evaluation [1]. In recent years, the alignment-based method [2] has become the de facto standard for conformance checking in the computation of conformance diagnostics, since it always returns the most accurate deviations, known as optimal alignment [3]. However, finding the optimal alignment is an NP-hard problem [4]. As the complexity of the log and model increases, the run-time complexity of optimal alignment computation grows exponentially, leading to extremely long computation times, sometimes even taking several weeks. This makes them impractical for real-world applications, especially large-scale event logs. Moreover, in certain cases, an exact conformance value is not necessary, such as when performing a preliminary evaluation of process models with various process discovery algorithms [5].

To address the problems, various approximation strategies have been proposed, including optimizing the search algorithm [6], [7] and decomposition schemes [8], [9]. However, sampling provides another angle for approximate conformance checking, such as sampling traces to represent the event log [10], [11] or selecting model traces to substitute for the process model [5], [12]. In this paper, we adopt the latter approach, focusing on model sampling. Two main model sampling methods exist: simulation [13] and candidate selection [5]. We concentrate on candidate selection due to its higher accuracy [5]. The candidate selection method identifies representative traces from the event log (i.e., log behaviour subset), and then computes their optimal alignments to determine the corresponding model traces (i.e., model behaviour subset). The accuracy of this approximation depends on the quality of the selected log traces [12]. However, existing log selection techniques (e.g., random, frequency-based [5], K-Medoids [14]) often lack behavioural diversity and model representativeness (see Section II), leading to reduced accuracy in the conformance approximation. Hence, there is significant potential for improving the quality of model behaviour subsets.

In this paper, we propose an enhanced model behaviour sampling method to select more representative subsets and obtain more accurate approximate values. First, we apply hierarchical clustering to the event log using our proposed distance criterion. Then, we propose two in-cluster methods to select typical traces from each cluster, which are then used to construct more representative model behaviour subsets. Finally, we extend the existing cost lower bound algorithm to achieve more accurate approximation results. The experimental results show that our approach yields more accurate approximations than existing baselines, though with an increased approximation time.

The remainder of this paper is organized as follows. Section II provides a motivating example to further illustrate the research problem. Section III discusses

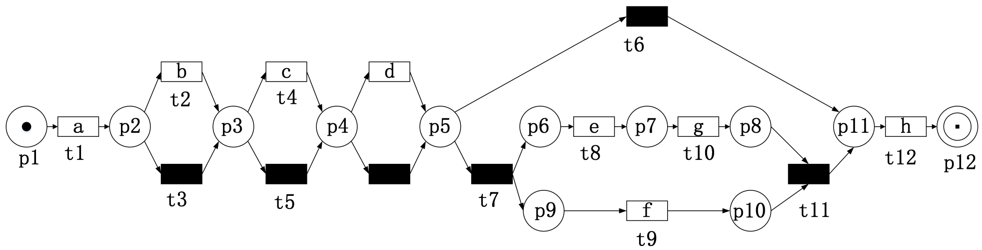

Fig. 1. The Process Model discovered by Inductive Miner with infrequent threshold equals to 0.9.

related work in approximate conformance checking. Section IV outlines the necessary preliminaries. In Section V, we propose our method for constructing model behaviour subsets using hierarchical clustering. Section VI details the evaluation setup. Section VII presents the experimental results, followed by Section VIII. Finally, Section IX concludes the paper and presents the limitations and future work.

II. MOTIVATING EXAMPLE

Research such as [5] and [15] has shown that selecting more typical log traces lead to higher approximation accuracy. Thus, the key challenge is determining which subset should be selected to improve approximate accuracy. Existing log selection methods, such as the frequency-based and K-medoids approaches, sometimes lack sufficient log representativeness.

To illustrate the potential limitations of these methods, we use a synthesized event log L. It contains 5,106 traces consisting of 32,600 events and 12 trace variants, as shown in Table 1.

TABLE 1. EVENT LOG

ID	Trace Variant	Freq	ID	Trace Variant	Freq
0	$\langle a,b,c,d,f,e,g,h\rangle$	1280	6	$\langle a,d,f,h\rangle$	250
1	$\langle a,b,c,d,e,f,g,h\rangle$	912	7	$\langle a,f,b,c\rangle$	96
2	$\langle a,b,c,d,e,g,f,h\rangle$	864	8	$\langle a,c,e,f,g\rangle$	64
3	$\langle a,b,c,h\rangle$	792	9	$\langle a,d,e,g,h\rangle$	56
4	$\langle a,b,c,d,h\rangle$	400	10	$\langle a,b,f,e,g,h\rangle$	48
5	$\langle a,h\rangle$	320	11	$\langle b,f,g\rangle$	24

To discover the event log presented in Figure 1, we applied the Inductive Miner algorithm [16] with infrequent thresholds of 0.9.

Assuming that we select three variants to represent the event log, i.e., the behavior subset consists of three variants. Table II shows the behaviour subsets generated by the frequency-based method, K-Medoids, and our proposed methods (see Section V for details). The frequency-based subsets show two key limitations:
a) Overestimation of Alignment Cost: Variant 5, $\langle a, h\rangle$, can be perfectly replayed in the model with an alignment cost of 0. But it is not included in our model behaviour subset, aligning it would require at least 6 insertions (i.e., cost of 6), resulting in an overestimated approximate cost.
b) Lack of Structural Diversity: The selected model traces $\langle a,b,c,d,f,e,g,h\rangle$ and $\langle a,b,c,d,e,f,g,h\rangle$ differ only in the order of e and f. This means that they represent essentially the similar structural path, potentially overlooking other important paths in the process model.

Also, the K-Medoids method has drawbacks: it clusters traces solely based on their control-flow information, that is, syntactic difference. For example, the trace $\langle b, f, g\rangle$ in log behaviour subset (as shown in Table II) may have significantly syntactic differences from other traces but, due to its low frequency (only 24 occurrences), it is still not enough to represent the model behaviour.

To address the issues, our approach proposed in Section V effectively balances frequency and control-flow information. Table II also shows the cost deviation. It refers to the difference in alignment cost between using model behaviour subset and optimal-alignment. The values indicate that the model behaviours generated by our methods significantly reduce the cost deviations compared to vanilla methods.

TABLE 2. BEHAVIOUR SUBSETS CONSTRUCTED BY FOUR METHODS

Method	Subset	Result	Cost Deviation
Frequency-based	Log Behaviour	$\Sigma_L = \{\langle a,b,c,d,f,e,g,h\rangle, \langle a,b,c,d,e,f,g,h\rangle, \langle a,b,c,d,e,g,f,h\rangle\}$	7806
	Model Behaviour	$\Sigma_M = \{\langle a,b,c,d,f,e,g,h\rangle, \langle a,b,c,d,e,f,g,h\rangle, \langle a,b,c,d,e,g,f,h\rangle\}$	
K-Medoids	Log Behaviour	$\Sigma_L = \{\langle a,h\rangle, \langle a,b,c,d,e,g,f,h\rangle, \langle b,f,g\rangle\}$	6596
	Model Behaviour	$\Sigma_M = \{\langle a,h\rangle, \langle a,b,c,d,e,g,f,h\rangle, \langle a,b,e,f,g,h\rangle\}$	
In-cluster frequency	Log Behaviour	$\Sigma_L = \{\langle a,h\rangle, \langle a,b,c,d,f,e,g,h\rangle, \langle a,b,c,h\rangle\}$	4698
	Model Behaviour	$\Sigma_M = \{\langle a,h\rangle, \langle a,b,c,d,f,e,g,h\rangle \langle a,b,c,h\rangle\}$	
In-cluster medoid	Log Behaviour	$\Sigma_L = \{\langle a,d,f,h\rangle, \langle a,b,c,d,f,e,g,h\rangle, \langle a,b,c,h\rangle\}$	4854
	Model Behaviour	$\Sigma_M = \{\langle a,d,h\rangle, \langle a,b,c,d,f,e,g,h\rangle \langle a,b,c,h\rangle\}$	

III. RELATED WORK

To cope with the complexity of alignment construction, approximation techniques have been developed to balance result quality and computational cost. Early studies explored replacing the A* algorithm with faster algorithms[7], [17], [18], laying the foundation for a more efficient alignment computation. For example, Taymouri and Carmona [17], introducing an evolutionary algorithm to enhance alignment approximations. Model decomposition has also been investigated as an efficiency-oriented approximation approach. The foundational work demonstrated how breaking models into smaller and more manageable parts can simplify alignment, although it may not always result in optimal alignments [19], [20]. Furthermore, the construction of automata capable of aligning the log and the model has been explored as another approximation technique [21], [22]. This approach provides good approximations of the optimal alignments in most cases. Recently, some researchers have proposed using RNN-based neural networks to obtain recall and precision metrics for event logs and process models, demonstrating the potential of this technique for conformance analysis [23], [24].

Reducing behaviour size is another promising strategy for approximate conformance checking. One sampling approach focuses on sampling the event log. For instance, [25] proposes a trace sampling method, assuming that a few log traces can estimate the conformance value. However, it lacks upper and lower bounds for the approximation and performs worse when the event log contains many unique behaviors.

Another recent sampling approach targets model behaviour. [5] introduced a model sampling method to construct subsets of the model behaviour that represent the entire process model, significantly reducing the approximation time while largely maintaining accuracy. The method also provides upper and lower bounds to give some certainty of the approximation.

Hierarchical clustering is widely used in process mining for its structural representativeness [26]. Furthermore, [27] demonstrates how hierarchical clustering aids in discovering a better model.

IV. PRELIMINARIES

This section presents the terminology and notation for conformance checking to support the subsequent sections. We use the basic definitions of Petri net, e.g., labeled Petri Net in [28].

Given a system net SN, $\phi_f(SN)$ is the set of all complete firing sequences of SN and $\phi_v(SN)$ is the set of all possible visible traces, i.e., complete firing sequences starting in its initial marking and ending in its final marking projected onto the set of observable activities (not silent transitions, e.g., t_3 in Figure 1).

To measure how a trace aligns to a process model, moves are represented by pairs (a, t), where a is a log activity, and t is a model transition. Legal moves can be: *log moves* , *model moves*, or *synchronous moves* . Any other combination is an *illegal move*.

Definition 1. ***(Alignment).*** *Let $\sigma_L \in L$ represent a log trace and $\sigma_M \in \phi_f(SN)$ denote a complete firing sequence of a system net SN. A_{LM} is the set of legal moves. An alignment of σ_L and σ_M is a sequence of pairs $\gamma \in A^*_{LM}$ such that the projection on the first element (ignoring $\gg$) yields σ_L and the projection on the second element (ignoring $\gg$ and transition labels) yields σ_M.*

To quantify the costs of alignments we introduce a cost function δ in Definition 2.

Definition 2. ***(Cost of Alignment).*** *Cost function $\delta \in A_{LM} \to \mathbb{N}$ assigns costs to legal moves. The cost of an alignment $\gamma \in A^*_{LM}$ is the sum of all costs:*

$$\delta(\gamma) = \sum_{(a,t)\in\gamma} \delta(a, t).$$

The cost values assigned to log moves, model moves, and synchronous moves are 1, 1, and 0, respectively. Note that an alignment is considered optimal if it has the minimum alignment cost.

Definition 3. ***(Optimal Alignment).*** *Let L be an event log and SN a system net where $\phi_v(SN) \neq \emptyset$.*

- *For $\sigma_L \in L$, we define: $\Gamma_{\sigma_L,SN} \in \{\gamma \in A^*_{LM} \mid \exists\sigma_M \in \phi_f(SN)$ is an alignment of σ_L and $\sigma_M\}$.*
- *An alignment $\gamma \in \Gamma_{\sigma_L,SN}$ is optimal for trace $\sigma_L \in L$ and system net SN if for any alignment $\gamma' \in \Gamma_{\sigma_L,M}$: $\delta(\gamma') \geq \delta(\gamma)$.*
- *$\gamma_{SN} \in A^*_{LM} \to A^*_{LM}$ is a mapping that assigns any log trace σ_L to an optimal alignment, i.e., $\gamma_{SN}(\sigma_L) \in \Gamma_{\sigma_L,SN}$ and $\gamma_{SN}(\sigma_L)$ is an optimal alignment.*

Definition 4. ***(Levenshtein Edit Distance).*** *As defined by [29], the Levenshtein edit distance $d(\sigma_1, \sigma_2) \to \mathbb{N}$ represents the minimum number of edit operations (i.e., insertions, deletions, and substitutions) required to transform one sequence into another. For instance, $d(\langle a, b\rangle, \langle c, d\rangle) = 2$, where the two edit operations are substitutions (a, c) and (b, d).*

Definition 5. ***(Edit Distance Cost Function).*** *We can calculate the distance between two traces (or sequences) faster by using a modified version of the Levenshtein edit distance [30]. Let $\sigma_1, \sigma_2 \in A^*$ be two sequences of activities. The Edit Distance Cost Function $\Delta(\sigma_1, \sigma_2) \to \mathbb{N}$ is defined as the minimum number of edits (insertion or deletion of activities) required to transform σ_1 into σ_2.*

Suppose that S is a set of sequences, $\Phi(\sigma_L, S) = \min_{\sigma_M\in S} \Delta(\sigma_L, \sigma_M)$ returns the distance of the most similar sequence in S. Let $\phi_v(SN)$ be the set of all visible firing sequences in SN, and $\gamma_{SN}(\sigma)$ be an optimal alignment for sequence σ. It is possible to prove that $\delta_S(\gamma_{SN}(\sigma)) = \Phi(\sigma, \phi_v(SN))$[12].

In the context of alignment, the edit distance function can be used as a cost function δ_S for evaluating the misalignment between a log trace σ_L and a model trace σ_M. This cost function assigns a value corresponding to the number of operations required to align the two sequences. For example, $\Delta(\langle a, c, b, e, d\rangle, \langle a, b, c, a, d\rangle) = 4$ corresponds to two deletions and two insertions.

Moreover, the alignment cost of a single trace can be converted into a fitness value between 0 (poor fitness, i.e., maximal costs) and 1 (perfect fitness, i.e., zero costs) using Equation 1 [5]. In this regard, we normalize this cost relative to the worst case, with one log move for each activity in the trace and one model move for each transition in the model's shortest path, $SPM = \min_{\sigma_M \in \phi_f}(|\sigma_M|)$. Here, the optimal alignment cost, $\delta(\gamma_{SN}(\sigma))$, can be replaced by an alternative cost (e.g., edit distance cost) to obtain a corresponding fitness value.

$$Fitness_{\text{Trace}}(\sigma_L, SN) = 1 - \frac{\delta_S(\gamma_{SN}(\sigma))}{|\sigma_L| + SPM} \tag{1}$$

Note that the overall fitness between the event log and the system net is the weighted average of single trace fitness values.

V. METHOD

In this section, we present the proposed conformance approximation method. An overview of our approach is shown in Figure 2. The method begins with a preprocessing stage using hierarchical clustering techniques. Next, two methods are proposed for constructing model behaviour subsets: in-cluster frequency and in-cluster medoid methods. Finally, the alignment approximation process is explained.

A. Preprocess event log using hierarchical clustering

In this stage, we apply agglomerative hierarchical clustering [31] on event logs. Specifically, we first partition the event log based on trace variants to get the trace variant subset Σ_{σ_v}. Then, we introduce the normalized weighted Levenshtein distance to measure the distance between these variants(see Definition 6) as a new in-cluster distance criterion. This criterion considers both frequency and control-flow information, alleviating the problem with current log selection methods mentioned in Section II. It is used to build a distance matrix, then forming a dendrogram. By cutting-off the dendrogram, we obtain the desired number of clusters. The framework is illustrated in Figure 3.

Definition 6. ***(Normalized Weighted Levenshtein Distance).*** *Let A^* be the set of all possible sequences of activities in A, and let σ_{v1}, σ_{v2} be two trace variants $\in A^*$. The normalized weighted Levenshtein distance between σ_{v1} and σ_{v2}, where each trace variant has a frequency $f(\sigma_{v1})$ and $f(\sigma_{v2})$, is defined as:*

$$d_{weighted}(\sigma_{v1}, \sigma_{v2}) = \frac{f(\sigma_{v1}) \cdot f(\sigma_{v2}) \cdot d_N(\sigma_{v1}, \sigma_{v2})}{\max\{f(\sigma_{v1})^2, f(\sigma_{v2})^2\}} \tag{2}$$

where the normalized Levenshtein distance $d_N(\sigma_{v1}, \sigma_{v2})$ is given by:

$$d_N(\sigma_{v1}, \sigma_{v2}) = \frac{d(\sigma_{v1}, \sigma_{v2})}{\max\{|\sigma_{v1}|, |\sigma_{v2}|\}} \tag{3}$$

Here, $d_N(\sigma_{v1}, \sigma_{v2}) = 0$ means the two traces are exactly the same, and $d_N(\sigma_{v1}, \sigma_{v2}) = 1$ means the two traces are completely different.

Definition 7. ***(Distance Matrix).*** *Let $\sigma_{v1}, \sigma_{v2}, \ldots, \sigma_{vi} \in A^*$ represent all trace variants in event log L. The matrix $D(L)$ is defined as, :*

$$D(L) = \begin{bmatrix} 0 & d(\sigma_{v1}, \sigma_{v2}) & \cdots & d(\sigma_{v1}, \sigma_{vi}) \\ d(\sigma_{v2}, \sigma_{v1}) & 0 & \cdots & d(\sigma_{v2}, \sigma_{vi}) \\ \vdots & \vdots & \ddots & \vdots \\ d(\sigma_{vi}, \sigma_{v1}) & d(\sigma_{vi}, \sigma_{v2}) & \cdots & 0 \end{bmatrix} \tag{4}$$

where d is the normalized weighted Levenshtein distance function.

B. Constructing Model Behaviour

In this stage, we first propose two in-cluster methods to get log behaviour subset Σ_L from the generated clusters and transform it into the model behaviour subset Σ_M. Specifically,

a) Candidate selection: After preprocessing, we obtain several clusters, each representing different behaviours within the model. The following question is how to choose the most representative traces from each cluster to construct a more effective log behaviour subset. Existing approaches in approximate conformance checking often rely on either random sampling or frequency-based selection without considering control-flow similarity, which may lead to biased or suboptimal subsets when the frequency distribution is highly imbalanced or when rare but structurally central behaviours exist. To address this, we extend the ideas of frequency-based and medoid selection by introducing two in-cluster methods — the in-cluster frequency method and the in-cluster medoid method — designed to balance efficiency and representativeness.

The **in-cluster frequency method** selects, from each cluster, the trace variant with the highest frequency of occurrence. This approach assumes that the most common behaviour within a cluster is also the most representative of that cluster's behaviour. Its main advantage lies in computational efficiency, as it does not require computing pairwise distances between traces. Compared to methods that sample traces uniformly at random [25], the frequency method reduces the risk of including low-relevance traces, especially in large-scale logs.

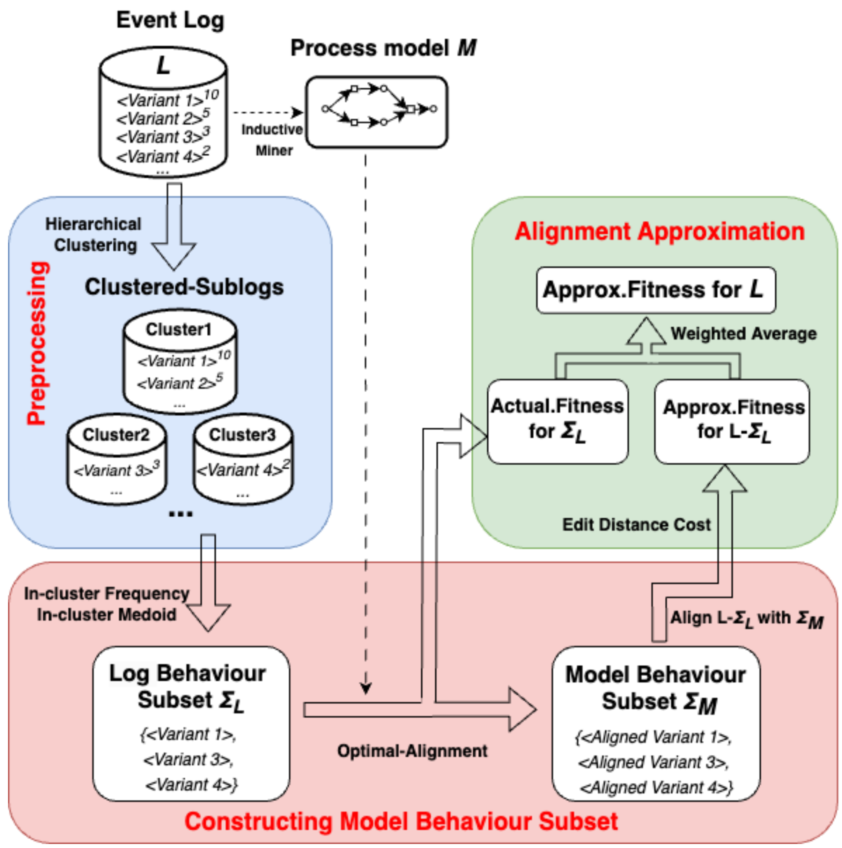

Fig. 2. Overview of our approach

The **in-cluster medoid method**, in contrast, selects the trace variant that minimises the total Levenshtein distance to all other traces in the cluster, effectively identifying the "central" trace in terms of control-flow similarity. Specifically, it computes the pairwise Levenshtein distances between all traces in each cluster, then constructs a distance matrix and obtains the medoid trace (see Definition 8). This ensures that the selected trace best represents the structural characteristics of its cluster, even if it is not the most frequent. Compared to traditional frequency-only methods, the medoid approach mitigates the bias towards dominant behaviours and is more robust when clusters contain diverse but equally important behaviours.

b) Optimal-alignment: In this step, we align Σ_L with process model to construct the Σ_M, that is, we compute the optimal alignments of selected traces in the event log and finding the corresponding model traces for these alignments.

Table 3 shows three clusters generated from the event log in Table 1. For example, applying the in-cluster frequency method to cluster 2 yields $\langle a, b, c, h \rangle^{792}$, the most frequent trace. Repeating this for each cluster, we obtain $\Sigma_L = \{\langle a, b, c, d, f, e, g, h \rangle^{1280}, \langle a, b, c, h \rangle^{792}, \langle a, h \rangle^{320}\}$. We then align Σ_L with the process model as shown in Figure 1, resulting in Σ_M.Note that Σ_L and Σ_M are same in this example, as all traces can be fully replayed in the model.

TABLE 3. THE CLUSTERS GENERATED FROM THE EXAMPLE LOG PROVIDED IN TABLE 1

Cluster ID	Traces in each cluster
1	$\{\langle a, b, c, d, f, e, g, h \rangle^{1280}, \langle a, b, c, d, e, f, g, h \rangle^{912}, \langle a, b, c, d, e, f, g, h \rangle^{864}\}$
2	$\{\langle a, b, c, h \rangle^{792}, \langle a, b, c, d, h \rangle^{400}, \langle a, f, b, c \rangle^{96}\}$
3	$\{\langle a, h \rangle^{320}, \langle a, d, h \rangle^{250}, \langle a, c, e, f, g \rangle^{64}, \langle a, d, e, g, h \rangle^{56}, \langle a, b, f, e, g, h \rangle^{48}, \langle b, f, g \rangle^{24}\}$

The specific algorithm steps for proposed methods are outlined in Algorithms 1 and 2.

Definition 8. ***(In-cluster Medoid).*** *Let L' be a clustered sublog, n denote the number of trace variants in L', and $D(L')$ be the distance matrix of L'. The trace $\sigma_j = \arg\min_{\sigma_j \in L'} \sum_{i \in [1,n]} d(\sigma_i, \sigma_j)$ represents the medoid trace of sublog L'.*

C. Computing Alignment Approximation

After constructing M_B, we use it to approximate alignments for the traces in $L - L_C$, where L_C refers to the frequency-based trace variants used to build Σ_L. The actual alignment fitness for the variants in Σ_L has already been computed during the construction of M_B, so we can directly use this value for more accurate approximations.

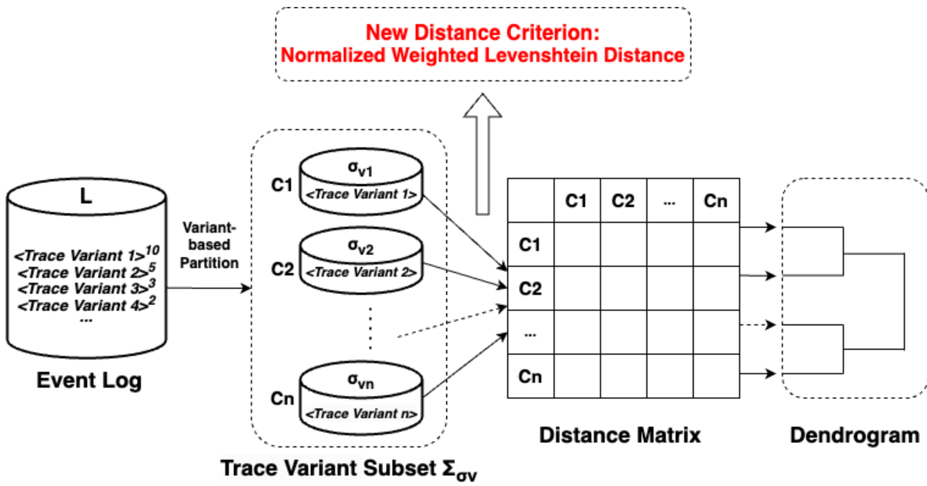

Fig. 3. Preprocessing workflow for hierarchical clustering

Algorithm 1 In-cluster Medoid Method

Input: Event log L; Process model M.
Output: Model behaviour subset Σ_M.

Initialize log behaviour subset: $\Sigma_L \leftarrow \emptyset$
Initialize model behaviour subset: $\Sigma_M \leftarrow \emptyset$
Partition L based on variants into Σ_{σ_v}
Cluster Σ_{σ_v} into k clusters $\{\Sigma_{\sigma_{v1}}, \Sigma_{\sigma_{v2}}, \ldots, \Sigma_{\sigma_{vk}}\}$ using hierarchical clustering
for $i = 1$ to k **do**
  Compute pairwise Levenshtein distances between all variants in $\Sigma_{\sigma_{vi}}$
  Construct distance matrix $D(\Sigma_{\sigma_{vi}})$
  Find the medoid trace $\sigma_L^{(i)}$ in $\Sigma_{\sigma_{vi}}$:

$$\sigma_L^{(i)} = \arg \min_{\sigma \in \Sigma_{\sigma_{vi}}} \sum_{\sigma' \in \Sigma_{\sigma_{vi}}} d(\sigma, \sigma')$$

  Update log behaviour subset: $\Sigma_L \leftarrow \Sigma_L \cup \{\sigma_L^{(i)}\}$
end for
for each $\sigma_L^{(i)} \in \Sigma_L$ **do**
  Compute optimal alignment γ_{SN}^{opt} between $\sigma_L^{(i)}$ and M
  Map to model trace: $\sigma_M^{(i)} \leftarrow \lambda_{SN}(\sigma_L^{(i)})$
  Update model behaviour subset: $\Sigma_M \leftarrow \Sigma_M \cup \{\sigma_M^{(i)}\}$
end for
return Σ_M

At this stage, we calculate the alignment approximations for the remaining variants.

Typically, the actual fitness is calculated using standard alignment costs. However, for the remaining variants, we use the edit distance cost function Δ (see Definition 5) to estimate fitness. This method provides guaranteed upper and lower bounds for the alignment cost, instead of exact values [5] (see Lemma 1 and

---

Algorithm 2 In-cluster Frequency Method

Input: Event log L; Process model M.
Output: Model behaviour subset Σ_M.

Initialize log behaviour subset: $\Sigma_L \leftarrow \emptyset$
Initialize model behaviour subset: $\Sigma_M \leftarrow \emptyset$
Partition L based on variants into Σ_{σ_v}
Cluster Σ_{σ_v} into k clusters $\{\Sigma_{\sigma_{v1}}, \Sigma_{\sigma_{v2}}, \ldots, \Sigma_{\sigma_{vk}}\}$ using hierarchical clustering
for $i = 1$ to k **do**
  Let $\Sigma_{\sigma_{vi}}$ denote the i-th cluster of variants
  Find the most frequent variant $\sigma_L^{(i)}$ in $\Sigma_{\sigma_{vi}}$:

$$\sigma_L^{(i)} = \arg \max_{\sigma \in \Sigma_{\sigma_{vi}}} f(\sigma)$$

  Update log behaviour subset: $\Sigma_L \leftarrow \Sigma_L \cup \{\sigma_L^{(i)}\}$
end for
for each $\sigma_L^{(i)} \in \Sigma_L$ **do**
  Compute optimal alignment γ_{SN}^{opt} between $\sigma_L^{(i)}$ and M
  Map to model trace: $\sigma_M^{(i)} \leftarrow \lambda_{SN}(\sigma_L^{(i)})$
  Update model behaviour subset: $\Sigma_M \leftarrow \Sigma_M \cup \{\sigma_M^{(i)}\}$
end for
return Σ_M

Lemma 2 below).

$$\begin{aligned} Fitness(L, SN) = {} & \frac{\sum_{\sigma \in L_C} f(\sigma) \times Fitness_{\text{Approximate}}(\sigma, SN)}{\sum_{\sigma \in L} f(\sigma)} \\ & + \frac{\sum_{\sigma \in L - L_C} f(\sigma) \times Fitness_{\text{Actual}}(\sigma, SN)}{\sum_{\sigma \in L} f(\sigma)} \end{aligned} \tag{5}$$

Lemma 1 (Alignment Cost Upper Bound). *Let $\sigma_L \in \mathcal{U}_A^*$ be a log trace and $\sigma_M \in \phi_v(SN)$ be a visible firing sequence of SN. We have $\delta_S(\gamma_{SN}(\sigma_L)) \leq \Delta(\sigma_L, \sigma_M)$,* *where $\gamma_{SN}(\sigma_L)$ is the optimal alignment.*

Proof: The proof is provided in Appendix A and demonstrates how the edit distance guarantees this upper bound. ∎

Simply put, if we align trace variant 4 $\langle a, b, c, d, h\rangle$ from Table 1 with σ_L from the in-cluster frequency subset in Table II, the alignment cost is 1 (i.e., removing "d"). However, since σ_M is a subset of the full model, the actual cost could be smaller or equal. Thus, we use 1 as the upper bound for this variant.

Lemma 2 (Alignment Cost Lower Bound)**.** *Let $SPM = \min_{\sigma_M \in \phi_v(SN)} |\sigma_M|$ and $LPM = \max_{\sigma_M \in \phi_v(SN)} |\sigma_M|$, representing the shortest and longest paths in the process model M. $\sigma_L\lceil_{A_v(SN)}$ and $\kappa(\sigma_L)$ are as defined in Definition 9.*

For any log trace σ_L, if $|\sigma_L\lceil_{A_v(SN)}| < SPM$, the alignment cost lower bound is $SPM - |\sigma_L\lceil_{A_v(SN)}| + \kappa(\sigma_L)$; if $|\sigma_L\lceil_{A_v(SN)}| > LPM$, the lower bound is $|\sigma_L\lceil_{A_v(SN)}| - LPM + \kappa(\sigma_L)$; if $SPM \leq |\sigma_L\lceil_{A_v(SN)}| \leq LPM$, the lower bound is $\kappa(\sigma_L)$.

Proof: The proof is provided in Appendix B. ∎

The cost lower bound is the minimum edit operations needed to transform σ_L into σ_M. We refine this algorithm using activity projection (see Definition 9) to improve approximation accuracy. Existing methods compare log trace length directly with the model's range, potentially yielding errors if irrelevant activities are present. For instance, in Figure 1, a trace $\langle a, x\rangle$ might seem aligned if its length falls within the model's shortest (SPM=2) and longest paths (LPM=8), even though x is not in the model, resulting in a miscalculated cost of 0. Our algorithm removes non-model activities (e.g., removing x from $\langle a, x\rangle$ to form $\langle a\rangle$) before comparing trace lengths. This adjustment yields a more accurate cost of 1 rather than 0, resulting in a smaller upper fitness and tighter bound width.

These bounds are then used to compute the corresponding upper and lower fitness bounds (with the cost upper bound giving the fitness lower bound, and vice versa) using Equation 1. The computations for the fitness bounds are provided in Algorithm 3 and 4. The average of these bounds provides the approximate fitness. Once we compute the approximate fitness for each remaining variant, we take the weighted average of these values along with the previously computed actual fitness to get the overall approximate fitness for the entire event log, as shown in Equation 5.

Definition 9 (Activity Projection)**.** *Let $A_v(SN)$ be the set of unique observable activities in the system net SN. For any log trace σ_L, let $\sigma_L\lceil_{A_v(SN)}$ represent the projection of σ_L onto $A_v(SN)$, which means the set of activities in σ_L that also appear in the model. Define $\kappa(\sigma_L) = |\sigma_L| - |\sigma_L\lceil_{A_v(SN)}|$ as the number of activities in σ_L that are not present in the model.*

For example, let $\sigma_L = \langle a, b, x\rangle$ be a log trace and the observable activities of the system net be $A_v(SN) = \{a, b, c, d, e\}$. Projecting σ_L onto $A_v(SN)$ results in $\sigma_L\lceil_{A_v(SN)} = \langle a, b\rangle$, as x is not part of $A_v(SN)$. Therefore, $\kappa(\sigma_L) = |\sigma_L| - |\sigma_L\lceil_{A_v(SN)}| = 3 - 2 = 1$, indicating one activity in σ_L is not present in the model.

Algorithm 3 Fitness lower bound computation

Input: Event log L; Optimal-aligned Log L_C; Model behaviour subset Σ_M.
Output: Lower bound fitness $L\_fitness(\sigma_L, M)$.

for each $\sigma_L \in L - L_C$ **do**
  $\Phi(\sigma_L, \Sigma_M)$ // Compute minimun edit distance cost
  $L\_fitness(\sigma_L, M) \leftarrow 1 - \frac{\Phi(\sigma_L, \Sigma_M)}{|\sigma_L| + \min_{\sigma_M \in \phi_v(SN)}(|\sigma_M|)}$
end for
return $L\_fitness(\sigma_L, M)$

Algorithm 4 Fitness upper bound computation

Input: Event log L; Optimal-aligned Log L_C; Model behaviour subset Σ_M.
Output: Upper bound fitness $U\_fitness(\sigma_L, M)$.

$SPM \leftarrow \min_{\sigma_M \in \phi_v(SN)} |\sigma_M|$ // Shortest path
$LPM \leftarrow \max_{\sigma_M \in \phi_v(SN)} |\sigma_M|$ // Longest path
for each $\sigma_L \in L - L_C$ **do**
  Project σ_L onto SN: $\sigma_L\lceil_{A_v(SN)}$
  Compute $\kappa(\sigma_L) = |\sigma_L| - |\sigma_L\lceil_{A_v(SN)}|$
  if $|\sigma_L\lceil_{A_v(SN)}| < SPM$ **then**
    $U\_fitness(\sigma_L, M) \leftarrow 1 - \frac{SPM - |\sigma_L\lceil_{A_v(SN)}| + \kappa(\sigma_L)}{|\sigma_L| + \min_{\sigma_M \in \phi_v(SN)}(|\sigma_M|)}$
  else if $|\sigma_L\lceil_{A_v(SN)}| > LPM$ **then**
    $U\_fitness(\sigma_L, M) \leftarrow 1 - \frac{|\sigma_L\lceil_{A_v(SN)}| - LPM + \kappa(\sigma_L)}{|\sigma_L| + \min_{\sigma_M \in \phi_v(SN)}(|\sigma_M|)}$
  else
    $U\_fitness(\sigma_L, M) \leftarrow 1 - \frac{\kappa(\sigma_L)}{|\sigma_L| + \min_{\sigma_M \in \phi_v(SN)}(|\sigma_M|)}$
  end if
end for
return $U\_fitness(\sigma_L, M)$

VI. Evaluation

In this section, we assess the accuracy and time performance of our proposed log selection methods compared to frequency-based and K-Medoids techniques, and evaluate their differences in accuracy and time against normal alignment. Note that the comparison between model behaviour sampling and other approximate methods has been discussed in [5], we focus here on comparisons with the baselines of model behaviour sampling. First, we briefly describe the implementation (Section VI-A) and experimental setup (Section VI-B), followed by a discussion of the experimental results (Section VII).

A. Implementation

Our implementation consists of two steps: first, we implemented the algorithms described in Sections V-A and V-B in Python, to generate log behaviour subsets from event logs. Specifically, we extended the pm4py.algo.clustering package in PM4py [32] by introducing the normalized weighted Levenshtein distance (Definition 6), to perform hierarchical clustering. And implemented two proposed in-cluster methods to get the log behaviour subset based on the clustering result. In the second step, we used an existing plugin in ProM [33], *Conformance Log to Log Approximation* [34], with the generated model behaviour subset and the original event log as input, obtaining approximate fitness bounds and values. For the baselines, we used the implementation proposed by Fanisani [5]. For normal alignment, we used PM4py to compute the time and fitness values. The source code and experimental results is available on Github [1].

B. Experimental Setup

Our experiments were based on six real event logs, with basic information about these event logs given in Table 4. Here, *Uniqueness* refers to $\frac{\text{Variant\#}}{\text{Trace\#}}$. A *Uniqueness* value close to 1 indicates that almost all traces are different, e.g., *Sepsis*. For process discovery, we used *Inductive Miner infrequent algorithm* [35] with *infrequent thresholds of 0.4* to get the process model. Two log selection methods, *frequency-based sampling*, *K-Medoids clustering*, were used as baselines to compare with our proposed methods, i.e., *In-cluster frequency method* and *In-cluster medoid method*. Furthermore, we set the *selection percentage* to 10%, 20%, 30%, 40%, and 50%, representing the ratio of the selected variants to the total number of variants in the event logs. Our experiment was repeated four times since the conformance approximation time is non-deterministic. Finally, we performed the experiments on a computer with Apple M1 (8 cores), 8 GB RAM running macOS.

TABLE 4. THE REAL-LIFE EVENT LOGS USED IN THE EXPERIMENTS

Event Log	Activities #	Traces #	Variants #	Uniqueness
BPIC2012 [36]	25	13087	4366	0.33
BPIC2013-closed problems [37]	4	1487	183	0.12
BPIC2016-Questions [38]	8	21533	2261	0.10
BPIC2017 [39]	28	31509	15930	0.51
Spesis [40]	18	1050	846	0.81
RTFMP [41]	13	150370	231	0.01

1) Evaluation Metrics: To measure approximation accuracy, we used *Approximate Error*, defined as $ApproximateError = |ActualFitness - ApproximateFitness|$, where a value closer to 0 indicates higher accuracy. Additionally, we assess the *Bound Width* as $BoundWidth = U\_fitness - L\_fitness$, with a smaller width indicating tighter bounds and a more accurate approximation.

We used the *Performance Improvement* (PI) metric, defined as $PI = \frac{\text{Actual Conformance Time}}{\text{Approximate Conformance Time}}$ to assess time performance. *Actual Conformance Time* refers to the time needed to compute normal alignment, while *Approximate Conformance Time* includes the total time for the approximation. A PI value greater than 1 indicates the approximation is faster than the actual conformance computation. Preprocessing time (e.g., hierarchical clustering) is included in the approximate conformance time.

[1] https://github.com/lvyl9909/Approximate-Conformance-Checking-using-Hierarchical-Clustering.git

VII. RESULT

Table 5 presents the *Actual Fitness* and *Approximate Fitness*, *Approximate Error*, and *PI* for four selection methods using 20% of the variants in six event logs. For each metric in a given row, the best value is highlighted in bold. The results show that the proposed in-cluster methods achieve the highest *fitness* and the lowest *approximate error* in most cases, indicating superior accuracy compared to the baselines. In terms of *PI*, the frequency-based method consistently achieves the highest values, reflecting its shorter approximate time. Our complete experimental data is provided in Appendix B.

Figure 4 shows that both *Approximate Error* and *Bound Width* decrease as the selection percentages increase. Here, *Bound Width* is represented by bars, and *Approximate Error* by lines, illustrating the improvements in these metrics as the selection percentage increases. Our in-cluster methods consistently achieve tighter bounds at each selection percentage. Notably, at a selection 50% in the BPIC2017 log, the bound widths of the baseline are around 0.05, while our methods reduce this by 40% to 0.03. Furthermore, in all data sets with different selection percentages, the in-cluster frequency method shows an average improvement of 19.1% in *Approximate Error* compared to the frequency-based method, while the in-cluster medoid method achieves an average improvement of 27.6% compared to the K-Medoid method. Moreover, the in-cluster frequency method often produces tighter bounds than in-cluster medoid method, especially on low *uniqueness* logs like BPIC2016-Questions, where selecting the most frequent trace is more effective than clustering. However, on high *Uniqueness* logs like *Sepsis*, in-cluster medoid method provides more accurate approximations. In Figure 5, we compare the time performance of different log selection methods and their improvement over normal alignment. Note that a value of 1 represents the normal alignment time. Consistent with the results in Table 5, the frequency method usually yields the highest performance improvement, followed by the K-Medoids method. Our methods are less efficient compared to these baselines, particularly on datasets with higher *Uniqueness* values.

VIII. DISCUSSION

Across Table 5 and Figure 4, our in-cluster methods consistently achieve higher *fitness*, lower *approximate error*, and tighter bounds than the baselines, with the in-cluster frequency method performing better on low-*Uniqueness* logs (e.g., BPIC2016-Questions) and the

TABLE 5. APPROXIMATE RESULT COMPARISONS (20% SELECTION) FOR FOUR DIFFERENT SELECTION METHODS.

Event Log	Actual	Frequency			K-Medoids			In-cluster freq.			In-cluster medoid		
		Fit.	Err.	PI	Fit.	Err.	PI	Fit.	Err.	PI	Fit.	Err.	PI
BPIC2012	0.9995	0.9741	0.0254	**61.8496**	0.9761	0.0234	41.1727	0.9788	0.0207	25.6113	**0.9806**	**0.0189**	24.8483
BPIC2013-closed problems	0.9997	0.9860	0.0138	**11.8502**	0.9711	0.0286	5.8732	**0.9894**	**0.0103**	1.6728	0.9875	0.0122	1.6443
BPIC2016-Questions	0.9997	0.9923	0.0074	**45.3310**	0.9463	0.0535	30.4731	**0.9944**	**0.0053**	13.1973	0.9565	0.0432	12.2026
BPIC2017	0.9995	0.9690	0.0305	**11.8531**	0.9700	0.0296	9.7231	**0.9749**	**0.0246**	1.9688	0.9747	0.0248	1.8838
Road	0.9999	0.9997	0.0002	**15.7220**	0.9996	0.0004	11.7262	**0.9998**	**0.0001**	7.5686	0.9995	0.0004	6.7700
Sepsis	0.9880	0.9202	0.0679	**53.4338**	0.9202	0.0678	44.9919	0.9313	0.0567	22.9238	**0.9319**	**0.0561**	20.0751

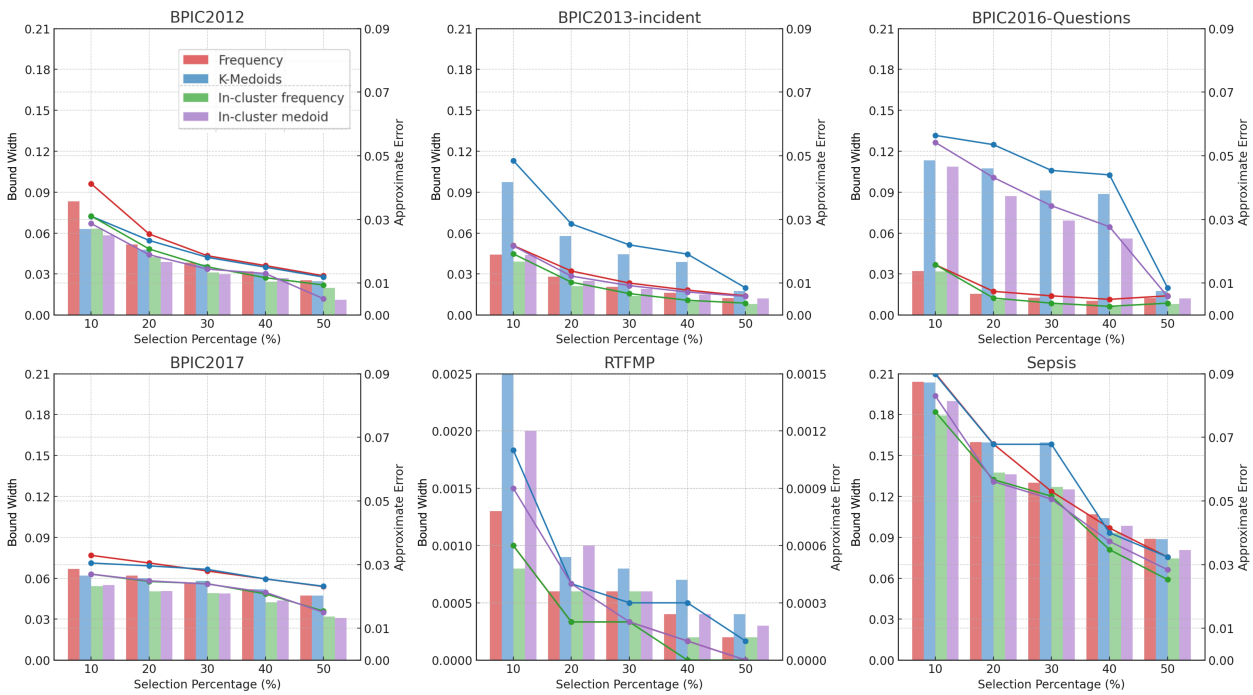

Fig. 4. The performance differences of different selection strategies on band width and approximate error.

in-cluster medoid method excelling on high-*Uniqueness* logs such as *Sepsis*, highlighting a key advantage of our approach over the baselines—improved approximation accuracy. Figure 5 shows that our methods have larger approximation times. This is because hierarchical clustering requires step-by-step merging and evaluating all possible cluster combinations, which increases preprocessing time compared to baselines. Nevertheless, they remain significantly faster than the normal alignment-based approach, keeping approximation times within acceptable limits while delivering higher accuracy—making them well-suited for large-scale processes where neither a quick, coarse estimate nor weeks of exact computation is desirable. Overall, our results indicate a clear trade-off: the proposed methods bring the approximations closer to the actual values at the cost of some additional but acceptable preprocessing time.

IX. CONCLUSION

In this paper, we propose an enhanced model behaviour sampling method using hierarchical clustering to construct more representative model behaviour subsets. By incorporating both frequency and control-flow information from the event log, our approach more effectively captures the model's behaviour, leading to improved approximation accuracy. Experimental results show that our method produces approximations that are on average over 19.1% closer to the actual alignment values than baseline methods, though it requires more computation time.

A potential limitation of this study is the lack of an explicit quantification of how much ”increased” time would be acceptable for the ”improvement” in accuracy, which is important to evaluate the practical utility of the method under different application scenarios. As a next step, we plan to conduct a systematic, quantitative analysis of the accuracy–time trade-off. Based on it, an incremental approximation tool could be developed to increase the size of model behaviour during the time, allowing the user to decide when the accuracy is enough. In addition, we plan to apply a time-optimized hierarchical clustering algorithm to reduce the approximation

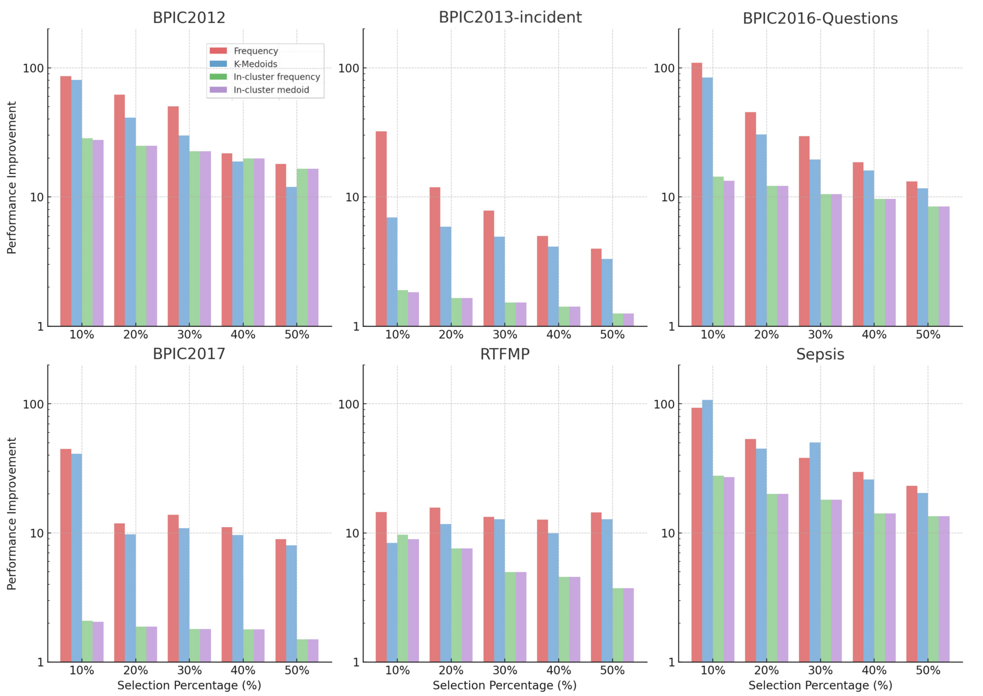

Fig. 5. The performance improvement using different methods in six event logs

time of our method. Furthermore, exploring how to make use of the distribution information (e.g., *Uniqueness*) in the event log to choose a better approximate method is also a direction for future research.

Acknowledgment

The authors would like to thank Artem Polyvyanyy for his invaluable guidance and support throughout this research project.

Declaration of competing interest

The author declares that there are no known financial interests or personal relationships that could have influenced the research presented in this paper.

References

[1] A. A. Mitsyuk, I. A. Lomazova, I. S. Shugurov, and W. M. van der Aalst. Process model repair by detecting unfitting fragments? In *Proceedings of the 6th International Conference on Analysis of Images, Social Networks and Texts (AIST 2017)*, pages 301–313. CEUR-WS. org, January 2017.

[2] Wil Van der Aalst, Arya Adriansyah, and Boudewijn Van Dongen. Replaying history on process models for conformance checking and performance analysis. *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery*, 2(2):182–192, 2012.

[3] J Carmona, B Dongen, A Solti, and M Weidlich. Conformance checking: Relating processes and models, january 2018.

[4] Massimiliano De Leoni and Wil MP Van Der Aalst. Aligning event logs and process models for multi-perspective conformance checking: An approach based on integer linear programming. In *Business Process Management: 11th International Conference, BPM 2013, Beijing, China, August 26-30, 2013. Proceedings*, pages 113–129. Springer, 2013.

[5] M. Fani Sani, S.J. van Zelst, and W.M.P. van der Aalst. Conformance checking approximation using subset selection and edit distance. In *In: Dustdar, S., Yu, E., Salinesi, C., Rieu, D., Pant, V. (eds) CAiSE 2020. LNCS*, volume 12127, pages 234–251. Springer, 2020.

[6] Alifah Syamsiyah and Boudewijn F van Dongen. Improving alignment computation using model-based preprocessing. In *2019 International Conference on Process Mining (ICPM)*, pages 73–80. IEEE, 2019.

[7] Farbod Taymouri and Josep Carmona. A recursive paradigm for aligning observed behavior of large structured process models. In *International Conference on Business Process Management*, pages 197–214. Springer, 2016.

[8] Jorge Munoz-Gama, Josep Carmona, and Wil MP Van Der Aalst. Single-entry single-exit decomposed conformance checking. *Information Systems*, 46:102–122, 2014.

[9] Alifah Syamsiyah and Boudewijn F van Dongen. Improving alignment computation using model-based preprocessing. In *2019 International Conference on Process Mining (ICPM)*, pages 73–80. IEEE, 2019.

[10] Martin Bauer, Han Van der Aa, and Matthias Weidlich. Estimating process conformance by trace sampling and result approximation. In *Business Process Management: 17th International Conference, BPM 2019, Vienna, Austria, September 1–6, 2019, Proceedings 17*, pages 179–197. Springer, 2019.

[11] Martin Bauer, Han van der Aa, and Matthias Weidlich. Sampling and approximation techniques for efficient process conformance checking. *Information Systems*, 104:101666, 2022.

[12] M.F. Sani, M. Kabierski, S.J. Van Zelst, et al. Model independent

error bound estimation for conformance checking approximation. Preprint on arXiv, 2021. Available at https://arxiv.org/abs/2103.13315.

[13] M. Fani Sani, J.J. Garza Gonzalez, S.J. van Zelst, and W.M.P. van der Aalst. Conformance checking approximation using simulation. In *2nd International Conference on Process Mining, ICPM 2020*, pages 105–112, Padua, Italy, 2020. IEEE.

[14] Mohammadreza Fani Sani, Mathilde Boltenhagen, and Wil van der Aalst. Prototype selection using clustering and conformance metrics for process discovery. In *Business Process Management Workshops: BPM 2020 International Workshops, Seville, Spain, September 13–18, 2020, Revised Selected Papers 18*, pages 281–294. Springer, 2020.

[15] Mohammadreza Fani Sani, Mathilde Boltenhagen, and Wil van der Aalst. Prototype selection using clustering and conformance metrics for process discovery. In *Business Process Management Workshops: BPM 2020 International Workshops, Seville, Spain, September 13–18, 2020, Revised Selected Papers 18*, pages 281–294. Springer, 2020.

[16] Sander JJ Leemans, Dirk Fahland, and Wil MP Van Der Aalst. Discovering block-structured process models from event logs containing infrequent behaviour. In *Business Process Management Workshops: BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers 11*, pages 66–78. Springer, 2014.

[17] Farbod Taymouri and Josep Carmona. An evolutionary technique to approximate multiple optimal alignments. In *Business Process Management: 16th International Conference, BPM 2018, Sydney, NSW, Australia, September 9–14, 2018, Proceedings 16*, pages 215–232. Springer International Publishing, 2018.

[18] Farbod Taymouri and Josep Carmona. Computing alignments of well-formed process models using local search. *ACM Transactions on Software Engineering and Methodology (TOSEM)*, 29(3):1–41, 2020.

[19] Jorge Munoz-Gama, Josep Carmona, and Wil M. P. Van Der Aalst. Single-entry single-exit decomposed conformance checking. *Information Systems*, 46:102–122, 2014.

[20] Wil M. P. Van der Aalst. Decomposing petri nets for process mining: A generic approach. *Distributed and Parallel Databases*, 31:471–507, 2013.

[21] Sander JJ Leemans, Dirk Fahland, and Wil MP Van der Aalst. Scalable process discovery and conformance checking. *Software & Systems Modeling*, 17:599–631, 2018.

[22] Daniel Reißner, Abel Armas-Cervantes, Raffaele Conforti, Marlon Dumas, Dirk Fahland, and Marcello La Rosa. Scalable alignment of process models and event logs: An approach based on automata and s-components. *Information Systems*, 94:101561, 2020.

[23] Yilin Lyu. A mamba-based approximate conformance checking method. In *Fourth International Conference on Advanced Algorithms and Neural Networks (AANN 2024)*, volume 13416, pages 885–891. SPIE, 2024.

[24] Jari Peeperkorn, Seppe vanden Broucke, and Jochen De Weerdt. Supervised conformance checking using recurrent neural network classifiers. In *International Conference on Process Mining*, pages 175–187. Springer, 2020.

[25] Matthias Bauer, Hilde van der Aa, and Matthias Weidlich. Sampling and approximation techniques for efficient process conformance checking. *Information Systems*, 104:101666, 2022.

[26] Jae-Yoon Jung, Joonsoo Bae, and Ling Liu. Hierarchical clustering of business process models. *International Journal of Innovative Computing, Information and Control*, 5(12):1349–4198, 2009.

[27] S.J. van Zelst and Y. Cao. A generic framework for attribute-driven hierarchical trace clustering. In A. Del Río Ortega, H. Leopold, and F.M. Santoro, editors, *Business Process Management Workshops. BPM 2020*, volume 397 of *Lecture Notes in Business Information Processing*. Springer, Cham, 2020.

[28] Wil M. P. Van der Aalst. Decomposing petri nets for process mining: A generic approach. *Distributed and Parallel Databases*, 31(4):471–507, 2013.

[29] V Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Proceedings of the Soviet physics doklady*, 1966.

[30] Peter H. Sellers. On the theory and computation of evolutionary distances. *SIAM Journal on Applied Mathematics*, 26(4):787–793, 1974.

[31] Daniel Müllner. Modern hierarchical, agglomerative clustering algorithms. *arXiv preprint arXiv:1109.2378*, 2011.

[32] Alessandro Berti, Sebastiaan van Zelst, and Daniel Schuster. Pm4py: A process mining library for python. *Software Impacts*, 17:100556, 2023.

[33] HMW Verbeek, JCAM Buijs, BF Van Dongen, and Wil MP van der Aalst. Prom 6: The process mining toolkit. *Proc. of BPM Demonstration Track*, 615:34–39, 2010.

[34] Mohammadreza Fani Sani, Juan J Garza Gonzalez, Sebastiaan J van Zelst, and Wil MP van der Aalst. Alignment approximator: A prom plug-in to approximate conformance statistics. In *BPM (Demos/Resources Forum)*, pages 102–106, 2023.

[35] Sander JJ Leemans, Dirk Fahland, and Wil MP Van Der Aalst. Discovering block-structured process models from event logs containing infrequent behaviour. In *Business Process Management Workshops: BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers 11*, pages 66–78. Springer, 2014.

[36] Boudewijn van Dongen. Bpi challenge 2012, 2012.

[37] Ward Steeman. Bpi challenge 2013, closed problems, 2013.

[38] M. Dees and Boudewijn van Dongen. Bpi challenge 2016: Questions, 2016.

[39] Boudewijn van Dongen. Bpi challenge 2017, 2017.

[40] Felix Mannhardt. Sepsis cases - event log, 2016.

[41] M. (Massimiliano) de Leoni and Felix Mannhardt. Road traffic fine management process, 2015.

APPENDIX

A. Proof of Alignment Cost Upper Bound

Proof: We have shown that $\min_{\sigma_M \in S} \Delta(\sigma_L, \sigma_M) = \delta_S(\gamma_{SN}(\sigma_L))$ in Definition 5, so $\Delta(\sigma_L, \sigma_M) \geq \delta_S(\gamma_{SN}(\sigma_L))$. Therefore, if $\delta_S(\gamma_{SN}(\sigma_L)) > \Delta(\sigma_L, \sigma_M)$, $\gamma_{SN}(\sigma_L)$ is not an optimal alignment. Consequently, for any $M_B \subseteq \phi_v(SN)$, $\Phi(\sigma_L, M_B)$ returns an upper bound for the cost of optimal alignment [5]. ■

B. Proof of Alignment Cost Lower Bound

Proof: When $|\sigma_L\lceil_{A_v(SN)}| < SPM$, at least $SPM - |\sigma_L\lceil_{A_v(SN)}|$ insertions are needed. Adding the initial alignment cost, the total minimum alignment cost is $|SPM - \sigma_L\lceil_{A_v(SN)}| + |\kappa(\sigma_L)|$. Similarly, when $|\sigma_L\lceil_{A_v(SN)}| > LPM$, at least $|\sigma_L\lceil_{A_v(SN)}| - LPM$ deletions are required. Thus, the total alignment cost is $|\sigma_L\lceil_{A_v(SN)} - LPM| + |\kappa(\sigma_L)|$. When $SPM \leq |\sigma_L\lceil_{A_v(SN)}| \leq LPM$, no insertions or deletions are needed, so the alignment cost is $|\kappa(\sigma_L)|$. ■

TABLE 6: Experimental results for datasets.

Log	Actual Fitness	Normal Alignment Time	Candidate Percentage	Parameter		Approximation Method: Baseline: Frequency	Approximation Method: Baseline: K-Medoids	Approximation Method: In-cluster medoid: In-cluster frequency	Approximation Method: In-cluster medoid: In-cluster medoid
BPIC2012	0.9995	35400000	10%	Approximate fitness	Lower Bound	0.9167	0.9371	0.9368	0.9416
					Approximate fitness	0.9583	0.9685	0.9684	0.9708
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0412	0.0310	0.0311	0.0287
				Band Width		0.0833	0.0629	0.0632	0.0584
				Preprocessing Time (ms)		/	/	1219923	1259201
				Approximate Time (ms)		411778	439928	25030	26102
				Total Approximate Time (ms)		411778	439928	1244953	1285303
				PI		85.9687	80.4677	28.4348	27.5421
			20%	Approximate fitness	Lower Bound	0.9482	0.9522	0.9576	0.9612
					Approximate fitness	0.9741	0.9761	0.9788	0.9806
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0254	0.0234	0.0207	0.0189
				Band Width		0.0518	0.0478	0.0424	0.0388
				Preprocessing Time (ms)		/	/	1342972	1392321
				Approximate Time (ms)		572356	859792	39232	32323
				Total Approximate Time (ms)		572356	859792	1382204	1424644
				PI		61.8496	41.1727	25.6113	24.8483
			30%	Approximate fitness	Lower Bound	0.9618	0.9629	0.9688	0.9702
					Approximate fitness	0.9809	0.9814	0.9844	0.9851
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0186	0.0181	0.0151	0.0144
				Band Width		0.0382	0.0371	0.0312	0.0298
				Preprocessing Time (ms)		/	/	1423219	1529312
				Approximate Time (ms)		702244	1186892	41992	42223
				Total Approximate Time (ms)		702244	1186892	1465211	1571535
				PI		50.4098	29.8258	24.1603	22.5257
			40%	Approximate fitness	Lower Bound	0.9681	0.9690	0.9756	0.9730
					Approximate fitness	0.9841	0.9845	0.9878	0.9865
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0155	0.0150	0.0117	0.0130
				Band Width		0.0319	0.0310	0.0244	0.0270
				Preprocessing Time (ms)		/	/	1591211	1730030
				Approximate Time (ms)		1229401	1480757	41503	49020
				Total Approximate Time (ms)		1229401	1480757	1632714	1779050
				PI		28.7945	23.9067	21.6817	19.8983
			50%	Approximate fitness	Lower Bound	0.9745	0.9752	0.9802	0.9888
					Approximate fitness	0.9873	0.9876	0.9901	0.9944
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0123	0.0119	0.0094	0.0051
				Band Width		0.0255	0.0248	0.0198	0.0112
				Preprocessing Time (ms)		/	/	1823900	2102097
				Approximate Time (ms)		1863573	1971131	42826	43503
				Total Approximate Time (ms)		1863573	1971131	1866726	2145600
				PI		18.9958	17.9592	18.9637	16.4989

TABLE 6: Experimental results for datasets.

Table 6 continued.

Log	Actual Fitness	Normal Alignment Time	Candidate Percentage	Parameter		Approximation Method			
						Baseline		In-cluster medoid	
						Frequency	K-Medoids	In-cluster frequency	In-cluster medoid
BPIC2013-incident	0.9997	135400	10%	Approximate fitness	Lower Bound	0.9559	0.9025	0.9610	0.9560
					Approximate fitness	0.9780	0.9513	0.9805	0.9780
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0218	0.0485	0.0192	0.0217
				Band Width		0.0441	0.0975	0.0390	0.0440
				Preprocessing Time (ms)		/	/	69233	70923
				Approximate Time (ms)		4200	19572	2033	2992
				Total Approximate Time (ms)		4200	19572	71266	73915
				PI		32.2381	6.9180	1.8999	1.8318
			20%	Approximate fitness	Lower Bound	0.9719	0.9422	0.9788	0.9750
					Approximate fitness	0.9860	0.9711	0.9894	0.9875
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0138	0.0286	0.0103	0.0122
				Band Width		0.0281	0.0578	0.0212	0.0250
				Preprocessing Time (ms)		/	/	78012	79232
				Approximate Time (ms)		11426	23054	2932	3111
				Total Approximate Time (ms)		11426	23054	80944	82343
				PI		11.8502	5.8732	1.6728	1.6443
			30%	Approximate fitness	Lower Bound	0.9795	0.9554	0.9860	0.9810
					Approximate fitness	0.9898	0.9777	0.9930	0.9905
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0100	0.0220	0.0067	0.0092
				Band Width		0.0205	0.0446	0.0140	0.0190
				Preprocessing Time (ms)		/	/	81203	85003
				Approximate Time (ms)		17294	27553	3504	4092
				Total Approximate Time (ms)		17294	27553	84707	89095
				PI		7.8293	4.9142	1.5985	1.5197
			40%	Approximate fitness	Lower Bound	0.9839	0.9612	0.9902	0.9850
					Approximate fitness	0.9920	0.9806	0.9951	0.9925
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0078	0.0191	0.0046	0.0072
				Band Width		0.0161	0.0388	0.0098	0.0150
				Preprocessing Time (ms)		/	/	89129	91892
				Approximate Time (ms)		27133	32868	3932	3902
				Total Approximate Time (ms)		27133	32868	93061	95794
				PI		4.9902	4.1195	1.4550	1.4134
			50%	Approximate fitness	Lower Bound	0.9875	0.9825	0.9920	0.9879
					Approximate fitness	0.9938	0.9913	0.9960	0.9940
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0060	0.0085	0.0037	0.0058
				Band Width		0.0125	0.0175	0.0080	0.0121
				Preprocessing Time (ms)		/	/	95002	104023
				Approximate Time (ms)		34006	41028	4002	4350
				Total Approximate Time (ms)		34006	41028	99004	108373
				PI		3.9817	3.3002	1.3676	1.2494

TABLE 6: Experimental results for datasets.

Table 6 continued.

Log	Actual Fitness	Normal Alignment Time	Candidate Percentage	Parameter		Approximation Method: Baseline: Frequency	Approximation Method: Baseline: K-Medoids	Approximation Method: In-cluster medoid: In-cluster frequency	Approximation Method: In-cluster medoid: In-cluster medoid
BPIC2016-Questions	0.9997	5200690	10%	Approximate fitness	Lower Bound	0.9679	0.8867	0.9680	0.8911
					Approximate fitness	0.9840	0.9434	0.9840	0.9455
					Upper Bound	1.0000	1.0000	0.9999	0.9999
				Approximation Error		0.0158	0.0564	0.0158	0.0542
				Band Width		0.0321	0.1133	0.0319	0.1088
				Preprocessing Time(ms)		/	/	359923	389454
				Approximate Time(ms)		47607	61807	2715	1551
				Total Approximate Time(ms)		47607	61807	362638	391005
				PI		109.2421	84.1440	14.3413	13.3008
			20%	Approximate fitness	Lower Bound	0.9845	0.8925	0.9888	0.9130
					Approximate fitness	0.9923	0.9463	0.9944	0.9565
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0074	0.0535	0.0053	0.0432
				Band Width		0.0155	0.1075	0.0112	0.0870
				Preprocessing Time(ms)		/	/	390239	421292
				Approximate Time(ms)		114727	170665	3832	4902
				Total Approximate Time(ms)		114727	170665	394071	426194
				PI		45.3310	30.4731	13.1973	12.2026
			30%	Approximate fitness	Lower Bound	0.9874	0.9087	0.9920	0.9309
					Approximate fitness	0.9937	0.9544	0.9960	0.9655
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0060	0.0454	0.0037	0.0343
				Band Width		0.0126	0.0913	0.0080	0.0691
				Preprocessing Time(ms)		/	/	448922	489322
				Approximate Time(ms)		176359	266266	6020	6334
				Total Approximate Time(ms)		176359	266266	454942	495656
				PI		29.4892	19.5319	11.4315	10.4925
			40%	Approximate fitness	Lower Bound	0.9896	0.9114	0.9940	0.9440
					Approximate fitness	0.9948	0.9557	0.9970	0.9720
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0049	0.0440	0.0027	0.0277
				Band Width		0.0104	0.0886	0.0060	0.0560
				Preprocessing Time(ms)		/	/	483200	530239
				Approximate Time(ms)		280456	325313	9910	10355
				Total Approximate Time(ms)		280456	325313	493110	540594
				PI		18.5437	15.9867	10.5467	9.6203
			50%	Approximate fitness	Lower Bound	0.9913	0.9294	0.9960	0.9503
					Approximate fitness	0.9957	0.9647	0.9980	0.9752
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0060	0.0085	0.0037	0.0058
				Band Width		0.0125	0.0175	0.0080	0.0121
				Preprocessing Time(ms)		/	/	566660	602030
				Approximate Time(ms)		395799	445163	15330	14340
				Total Approximate Time(ms)		395799	445163	581990	616370
				PI		13.1397	11.6827	8.9360	8.4376

TABLE 6: Experimental results for datasets.

Table 6 continued.

Log	Actual Fitness	Normal Alignment Time	Candidate Percentage	Parameter		Approximation Method			
						Baseline		In-cluster medoid	
						Frequency	K-Medoids	In-cluster frequency	In-cluster medoid
BPIC2017	0.9995	180829300	10%	Approximate fitness	Lower Bound	0.9332	0.9381	0.9454	0.9450
					Approximate fitness	0.9666	0.9691	0.9726	0.9725
					Upper Bound	1.0000	1.0000	0.9997	1.0000
				Approximation Error		0.0329	0.0305	0.0270	0.0270
				Band Width		0.0668	0.0619	0.0543	0.0550
				Preprocessing Time (ms)		/	/	86490212	87983292
				Approximate Time (ms)		4049416	4399280	400366	509232
				Total Approximate Time (ms)		4049416	4399280	86890578	88492524
				PI		44.6556	41.1043	2.0811	2.0434
			20%	Approximate fitness	Lower Bound	0.9380	0.9399	0.9497	0.9493
					Approximate fitness	0.9690	0.9700	0.9749	0.9747
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0305	0.0296	0.0247	0.0249
				Band Width		0.0620	0.0601	0.0503	0.0507
				Preprocessing Time(ms)		/	/	91423432	95431122
				Approximate Time(ms)		15255832	18597920	424210	561543
				Total Approximate Time(ms)		15255832	18597920	91847642	95992665
				PI		11.8531	9.7231	1.9688	1.8838
			30%	Approximate fitness	Lower Bound	0.9431	0.9420	0.9510	0.9512
					Approximate fitness	0.9715	0.9710	0.9755	0.9756
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0280	0.0285	0.0240	0.0239
				Band Width		0.0569	0.0580	0.0490	0.0488
				Preprocessing Time(ms)		/	/	95294232	99874342
				Approximate Time(ms)		13089388	16606568	502321	424931
				Total Approximate Time(ms)		13089388	16606568	95796553	100299273
				PI		13.8150	10.8890	1.8876	1.8029
			40%	Approximate fitness	Lower Bound	0.9481	0.9480	0.9575	0.9564
					Approximate fitness	0.9741	0.9740	0.9788	0.9782
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0255	0.0255	0.0208	0.0213
				Band Width		0.0519	0.0520	0.0425	0.0436
				Preprocessing Time(ms)		/	/	99034313	100293122
				Approximate Time(ms)		16294010	18807577	582312	510124
				Total Approximate Time(ms)		16294010	18807577	99616625	100803246
				PI		11.0979	9.6147	1.8153	1.7939
			50%	Approximate fitness	Lower Bound	0.9528	0.9527	0.9682	0.9691
					Approximate fitness	0.9764	0.9764	0.9841	0.9846
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0231	0.0232	0.0154	0.0150
				Band Width		0.0472	0.0473	0.0318	0.0309
				Preprocessing Time(ms)		/	/	108224313	119901232
				Approximate Time(ms)		20183838	22539508	391222	454002
				Total Approximate Time(ms)		20183838	22539508	108615535	120355234
				PI		8.9591	8.0228	1.6649	1.5025

TABLE 6: Experimental results for datasets.

Table 6 continued.

Log	Actual Fitness	Normal Alignment Time	Candidate Percentage	Parameter		Approximation Method			
						Baseline		In-cluster medoid	
						Frequency	K-Medoids	In-cluster frequency	In-cluster medoid
RTFMP	0.9999	130430	10%	Approximate fitness	Lower Bound	0.9987	0.9975	0.9989	0.9980
					Approximate fitness	0.9994	0.9988	0.9993	0.9990
					Upper Bound	1.0000	1.0000	0.9997	1.0000
				Approximation Error		0.0006	0.0011	0.0006	0.0009
				Band Width		0.0013	0.0025	0.0008	0.0020
				Preprocessing Time(ms)		/	/	10585	11021
				Approximate Time(ms)		8986	15555	2901	3531
				Total Approximate Time(ms)		8986	15555	13486	14552
				PI		14.5148	8.3851	9.6715	8.9630
			20%	Approximate fitness	Lower Bound	0.9994	0.9991	0.9994	0.9990
					Approximate fitness	0.9997	0.9996	0.9997	0.9995
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0002	0.0004	0.0002	0.0004
				Band Width		0.0006	0.0009	0.0006	0.0010
				Preprocessing Time(ms)		/	/	14012	15432
				Approximate Time(ms)		8296	11123	3221	3834
				Total Approximate Time(ms)		8296	11123	17233	19266
				PI		15.7220	11.7262	7.5686	6.7700
			30%	Approximate fitness	Lower Bound	0.9994	0.9992	0.9994	0.9994
					Approximate fitness	0.9997	0.9996	0.9997	0.9997
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0002	0.0003	0.0002	0.0002
				Band Width		0.0006	0.0008	0.0006	0.0006
				Preprocessing Time(ms)		/	/	15236	22293
				Approximate Time(ms)		9831	10222	3232	3923
				Total Approximate Time(ms)		9831	10222	18468	26216
				PI		13.2672	12.7597	7.0625	4.9752
			40%	Approximate fitness	Lower Bound	0.9996	0.9993	0.9998	0.9996
					Approximate fitness	0.9998	0.9997	0.9999	0.9998
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0001	0.0003	0.0000	0.0001
				Band Width		0.0004	0.0007	0.0002	0.0004
				Preprocessing Time(ms)		/	/	17222	24422
				Approximate Time(ms)		10323	13123	4442	4232
				Total Approximate Time(ms)		10323	13123	21664	28654
				PI		12.6349	9.9390	6.0206	4.5519
			50%	Approximate fitness	Lower Bound	0.9998	0.9996	0.9998	0.9997
					Approximate fitness	0.9999	0.9998	0.9999	0.9999
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0000	0.0001	0.0000	0.0000
				Band Width		0.0002	0.0004	0.0002	0.0003
				Preprocessing Time(ms)		/	/	19203	30020
				Approximate Time(ms)		9050	10212	4301	5021
				Total Approximate Time(ms)		9050	10212	23504	35041
				PI		14.4122	12.7722	5.5493	3.7222

TABLE 6: Experimental results for datasets.

Table 6 continued.

Log	Actual Fitness	Normal Alignment Time	Candidate Percentage	Parameter		Approximation Method: Baseline – Frequency	Approximation Method: Baseline – K-Medoids	Approximation Method: In-cluster medoid – In-cluster frequency	Approximation Method: In-cluster medoid – In-cluster medoid
Sepsis	0.9880	3035200	10%	Approximate fitness	Lower Bound	0.7959	0.7965	0.8204	0.8100
					Approximate fitness	0.8980	0.8983	0.9101	0.9050
					Upper Bound	1.0000	1.0000	0.9997	1.0000
				Approximation Error		0.0901	0.0898	0.0780	0.0830
				Band Width		0.2041	0.2035	0.1793	0.1900
				Preprocessing Time(ms)		/	/	107478	110312
				Approximate Time(ms)		32599	28302	1902	2032
				Total Approximate Time(ms)		32599	28302	109380	112344
				PI		93.1072	107.2433	27.7491	27.0170
			20%	Approximate fitness	Lower Bound	0.8403	0.8404	0.8626	0.8638
					Approximate fitness	0.9202	0.9202	0.9313	0.9319
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0679	0.0678	0.0567	0.0561
				Band Width		0.1597	0.1596	0.1374	0.1362
				Preprocessing Time(ms)		/	/	130101	148903
				Approximate Time(ms)		56803	67461	2303	2289
				Total Approximate Time(ms)		56803	67461	132404	151192
				PI		53.4338	44.9919	22.9238	20.0751
			30%	Approximate fitness	Lower Bound	0.8701	0.8405	0.8730	0.8748
					Approximate fitness	0.9351	0.9203	0.9365	0.9374
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0530	0.0678	0.0515	0.0506
				Band Width		0.1299	0.1595	0.1270	0.1252
				Preprocessing Time(ms)		/	/	159232	162820
				Approximate Time(ms)		79763	60393	5201	5433
				Total Approximate Time(ms)		79763	60393	164433	168253
				PI		38.0527	50.2575	18.4586	18.0395
			40%	Approximate fitness	Lower Bound	0.8931	0.8959	0.9066	0.9015
					Approximate fitness	0.9466	0.9480	0.9533	0.9508
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0415	0.0400	0.0347	0.0373
				Band Width		0.1069	0.1041	0.0934	0.0985
				Preprocessing Time(ms)		/	/	182782	209212
				Approximate Time(ms)		102649	116824	6123	5736
				Total Approximate Time(ms)		102649	116824	188905	214948
				PI		29.5687	25.9810	16.0673	14.1206
			50%	Approximate fitness	Lower Bound	0.9112	0.9113	0.9255	0.9192
					Approximate fitness	0.9556	0.9557	0.9628	0.9596
					Upper Bound	1.0000	1.0000	1.0000	1.0000
				Approximation Error		0.0324	0.0324	0.0253	0.0284
				Band Width		0.0888	0.0887	0.0745	0.0808
				Preprocessing Time(ms)		/	/	209823	222011
				Approximate Time(ms)		126803	137461	3508	3769
				Total Approximate Time(ms)		126803	137461	213331	225780
				PI		23.9363	22.0804	14.2277	13.4432