Graph-based Similarity Measures for the Structural Comparison of Process Traces

Clemens Schreiber[a] (clemens.schreiber@kit.edu), Amine Abbad-Andaloussi[b] (amine.abbad-andaloussi@unisg.ch), Andrea Burattin[c] (andbur@dtu.dk), Andreas Oberweis[a] (andreas.oberweis@kit.edu), Barbara Weber[b] (barbara.weber@unisg.ch),

[a] Karlsruhe Institute of Technology, Karlsruhe, Germany

[b] University of St. Gallen, St. Gallen, Switzerland

[c] Technical University of Denmark, Lyngby, Denmark

Corresponding Author:

Clemens Schreiber

Kaiserstraße 89, 76133 Karlsruhe, Germany

Tel: +49 721 608-45688

Email: clemens.schreiber@kit.edu

Graph-based Similarity Measures for the Structural Comparison of Process Traces

Clemens Schreiber[a,*], Amine Abbad-Andaloussi[b], Andrea Burattin[c], Andreas Oberweis[a], Barbara Weber[b]

[a] *Karlsruhe Institute of Technology, Karlsruhe, Germany*
[b] *University of St. Gallen, St. Gallen, Switzerland*
[c] *Technical University of Denmark, Lyngby, Denmark*

Abstract

Similarity measures are commonly applied for a variety of process mining techniques, such as trace clustering, conformance checking, and event abstraction. Yet, these measures generally fail to recognize similarity based on structural process features, such as the order of activities, loops, skips, choices, and parallelism. To make this more explicit, we propose a set of properties that allow to evaluate, what kind of structural features are reflected by a similarity measure. We further propose a novel approach leveraging existing graph-based algorithms and instance graphs to extract high-level structural features (loops, skips, choices, and parallelism) from traces, such that they can be used to extend and improve existing similarity measures. These algorithms are well-established in graph theory and can be computed efficiently. Finally, we provide an evaluation of the proposed approach based on synthetic and real-world datasets. The evaluation provides evidence that the additional graph-based features can substantially improve the similarity comparison of traces in several cases. This applies in particular for the comparison of user behavior (e.g., based on eye-tracking data) where structural features enable the detection of specific behavioral patterns.

*Corresponding author.
Email addresses: clemens.schreiber@kit.edu (Clemens Schreiber), amine.abbad-andaloussi@unisg.ch (Amine Abbad-Andaloussi), andbur@dtu.dk (Andrea Burattin), andreas.oberweis@kit.edu (Andreas Oberweis), barbara.weber@unisg.ch (Barbara Weber)

Keywords: Process Mining, Similarity Measure, Trace Variant Analysis

1. Introduction

Pairwise comparison of process traces is essential for many process mining tasks, such as trace clustering [1, 2, 3, 4], conformance checking [5, 6], process discovery [7, 8, 9], event log sampling [10], change point detection [11], and variability analysis [12, 13]. Although the literature comprises a wide range of trace similarity measures (for an overview, see Back and Simonsen [14]), they generally exhibit several limitations.

The first main limitation of existing trace similarity measures is their failure to account for process patterns [7], such as: loops (i.e., the repeated execution of activity sequences), choices (i.e., alternative activity sequences), skips (i.e., alternative activity sequences that involves the omission of activities), and parallelism (i.e., a timely overlap of activity sequences). One reason why these patterns are neglected is that commonly applied similarity measures in process mining, such as edit distance [3, 4, 6, 7, 12, 13] and sequence alignment [5, 8, 9], originate from other research disciplines and were developed for different purposes, such as the comparison of binary code [15] or the comparison of DNA sequences [16]. Yet, the named process patterns are essential for the description of process behavior and for the comparison of the traces' structure [17, 7, 9, 3, 4].

A second shortcoming of existing similarity measures is that they do not allow for an activity-agnostic comparison between traces. When, for example, comparing the two traces $\sigma_1 = \langle A, B, D, A, C, D, A, D \rangle$ and $\sigma_2 = \langle W, X, Z, W, Y, Z, W, Z \rangle$, none of the commonly applied similarity measures in process analysis [14] would identify any similarity between them. However, as illustrated in Fig. 1, the ordering and repetition of activities within the two traces reveal three common process patterns: a loop, a choice, and a skip, each involving activity sequences of identical length.

The structural comparison between traces, independent of their specific activity labels, can be crucial for process analysis across various domains. In this

paper we focus on two particular types of processes: (unstructured) behavioral processes [18, 19] and (well-structured) business processes [20]. *Behavioral processes* document some user behavior, e.g., when interacting with an information system through eye-tracking [21], or click-streams [22]. They commonly contain minimal constraints on the ordering and number of activities. However, the structural differences between the user behavior can provide valuable insights for the analysis of behavioral processes. For example, a trace that represents the scan path of a user's visual fixations on a screen can be analyzed according to the order and reoccurrence of these fixations. Here, the activities of the trace are associated with visual fixations of a user on a particular area on a screen, e.g., representing specific graphical elements [23, 24, 25]. The structural analysis of a scan path can thereby reveal the cognitive processes of a user. For example, the repeated returning of a user to a particular area on a screen, manifested as loops within the scan path, might indicate high cognitive effort due to the distribution of information [23, 24] or due to the ambiguity of the information [25]. In order to obtain these insights, we are much more interested in the structure of the user behavior, such as the returning to particular areas on a screen, rather than what specific areas a user looked at, which is indicated by the activity labels.

Business processes refer to administrative processes, which are more restrictive in terms of the number and order of activities than behavioral processes. Here, the activity labels might differ especially across different business units and countries of operation [20]. Therefore, the comparison of traces at a structural level can support the identification of deviations related to compliance and performance, e.g., based on the rate of rework [26], as well as the level of business process standardization [13]. Another related example is the analysis of business process changes, commonly referred to as concept drift [26]. Business process changes caused by the introduction of new digital technologies, such as automated technology or AI-based technology, can significantly alter the structural properties of a process, while at the same time introducing new activities [27]. Similar to the analysis of behavioral processes, in all these cases

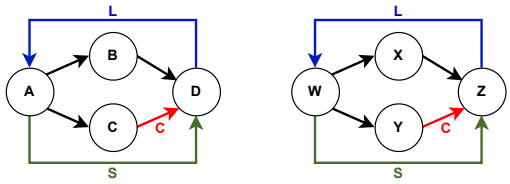

Figure 1: Graph representations of the traces $\sigma_1 = \langle A, B, D, A, C, D, A, D \rangle$ (left side) and $\sigma_2 = \langle W, X, Z, W, Y, Z, W, Z \rangle$ (right side), where a blue edge indicates a loop (denoted by L), a red edge indicates a choice (denoted by C), and a green edge indicates a skip (denoted by S). The graph representations are derived based on the directly-follows relations between the activities within the traces.

the primary focus of the analysis lies on the structural characteristics of the process execution rather than on the actual activities denoted by the activity labels.

A third main shortcoming of existing trace similarity measures is that their derived similarity values are commonly not transparent and are therefore not explainable. This is the case when the structural features, i.e., process patterns, of the traces are not made explicit. For example, when comparing the edit distances $d(\langle A \rangle, \langle A, A, A \rangle)$ and $d(\langle A \rangle, \langle A, X, Y \rangle)$, both distances yield the same value due to the insertions of two additional activities in the second trace. However, in the first case the insertion is necessary due to a loop in the second trace, while in the second case the insertion is necessary due to a sequence of two alternative activities. This distinction is not considered by the similarity measure.

However, such structural differences are for example relevant for the analysis of user behavior, as they can potentially reveal different causes for comprehension issues, requiring specific user support [23, 24, 25]. Moreover, the transparency of similarity measures can be essential to better understand anomalies and deviations in business processes [28, 6].

Finally, a fourth shortcoming is that some similarity measures require high

computational effort [14, 29, 30], making them impractical to apply to large event logs and in online settings, which require timely evaluation [31].

To address the four identified challenges, we first conduct a formal comparative analysis of existing similarity measures, considering: (1) the extent to which the measures capture different process patterns, (2) their ability to recognize activity-agnostic similarity, (3) the degree of transparency and explainability they offer, and (4) their computational efficiency. The findings indicate that none of the existing measures adequately capture the structural similarities and differences between traces, particularly with respect to the representation of process patterns and the transparency of the similarity computation.

Subsequently, we introduce a novel feature-extraction approach based on graph-algorithms, which allows to derive high-level structural trace features, i.e., loops, skips, and choices, including the length of the involved activity sequences. Thus, enabling an activity-agnostic comparison between traces. Furthermore, we leverage instance-graphs [17, 19] to also account for parallelism within traces. The parallel execution of activity sequences is thereby assumed, when activities occur in an interchangeable order within one or multiple traces of a business process.

To account for the diverse structural characteristics of traces identified in the formal comparison, we introduce four aggregated similarity measures. Each aggregation combines distinct similarity measures that reflect distinct structural aspects of the traces, i.e., activities, directly-follows relations, and high-level structural features. The approach therefore enables a more comprehensive assessment of structural similarities and differences between traces than existing measures.

The subsequent empirical evaluation demonstrates that incorporating high-level structural features, as well as aggregating different feature types, can substantially enhance the accuracy of trace similarity comparisons in various cases. The empirical evaluation involves synthetic datasets, designed to exhibit diverse structural characteristics, and real-world datasets that represent business processes and user behavior processes.

The main contributions of this paper are as follows:

1. We introduce ten formal properties, which allow for a comprehensive comparison of similarity measures, considering structural properties, activity-agnostic behavior, transparency, and computational complexity.
2. Drawing upon the identified limitations of current trace similarity measures, we propose a novel feature-extraction approach based on graph algorithms, which enables the detection of loops, skips, and choices, along with their respective lengths within individual traces. Furthermore, we leverage instance graphs to also enable the consideration of parallelism within individual traces.
3. Furthermore, we consider different aggregations of similarity measures, which allows to address structural similarities and differences between traces in a more comprehensive way than previous measures.
4. Finally, we conduct a comprehensive empirical evaluation of the newly introduced similarity measures. We thereby demonstrate the superiority of these measures to existing once in terms of correctly detecting similarity among traces, which share common structural characteristics. We thereby extend the evaluation of existing studies on trace similarity measures, by considering business processes, as well as behavioral processes.

In the remainder, Sect. 2 defines the basic notation and definitions used throughout the paper. In Sect. 3 we introduce related work on trace similarity measures, followed by a formal comparison of existing measures in Sect. 4. Sect. 5 introduces the graph-based approach used to overcome the identified shortcomings of the existing measures, followed by an empirical evaluation in Sect. 6 and a discussion of the results in Sect. 7. Sect. 8 concludes the paper.

2. Preliminaries

Process mining is generally concerned with the analysis of processes based on event logs. An event log comprises a set of process instances that follow a specific business process or, more broadly, represent user interactions with

an information system. A process instance is represented as a trace, which is defined as an ordered sequence of events.

Definition 2.1 (Event). *Let $\mathcal{A}$ be the set of all possible activities, $\mathcal{C}$ the set of all possible case identifiers, and $\mathcal{T}$ the set of all possible timestamps. An event is a tuple $e = (a, c, t)$ consisting of three attributes: an activity $a \in \mathcal{A}$, a case id $c \in \mathcal{C}$, and a timestamp $t \in \mathcal{T}$. The event universe is denoted by $\mathcal{E} = \mathcal{A} \times \mathcal{C} \times \mathcal{T}$.*

We further define the following attribute-value mappings for an event e: $\#_{act}(e) = a$, $\#_{case}(e) = c$, and $\#_{time}(e) = t$. A process trace, henceforth denoted as trace, can be derived by considering multiple events in an ordered fashion based on their timestamp and case identifier.

Definition 2.2 (Trace, Event Log). *A trace $\sigma_c = \langle \#_{act}(e_1), \#_{act}(e_2), \ldots, \#_{act}(e_n) \rangle = \langle a_1, a_2, \ldots, a_n \rangle \in \mathcal{A}^*$ is an event sequence corresponding to the case c, where all events in σ_c are mapped to an activity $a \in A$, and the order of events respects time, i.e., if $e_1, e_2 \in \sigma_c$ and $\#_{time}(e_1) < \#_{time}(e_2)$, then $e_2 \not\prec e_1$. An event log L is a set of traces over $\mathcal{A}^*$.*

The length of a trace is denoted by $|\sigma_c| = n$ and a_k represents the k^{th} activity in a trace σ_c, with $1 \leq k \leq n$.

It is assumed that each event possesses a unique case id, which enables a comparison at trace level. Nevertheless, in reality it is also possible that an event possesses multiple case ids, e.g., in the case of object-centric event logs [32]. In this case it is assumed that the similarity comparison is conducted among identical case notions, e.g., defined by a specific object type [33]. Accordingly, a similarity measures for two distinct process traces $\sigma_i, \sigma_j \in L$ is defined as follows.

Definition 2.3 (Similarity Measure). *A similarity measure $s(\sigma_i, \sigma_j) = (d \circ h)(\sigma_i, \sigma_j)$ is a composition of a distance function d and a feature-extraction function h. The function h defines a set of features for each trace $\sigma \in L$, such that $h : L \to \mathcal{P}(F)$, where $F \subseteq \mathcal{F}$ is a set of possible trace features and $\mathcal{P}(F)$*

denotes its power set. The function d defines a pairwise distance for all distinct traces $\sigma_i, \sigma_j \in L$ *according to the derived features, i.e.,* $d : h(L) \times h(L) \to \mathbb{R}_0^+$.

A similarity measure s and an event log L form a metric space (L, s), if for all $\sigma_i, \sigma_j, \sigma_k \in L$ the following properties hold:

Property P1. $s(\sigma_i, \sigma_j) \geq 0$ (non-negativity)

Property P2. $s(\sigma_i, \sigma_j) = s(\sigma_j, \sigma_i)$ (symmetry)

Property P3. $s(\sigma_i, \sigma_j) = 0 \Leftrightarrow \sigma_i = \sigma_j$ (identity of indiscernibles)

Property P4. $s(\sigma_i, \sigma_k) \leq s(\sigma_i, \sigma_j) + s(\sigma_j, \sigma_k)$ (triangle-inequality).

In general, these are considered desirable properties for a similarity measure, as they ensure its consistency and uniqueness [14, 29].

Furthermore, one can distinguish between *syntactic* and *feature-based similarity measures* [1, 4, 14]. Syntactic similarity measures are calculated directly using the traces without any transformation. In this case h can be considered as an identity function. The pairwise distance d is calculated based on the number of operations required to convert one trace to the other. We will consider edit distance as one particular type of syntactic similarity measure [15].

Definition 2.4 (Edit Distance). *The similarity measure* $s_e(\sigma_i, \sigma_j)$ *based on the edit distance between two distinct traces* σ_i *and* σ_j *is determined by optimizing a given cost function, taking into account the insertions, deletions, and substitutions of activities necessary to convert one trace into the other.*

Two particular variants of edit distance commonly applied in process analysis are the Levenshtein distance and the normalized Levenshtein distance. The *Levenshtein distance* assigns to each editing operation an equal cost of one. The *normalized Levenshtein distance* additionally divides the derived editing costs by the maximum possible number of required edits, i.e., $max(|\sigma_i|, |\sigma_j|)$.

Compared to syntactic similarity measures, feature-based similarity measures first require a transformation of the traces based on some function $h(\sigma_i) = F_i$ before calculating a distance. There exist two types of feature-based similarity measures: set comparison and vector-space embedding.

Definition 2.5 (Set Comparison). *The similarity measure $s_s(\sigma_i, \sigma_j)$ based on set comparison is directly calculated based on the extracted sets of features, i.e., $s_s(\sigma_i, \sigma_j) = d(F_i, F_j)$.*

One similarity measure based on set comparison considered in this work is the Jaccard similarity [34], defined as the complement of the Jaccard coefficient, which is the ratio of the size of the intersection of two sets ($|F_i \cap F_j|$) to the size of their union ($|F_i \cup F_j|$).

In addition to feature extraction, vector-space embedding involves the transformation of a derived set of trace features into a vector.

Definition 2.6 (Vector-space Embedding). *The similarity measure $s_v(\sigma_i, \sigma_j)$ based on vector-space embedding, involves the mapping of the trace feature into a vector space $v : \mathcal{F} \to \mathbb{R}^n$ and the subsequent calculation of the distance between the vectors, i.e., $s_v(\sigma_i, \sigma_j) = (d \circ v)(F_i, F_j)$.*

One particular variant of similarity measure based on vector-space embedding considered in this work is the cosine distance [34]. The cosine distance is calculated as the complement of the dot product of two vectors ($\mathbf{F_i} \cdot \mathbf{F_j}$), divided by the product of their magnitudes ($\|\mathbf{F_i}\|\|\mathbf{F_j}\|$).

In addition to the different distance functions for the calculation of feature-based similarity, there exist also different types of feature-extraction functions h. In process mining, trace features are commonly derived on the basis of n-grams.

Definition 2.7 (n-gram). *n-gram is a feature-extraction function $h_n : \mathcal{A}^* \to \mathcal{P}(F_n)$ that takes a trace σ and returns a set of n-grams of length n: $h_n(\sigma) = \{\langle a_1, a_2, \ldots, a_n\rangle, \langle a_2, a_3, \ldots, a_{n+1}\rangle, \ldots, \langle a_{|\sigma|-n+1}, a_{|\sigma|-n+2}, \ldots, a_{|\sigma|}\rangle\}$ for $1 \leq n \leq |\sigma|$.*

Based on n-grams it is possible to capture structural properties of a trace σ, such as the activities in σ (based on 1-gram), or the directly follows relations in σ (based on 2-gram).

However, identifying high-level structural features (such as loops, skips, choices, and parallelism) requires the application of a discovery algorithm [35].

Process discovery is a method used to construct process models, i.e., graph-based process representations, from event logs, such that high-level structural features are revealed. While process discovery algorithms are commonly applied based on event logs that contain multiple traces, it is also possible to derive graph representations for single traces [17, 19].

To identify parallel executions of activities within traces, we adopt a process discovery algorithm based on instance graphs [17, 19]. This algorithm infers parallelism within a trace based on the ordering of activities within the entire event log, that the trace belongs to. The algorithm first detects causal relations between activities, which then serve as the basis for identifying parallel executions.

Following the definition of causal ordering by Dongen and van der Aalst [17], causal relations can be derived by applying the feature-extraction functions 2-gram (h_2) and 3-gram (h_3).

Definition 2.8 (Causal Relation). *Let L be an event log over a set of Activities A. A causal relation between two activities $a_i, a_j \in A$, denoted as $a_i \rightarrow_L a_j$, is defined in the following way:*
- $a_i >_L a_j$ if and only if there is a trace $\sigma \in L$, such that $\langle a_i, a_j \rangle \in h_2(\sigma)$,
- $a_i \triangle_L a_j$ if and only if there is a trace $\sigma \in L$, such $\langle a_i, a_j, a_k \rangle \in h_3(\sigma)$ where $a_i = a_k$ and $a_i \neq a_j$ and not $a_i >_L a_i$,
- $a_i \rightarrow_L a_j$ if and only if $a_i >_L a_j$ and ($a_j \not>_L a_i$ or $a_i \triangle_L a_j$ or $a_j \triangle_L a_i$), or $a_i = a_j$.

According to this definition, the causal relation between two activities a and b, denoted as $a \rightarrow_L b$, is established if there exists a trace $\sigma \in L$ in which a is directly followed by b, and b is never directly followed by a. However, this definition may be problematic when a and b are involved in a loop of length two. To address this, $a \rightarrow_L b$ also holds true if a trace $\sigma \in L$ contains the sequence $\langle a, b, a \rangle$ or $\langle b, a, b \rangle$, provided that neither a nor b can directly follow themselves. If an activity a does directly follow itself in a trace, then the relation $a \rightarrow_L a$ holds.

The definition of a causal relation is based on the assumption that if one activity consistently precedes another, it is likely that a causal dependency exists between them [36]. This assumption enables the identification of parallelism within traces, even when those traces represent a strict total ordering of activities.

Definition 2.9 (Parallel Relation). *Let $\sigma \in L$ be a trace contained in L. A parallel relation between two activities $a_i, a_j \in \sigma$, denoted as $a_i || a_j$, is defined in the following way:*
- $a_i >_\sigma a_j$ if and only if $\langle a_i, a_j \rangle \in h_2(\sigma)$,
- $a_i || a_j$ if and only if ($a_i >_\sigma a_j$ and $a_i \not\rightarrow_L a_j$) or ($a_j >_\sigma a_i$ and $a_j \not\rightarrow_L a_i$).

Following this definition, it is further possible to identify parallelism between multiple activities.

Definition 2.10 (Set of Parallel Activities). *Let $\sigma \in L$ be a trace contained in L and let $\lambda_i = \langle a_j, ..., a_n \rangle \in \sigma$ denote an n-gram of activities contained in σ. The n-gram is considered to be a set of parallel activities, denoted by P_i, if for all 2-grams $\langle a_k, a_l \rangle \in h_2(\lambda_i)$ it holds that $a_k || a_l$.*

A set of parallel activities, contains a minimum of two activities and a maximum of $|\sigma| = n$ activities. Furthermore, the subscript i represents its relative position in relation to its proceeding and succeeding activities. For example, given a set of parallel relations $R = \{B||C, C||D\}$ and a trace $\sigma_1 = \langle A, B, C, D, E \rangle$ then $P_2 = \{B, C, D\}$ is considered as a set of parallel activities within σ_1. It is also possible that a set of parallel activities occurs multiple times within a trace leading to a loop as depicted in Fig. 2.

To account for additional high-level features that might occur within an individual trace, i.e., loops, skips, or choices, we introduce the following graph-based trace abstraction.

Definition 2.11 (Trace Graph). *Let $A = [a \in \sigma]$ denote the set of all activities in σ, which are not part of a set of parallel activities and let $P = [P_i \in \sigma]$*

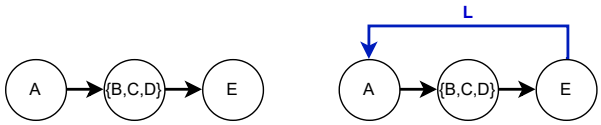

Figure 2: Graph representations of the traces $\sigma_1 = \langle A, B, C, D, E \rangle$ (left side) and $\sigma_2 = \langle A, B, C, D, E, A, B, C, D, E \rangle$ (right side) with $R = \{B||C, C||D\}$. The blue edge (denoted by L) indicates a loop between the activities A and E in σ_2.

denote the set of all sets of parallel activities in σ*. Furthermore, let* V *be a set of vertices and* $D \subseteq V \times V$ *a set of edges. A trace graph is a directed graph* $G_\sigma(V, D)$ *of a trace* σ *with:*
- $V = A \cup P$*, and*
- the edges respect the order of activities and sets of parallel activities, i.e., $D = \{(v_i, v_j) \in V \times V | i < j\}$*.*

By considering the order of activities and the sets of parallel activities within a trace, it is possible to identify specific types of edges within a trace graph.

Definition 2.12 (Edge-type). *Edge-type is a feature-extraction function* $h_e : \mathcal{A}^* \to \mathcal{P}(F_e)$*, where each feature* $f_e = (w, l, d) \in F_e$*, carries information regarding the edge type* $w \in \{sequence, skip, choice, loop\}$ *and the edge length* $l \in \mathbb{N}$ *of an edge* $d \in G$*.*
The four different edge types are defined according to the order of the activities within a trace when iteratively exploring the trace from its start activity a_1 to its end activity a_n [1]. Following the example of $\sigma_1 = \langle A, B, D, A, C, D, A, D \rangle$ depicted in Fig. 1, the edge types can be described as follows:

- The edge (a_i, a_{i+1}) is of the type *sequence*, if the activities that a_{i+1} refers to is explored for the first time when starting from the activity that a_i refers to. For σ_1 this is for example the case for the egde connecting activity A and activity B.

[1] This approach relates to the depth-first search algorithm [37] as will be shown in Sect. 5.

- The edge (a_i, a_{i+1}) is of the type *skip*, if the activity that a_{i+1} refers to has occurred before a_i, and can be reached from the activity that a_i refers to by an alternative (already explored) shortest path that does not include a_i. For σ_1 this is the case for the egde connecting activity A and activity D.

- The edge (a_i, a_{i+1}) is of the type *choice*, if the activity that a_{i+1} refers to has occurred before the activity that a_i refers to and a_i and a_{i+1} share a common preceding activity, respectively connected based on distinct paths. For σ_1 this is the case for the egde connecting activity C and activity D with the common preceding activity A.

- The edge (a_i, a_{i+1}) is of the type *loop*, if the activity that a_{i+1} refers to has occurred before the activity that a_i refers to and is not involved in a relation of type skip nor choice. For σ_1 this is the case for the egde connecting activity D and activity A.

These edge types represent four distinct structural features commonly observed in process traces, i.e., the sequential order of activities (sequence), the recurrence of activities (loop), the skipping of activities (skip), and the selection of an alternative activity (choice). While the lengths of loops, skips, and choices can differ (with a minimum length of one), the length of a sequence is always one.

3. Related Work

A foundation for the measurement of trace similarity can be found in measurement theory [38, 39]. A measurement can be formally defined as a mapping from the domain of the empirical world to a numerical representation. A critical aspect in constructing such a mapping is its *validity*. Three key aspects are commonly considered when assessing validity [39, 40]:

- *Construct validity*: This evaluates whether a measurement accurately represents the theoretical construct it is intended to measure. For example, the execution duration of a trace is generally not suitable for measuring structural similarity.

- *Content validity*: This refers to the extent to which a measurement represents the full range of the empirical phenomenon being studied. For example, to compare the structural similarity between two traces, a similarity measure should reflect a range of structural features.

- *Criterion validity*: This is determined by the relationship between a measure and an external criterion known to be an accurate indicator of the empirical phenomenon. For example, a structural similarity measure should accurately reflect the similarity between traces according to some predefined structural classification of these traces, e.g., indicating complex or non-complex user behavior.

When it comes to the validity of existing similarity measures, one can make the following observations. Regarding construct validity, several different features have been discussed in the literature for the analysis of trace structure, such as n-grams [1, 4, 10, 41, 7, 42], maximal repeat alphabets [28, 7], and eventually follows relationships between activities [2]. In this study, we additionally consider edge-types and parallelism as structural features. These features reflect fundamental structural process properties in the context of user behavior analysis [21, 42] and business process analysis [35, 20, 29, 7, 43].

The content validity of trace similarity measures has been largely neglected in research, as these measures have primarily been considered in conjunction with downstream tasks such as process discovery, trace clustering, or anomaly detection [14]. However, it has been recognized that similarity measures originating from other research disciplines such as natural language processing, control theory, and bioinformatics fail to recognize process specific structural properties [14, 44]. To address this issue more systematically, we provide a formal

comparison of trace similarity measures in the following section, allowing us to determine which specific structural features are reflected by each measure.

Criterion validity of trace similarity measures has been initially addressed by Back and Simon [14]. We build on their evaluation approach, by additionally introducing synthetic event logs, that allows a clear classification of traces according to their structural properties. Additionally, we consider not only business processes, but also user behavior processes, where the ground truth is determined by the complexity of user behavior.

In general, one can distinguish between the measurement of process similarity at an instance level [1, 4, 14, 41, 44, 45, 46] and at a model level [41, 29, 43, 47]. At the instance level, a similarity measure is directly calculated between two traces. This approach is commonly applied for the analysis of business processes [1, 14, 44] as well as behavioral processes [21, 42]. In business process analysis, these measures are utilized, for example, in trace clustering or anomaly detection. In user behavior analysis, these measures are applied, for example, in the comparison of visual scanpaths, which are detected based on eye-tracking. A scanpath can be considered as a process trace, which contains the order of a user's visual fixation points on a screen. By comparing the scanpaths, one can identify the similarity in user behavior, e.g., when the users are solving tasks of different complexity [23, 24].

At the model level, a similarity measure is calculated between two process models, i.e., process graphs. This approach is commonly used for the analysis of business processes, e.g., to measure the compliance between a reference and an actual model [6], or to search for models in a repository [29]. The comparison at the model level has the advantage that high-level structural features of the process are explicit in the graph, which is not the case at the trace level. One way to overcome this issue is to apply process discovery [35] to derive process models from traces, which subsequently allow to extract additional structural features for the trace comparison.

Thus, in this study we seek to combine both research approaches, based on trace comparison (Sect. 3.1) and based on graph comparison (Sect. 3.2) by

leveraging process discovery (Sect. 3.3).

3.1. Trace Comparison

The measures used for process comparison at trace level can be classified as syntactic and feature-based similarity measures [1, 45, 46, 14] [2].

3.1.1. Syntactic Similarity Measures

A commonly applied syntactic similarity measure in process analysis is ***edit distance***, which includes the two variants: ***Levenshtein distance*** and ***normalized Levenshtein distance*** [6, 48, 7, 44, 45, 46]. An additional variant of an edit distance is proposed by Bose and van der Aalst [44], where editing costs are adjusted according to the context in which they occur. The cost values are calculated on the basis of the relative frequency of activity pairs within the traces. The approach thus does not explicitly consider structural features.

An alternative syntactic similarity measure is ***sequence alignment*** [10, 14, 20, 28, 45, 46]. Similarly to the edit distance, sequence alignment is calculated by optimizing a given scoring function, taking into account operations necessary to align one trace with the other. In general, edit distance and sequence alignment yield equivalent outcomes (cf. Sellers [49]).

3.1.2. Feature-based Similarity Measures

A common approach for the calculation of feature-based similarity is the transformation of traces into sets of ***n-grams*** [1, 48, 10, 20, 44, 50, 51]. However, the length of n differs between the different studies. An issue is that long n-grams ($n > 2$) can capture more complex structural properties, e.g. loops of length > 2, but at the same time fail to capture structural properties in shorter n-grams ($n \leq 2$). One solution is to aggregate similarity measures, considering n-grams of different length. While Delias et al. [2] propose to combine 1 and 2-grams,

[2] Back and Simonsen [14] use the distinction between *syntactic similarity measures* and *vector-space embedding*. Our distinction is more generic, since we also include feature-based similarity measures that do not depend on vector-space embedding, such as Jaccard similarity.

Back et al. [12] propose to combine all possible substrings of a trace, from length one to length $n = min(|\sigma_i|, |\sigma_j|)$. However, this approach is computationally expensive and also does not differentiate between higher-level structural features such as loops, skips, and choices.

An alternative to trace comparison based on n-grams is the comparison based on ***sequential patterns*** [7], i.e., ordered sequences of activities that occur repeatedly within a trace. Sequential pattern mining algorithms are commonly applied in a variety of domains, such as bioinformatics, e-learning, market basket analysis, and click-stream analysis of webpages [22]. Although there exist a vast number of sequential pattern mining algorithms in the literature (for an overview, see Fournier-Viger et al. [22]), they are not commonly applied in the context of process analysis. The comparative study by Back and Simonsen [14] showed that similarity measures based on sequential patterns do not perform as well as other measures. One reason for this could be that the algorithms detect very detailed sequential patterns that are specific to single traces, and thereby fail to capture structural features that are common among multiple traces. We therefore restrict our analysis to similarity measures using 1-grams and 2-grams.

Beyond the structural trace characteristics, there are additional features that can be extracted from a trace and used for comparison. One such feature is the relative frequency of n-grams within a trace, which can be captured by ***eventually-follows relations*** between the activities. Delias et al. [2] propose an aggregated similarity measure based on cosine similarity, considering 1-grams and eventually-follows relations. According to the study by Back et al. [12] this measure yields a particularly high performance for a number of different event logs. The measurement approach is therefore also included in our analysis.

Additionally, traces can be compared not only in terms of structural features, but also based on features representing different process perspectives. For business processes, this often includes performance and resource attributes corresponding to events in a trace [48, 10, 20, 52]. For behavioral analysis based on scanpaths this often includes the position and duration of a visual fixation on a screen [45, 46]. However, for this study, we will only consider trace simi-

larity measures on the basis of structural properties, also commonly defined as control-flow perspective.

Finally, for the calculation of a similarity value based on features, existing studies either use ***vector space embedding*** [1, 4, 44] or ***set comparison*** [1, 10, 11, 50]. Vector-space embedding involves the comparison of trace features represented in a vector space, whereas set comparison involves the direct comparison between sets of features (cf. Sect. 2). To the best of our knowledge, there exists no study with a direct comparison of the two approaches.

There exist different methods for representing a trace in vector space. A commonly applied method is to construct a trace vector by evaluating the frequency of specific trace features [1, 4, 44]. Another alternative method involves employing neural networks [20, 41, 51]. Neural networks thereby learn a vector space representation for a trace based on a set of training data.

However, neural network-based approaches exhibit three key limitations in the context of structural trace similarity measurement. First, they require labeled training data, which may not always be readily available. Second, the resulting vector representations are typically opaque, making it difficult to interpret the learned features and their associated weights [41]. Third, the similarity outcomes are sensitive to the choice of neural network architecture (see [51] for an overview), as well as to several parameters, including the selection of input features and the dimensionality of the output vectors.

To the best of our knowledge, these limitations have not yet been systematically investigated in the context of trace similarity measurement. Nevertheless, for the purpose of empirical evaluation (cf. Sect. 6.4), we include a neural network-based representation using the Trace2Vec architecture and parameter configuration as proposed by De Koninck et al. [41].

3.1.3. Aggregation of Trace Similarity Values

Building on feature-based similarity measures, a common approach to improve the outcome of the structural comparison between traces, is the aggregation of multiple distinct similarity measures, which reflect different structural

	J1	J2	J1 + J2
$s(\sigma_1 = \langle A, B, C, D\rangle, \sigma_2 = \langle A, B, C, D\rangle)$	0	0	0
$s(\sigma_1 = \langle A, B, C, D\rangle, \sigma_2 = \langle C, D, A, B\rangle)$	0	$\frac{1}{3}$	$\frac{1}{3}$
$s(\sigma_1 = \langle A, B, C, D\rangle, \sigma_2 = \langle W, X, Y, Z\rangle)$	1	1	2

Table 1: Example for the aggregation of the two saperate similarity values J1 (Jaccard similarity based on 1-gram) and J2 (Jaccard similarity based on 2-gram).

properties of the traces [2, 11, 50]. This approach allows to combine different types of feature, such as n-grams of different length, in order to consider structural trace similarities and differences in a more comprehensive way. A simple example, for the aggregation of two similarity values based on 1-grams (the set of activities) and 2-grams (the set of directly-follows relations) as proposed by Burattin et al. [11] is shown in Table 1. Other proposed aggregations involve the combination of 1-grams and eventually-follows relations [2], and the combination of different order relations between activities, i.e., directly-follows relations, exclusiveness, and parallelism [50]. The separate values are thereby either calculated based on Jaccard similarity [11, 50] or cosine distance [2].

The proposed aggregations, always involve the aggregation according to the weighted sum of the considered similarity measures. This allows to weight feature types, according to the context in which the measurement is applied [2, 11, 50]. These weights can either by assigned randomly [50], according to some heuristic [2], or by solving an optimization problem, e.g., to find an optimal clustering of traces [11].

In Sect. 5, we extend existing aggregation approaches by combining trace similarity measures that account for activities, directly-follows relations, and high-level structural features. While context-specific weighting of these components could potentially improve the accuracy of the aggregated measures, its definition lies beyond the scope of this work. Accordingly, all distinct measures are weighted equally.

3.2. Model-based Similarity Measures

The measures used for process comparison at model level commonly leverage structural features, which are explicit in the models (for an overview, see [29, 43, 47]). These measures focus on the comparison of process model elements, such as node types and edges. Their outcome depends on the labeling of the nodes and edges, as well as the applied modeling language. Other measures focus only on the comparison of the graph structure, without considering specific process model elements (for an overview on graph similarity measures, see Emmert-Streib et al. [53]). However, these measures do not reflect the higher-level structural features (loops, skips, and choices), which are an important aspect of process analysis.

Finally, some measures are based on the comparison of sets of traces which can be derived from the execution of process models, such as trace alignments [6]. These measures are similar to those once proposed for trace comparison (Sect. 3.1), except that they focus on the comparison of sets that contain multiple traces. These measures also fail to capture higher-level structural features of the traces and therefore provide little explanation for the derived similarity values.

Overall, it can be stated that the similarity measures developed for process comparison at model level have several shortcomings when applied for the comparison at trace level, i.e., the dependency on specific modeling languages, the comparison between sets of traces rather than single traces, and the failure to consider high-level structural features. To overcome these shortcomings, we propose a similarity comparison based on trace graphs (cf. Sect. 2). Deriving a respective trace graph for a trace requires some form of abstraction, which is commonly achieved based on process discovery.

3.3. Process Discovery

Process discovery can be seen as a way to derive structural features from a set of traces. There exist several process discovery algorithms that are capable of deriving sequential patterns [19, 17, 54] and non-sequential patterns [8],

which represent reoccurring structural properties within an event log. However, these algorithms are based on the assumption that the process structure can be abstracted from multiple distinct traces, that arise from multiple executions of the same process. This generally involves a high level of abstraction in order to generate comprehensible graphical process representations. Some of these algorithms thereby focus only on the detection of frequently reoccurring patterns and neglect non-frequent once [54, 8]. Thus, the detection of all process patterns contained in a single trace is not guaranteed. Furthermore, some of the discovery algorithms focus on the discovery of only a single type of pattern, such as the parallel execution of activities, as in the case of instance graphs [17, 19], and frequent episodes [54].

In particularly, parallelism requires a high level of abstraction, since the number of possible orderings of activities increases exponentially with the number of activities that can be executed in parallel. This makes it difficult to detect parallelism based on a single trace.

Furthermore, due to the abstraction, more fine-granular structural properties are ignored. E.g., based on $\sigma_1 = \langle A, B, C, D, A, C, B, D \rangle$, a process discovery algorithm, such as the local process model miner [8], would assume that activities B and C can be executed in parallel, suggesting that the structure of σ_1 equals the structure of $\sigma_2 = \langle A, C, B, D, A, B, C, D \rangle$. However, this can be particularly misleading for the comparison of traces describing user behavior [55, 56]. For example, the particular order of clicks can be an indicator for different approaches to process modeling [55], and the order of visual fixations on a screen can be an indicator for exploratory or goal oriented behavior [56]. Different behaviors can thereby be associated with different cognitive processes of a user [24, 25, 57], which require different interpretations and potentially different user support.

Nevertheless, parallelism has been consistently considered as a relevant structural feature in the context of business processes [17, 36, 19, 17, 54, 8, 35, 58]. We therefore consider parallelism as an additional high-level feature for the comparison of the structural similarity between traces in the context of business processes. To discover parallelism within a single trace, we leverage a

discovery algorithm initially proposed by van Dongen and van der Aalst [17], which explicitly focuses on the structural analysis of single traces. Furthermore, the algorithm has been successfully applied in the context of business process analysis [19]. An additional advantage of the algorithm is its assumption of a strict total order of activities within a trace, which eliminates the need for partial ordering, i.e., the documentation of overlapping time intervals, to identify parallelism [58].

To better distinguish which similarity aspects are considered by different similarity measures, especially with respect to structural trace features, we provide a formal comparison on the basis of several measurement properties in the following section.

4. Formal Comparison of Trace Similarity Measures

For the formal comparison of trace similarity measures, we propose ten desirable properties. These include the already introduced metric properties in Sect. 2 (P1-P4), properties concerning the structural trace features (P5-P7), as well as general similarity properties (P8-P10).

4.1. Properties Related to Structure and Similarity

Existing studies on trace similarity measures [14, 41, 28, 44] show that trace similarity measures perform differently depending on the characteristics of an event log, such as the number of activities, length of traces, and number of trace classes. While this event log perspective provides some insights into the applicability of similarity measures, it does not consider how these measures reflect specific trace characteristics, such as structural features. To solve this issue, we identify three desirable properties a similarity measure should have, in order to reflect the traces' structure. These properties build on the assumption that a similarity measure should be monotonically increasing with respect to the number of structural differences between traces. We consider three types of structural differences based on (1) activities, (2) directly-follows relations, and

(3) high-level structural features (loops, skips, and choices). These three types of structural features provide a comprehensive description of the trace structure and are commonly deemed relevant in the literature [35, 20, 29, 43]. It is worth noting that parallelism, which is also commonly deemed a relevant high-level structural feature in the literature [17, 36, 19, 17, 54, 8, 35, 58] is implicitly captured through differences based on directly-follows relations.

The first structural property refers to the differences among the activities. A similarity measure should be strictly increasing with respect to differences in the number of activities. This is based on the intuitive notion that the more distinct the respective sets of activities, the greater the dissimilarity between the traces. For comparison of the dissimilarity between two sets, we use their symmetric difference $F_i \triangle F_j = (F_i \cup F_j \setminus F_i \cap F_j)$. The set of activities contained within a trace can be derived according to the feature-extraction function based on 1-gram, i.e., $h_1(\sigma_i) = F_{i,1}$.

Property P5. *The similarity measure is strictly increasing with an increase in the number of dissimilar activities between two traces, i.e.:* $s(F_{i,1}, F_{j,1}) < s(F_{i,1}, F_{k,1})$ *for all* $|F_{i,1} \triangle F_{j,1}| < |F_{i,1} \triangle F_{k,1}|$.

Looking at the traces σ_0, σ_3 and σ_4 depicted in Fig. 3, according to Property P5 we would expect $s(\sigma_0, \sigma_3) < s(\sigma_0, \sigma_4)$ since the sets of activities contained in σ_0 and σ_4 are more distinct than it is the case for σ_0 and σ_3.

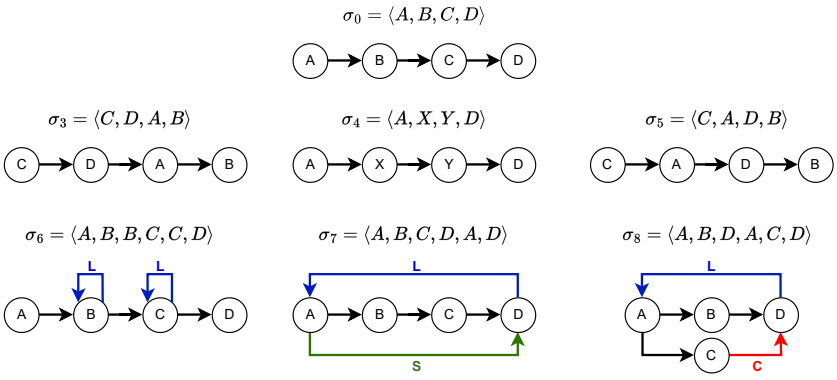

Figure 3: Different traces and their representation as trace graph.

The second structural property refers to the differences among the directly-follows relations. A similarity measure should be strictly increasing with respect to the differences in the number of directly-follows relations. This is based on the intuitive notion that the more distinct the ordering of activities, with respect to their directly-follows relations, the greater the dissimilarity between the traces. The set of directly-follows relations contained within a trace can be derived according to the feature-extraction function based on 2-gram, i.e., $h_2(\sigma_i) = F_{i,2}$.

Property P6. *The similarity measure is strictly increasing with an increase in the number of dissimilar directly-follows relations between two traces, i.e.:* $s(F_{i,2}, F_{j,2}) < s(F_{i,2}, F_{k,2})$ *for all* $|F_{i,2} \triangle F_{j,2}| < |F_{i,2} \triangle F_{k,2}|$.

Looking at the traces σ_0, σ_3 and σ_5 depicted in Fig. 3, according to Property P6 we would expect $s(\sigma_0, \sigma_3) < s(\sigma_0, \sigma_5)$, since the sets of directly-follows relations contained in σ_0 and σ_5 are more distinct than it is the case for σ_0 and σ_3.

In addition to the set of activities and their relative order, a similarity measure should also consider high-level structural features of the traces on the basis of edge-types (cf., Def. 2.12). The set of edge-types contained within a trace can be derived according to the feature-extraction function $h_{et}(\sigma_i) = F_{i,et}$.

Property P7. *The similarity measure is strictly increasing with an increase in the number of dissimilar edge-types between two traces, i.e.:* $s(F_{i,et}, F_{j,et}) < s(F_{i,et}, F_{k,et})$ *for all* $|F_{i,et} \triangle F_{j,et}| < |F_{i,et} \triangle F_{k,et}|$.

Looking at the traces depicted in Fig. 3, according to Property P7 we would expect $s(\sigma_0, \sigma_3) < s(\sigma_0, \sigma_6)$ due to the loop in σ_6. Similarly we would expect $s(\sigma_0, \sigma_3) < s(\sigma_0, \sigma_7)$ due to the loop and skip in σ_7, and $s(\sigma_0, \sigma_3) < s(\sigma_0, \sigma_8)$ due to the loop and choice in σ_8.

In addition to the structure-related properties P5-P7, we define three additional properties related to general similarity considerations. A trace similarity measure should also consider the overall size of the features contained in the traces. For example, the similarity between $\sigma_0 = \langle A, B, C, D \rangle$ and $\sigma_9 = \langle A, B, C, D, X \rangle$ should be considered to be greater than between $\sigma_9 = \langle A \rangle$

and $\sigma_{10} = \langle A, X \rangle$, although in both cases the traces differ only based on a single activity. Thus, a new feature should contribute less to the dissimilarity between two traces, the larger the overall set of features contained in both traces. This property is also known as *submodularity*.

Property P8. *For all* $(F_i \cup F_j) \subseteq (F_k \cup F_l)$ *and* $f \notin F_k, F_l$ *we have that* $s(F_i, F_j \cup f) - s(F_i, F_j) \geq s(F_k, F_l \cup f) - s(F_k, F_l)$.

However, this property only considers the overall set of features contained in the traces, but not the ratio between similar and dissimilar features. From a process analysis perspective, Becker and Laue [29] argue that a similarity measure should take into account commonalities, as well as differences between the traces.

Property P9. *A similarity measure should consider both commonalities* $F_i \cap F_j$ *and differences* $F_i \triangle F_j$ *between two traces.*

Finally, we also consider the computational costs of the similarity measures to ensure that they can be calculated for large event logs and in online settings (cf. Sect. 1), which was also proposed by Becker and Laue [29] as a relevant property for the comparison of similarity measures.

Property P10. *The similarity measure can be calculated efficiently.*

Table 2 provides an overview of all desirable properties for trace similarity measures, including the metric properties defined in Sect. 2.

Metric	**P1**	non-negativity
	P2	Symmetry
	P3	Identity of indiscernibles
	P4	Triangle-inequality
Structure	**P5**	Strict monotonicity with an increase in the number of dissimilar activities
	P6	Strict monotonicity with an increase in the number of dissimilar directly-follows relations
	P7	Strict monotonicity with an increase in the number of dissimilar edge types
Similarity	**P8**	Submodularity
	P9	Consideration of commonalities and differences
	P10	Computational efficiency

Table 2: Overview of the desirable properties for similarity measures.

4.2. Formal Comparison

Based on the defined properties in Table 2 we will now provide a comparison of the trace similarity measures shown in Table 3. One main criterion for selecting these measures is their ability to represent structural features (cf. Sect. 3). Furthermore, we consider similarity measures, which are commonly applied in the literature (cf. Sect. 3) and can be computed with reasonable computational effort in the context of both behavioral processes and business processes.

Type	Measure	Label
Syntactic	Levenshtein distance	LD
	Normalized Levenshtein distance	N-LD
Feature-based Vector-space embedding	Euclidean similarity based on MR	EMR
	Cosine similarity based on MR	CMR
	Jaccard similarity based on MR	JMR
	Eventually-Follows	EF
	Euclidean similarity based on 1-gram	E1
	Euclidean similarity based on 2-gram	E2
	Euclidean similarity based on 3-gram	E3
	Cosine similarity based on 1-gram	C1
	Cosine similarity based on 2-gram	C2
	Cosine similarity based on 3-gram	C3
Feature-based Set comparison	Jaccard similarity based on 1-gram	J1
	Jaccard similarity based on 2-gram	J2
	Jaccard similarity based on 3-gram	J3

Table 3: Selection of similarity measures.

To investigate whether the selected similarity measures reflect the structural properties defined by P5-P7, we calculate the respective similarity values between σ_0 and $\sigma_3 - \sigma_8$ that are shown in Fig. 3. The traces differ according to the activities, the order of activities, and high-level structural features. For a measure to accurately distinguish between traces based on the number of activities (P5), it should satisfy the condition that $s(\sigma_0, \sigma_3) < s(\sigma_0, \sigma_4)$, since the sets of activities that can be extracted from σ_0 and σ_3 are more similar compared to those derived from σ_0 and σ_4. Similarly, for a measure to accurately

	$s(\sigma_0, \sigma_3)$	$s(\sigma_0, \sigma_4)$	$s(\sigma_0, \sigma_5)$	$s(\sigma_0, \sigma_6)$	$s(\sigma_0, \sigma_7)$	$s(\sigma_0, \sigma_8)$
LD	4.000	2.000	4.000	2.000	2.000	2.000
N-LD	1.000	0.500	1.000	0.333	0.333	0.333
EMR	0.000	0.000	0.000	1.414	1.414	1.414
CMR	0.000	0.000	0.000	0.000	0.000	0.000
JMR	0.000	0.000	0.000	0.000	0.000	0.000
EF	0.179	0.150	0.299	0.169	0.072	0.113
E1	0.000	2.000	0.000	1.414	1.414	1.414
E2	2.449	2.449	3.162	1.414	1.414	2.000
E3	2.828	2.828	2.828	2.449	2.000	2.449
C1	0.000	0.500	0.000	0.051	0.051	0.051
C2	0.600	0.600	1.000	0.155	0.155	0.324
C3	1.000	1.000	1.000	0.592	0.388	0.592
J1	0.000	0.667	0.000	0.000	0.000	0.000
J2	0.750	0.750	1.000	0.286	0.286	0.500
J3	1.000	1.000	1.000	0.750	0.571	0.75

Table 4: Calculated similarity values for the pairwise comparison between σ_0 and $\sigma_3 - \sigma_8$, where $s(\sigma_0, \sigma_3)$ is considered as baseline. According to the defined properties P5-P7, $s(\sigma_0, \sigma_4)$ should reflect an increase in the number of activities (P5), $s(\sigma_0, \sigma_5)$ should reflect an increase in the number directly-follows relations (P6), and $s(\sigma_0, \sigma_{6-8})$ should reflect an increase in the number of edge-types (P7). Values that are less than or equal to $s(\sigma_0, \sigma_3)$ indicate a violation of one of the defined properties P5-P7 and are highlighted in gray.

distinguish between traces based on the number directly-follows relations (P6) it should hold that $s(\sigma_0, \sigma_3) < s(\sigma_0, \sigma_5)$. Finally, for a measure to accurately distinguish between traces based on the of number of edge-types it should hold that $s(\sigma_0, \sigma_3) < s(\sigma_0, \sigma_{6-8})$. Table 4 shows all calculated similarity values. All similarity values that are less than or equal to $s(\sigma_0, \sigma_3)$ indicate a violation of one of the defined properties P5-P7 and are highlighted in gray. The results of the calculated similarity values are summarized in Table 5.

Table 5 provides an overview of the considered similarity measures with an indication of whether they adhere to the defined properties P1-P10. Regarding

	Metric				Structure			Similarity		
	P1	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**
LD	yes	yes	yes	yes	no	no	no	no	no	$\mathcal{O}(\lvert\sigma_1\rvert \cdot \lvert\sigma_2\rvert)$
N-LD	yes	yes	yes	no	no	no	no	no	no	$\mathcal{O}(\lvert\sigma_1\rvert \cdot \lvert\sigma_2\rvert)$
EMR	yes	yes	yes	yes	no	no	yes	no	yes	$\mathcal{O}(\lvert F_{i,MR} \cup F_{j,MR}\rvert)$
CMR	yes	yes	yes	no	no	no	no	no	yes	$\mathcal{O}(\lvert F_{i,MR} \cup F_{j,MR}\rvert)$
JMR	yes	yes	yes	yes	no	no	no	yes	yes	$\mathcal{O}(\lvert F_{i,MR} \cup F_{j,MR}\rvert)$
EF	yes	yes	yes	no	no	yes	no	no	yes	$\mathcal{O}(\lvert F_{i,2} \cup F_{j,2}\rvert)$
E1	yes	yes	yes	yes	yes	no	yes	no	yes	$\mathcal{O}(\lvert F_{i,1} \cup F_{j,1}\rvert)$
E2	yes	yes	yes	yes	no	yes	no	no	yes	$\mathcal{O}(\lvert F_{i,2} \cup F_{j,2}\rvert)$
E3	yes	yes	yes	yes	no	no	no	no	yes	$\mathcal{O}(\lvert F_{i,3} \cup F_{j,3}\rvert)$
C1	yes	yes	yes	no	yes	no	yes	no	yes	$\mathcal{O}(\lvert F_{i,1} \cup F_{j,1}\rvert)$
C2	yes	yes	yes	no	no	yes	no	no	yes	$\mathcal{O}(\lvert F_{i,2} \cup F_{j,2}\rvert)$
C3	yes	yes	yes	no	no	no	no	no	yes	$\mathcal{O}(\lvert F_{i,3} \cup F_{j,3}\rvert)$
J1	yes	yes	yes	yes	yes	no	no	yes	yes	$\mathcal{O}(\lvert F_{i,1} \cup F_{j,1}\rvert)$
J2	yes	yes	yes	yes	no	yes	no	yes	yes	$\mathcal{O}(\lvert F_{i,2} \cup F_{j,2}\rvert)$
J3	yes	yes	yes	yes	no	no	no	yes	yes	$\mathcal{O}(\lvert F_{i,3} \cup F_{j,3}\rvert)$

Table 5: Comparison between similarity measures based on the identified desirable properties $P1 - P10$.

the metric properties (P1-P4) it can be stated that the Levenshtein distance (LD), as well as the measures based on Jaccard similarity (JMR, J1,J2, and J3) and based on Euclidean similarity (EMR, E1, E2, and E3) adhere to the four metric properties (cf. Yujian and Bo [59], Kosub [60]). The normalized Levenshtein distance (N-LD) and the measures based on cosine similarity (CMR, EF, C1, C2, and C3) cannot be considered a metric since they violate triangle-inequality (cf. Yujian and Bo [59], Han et al. [34]).

Regarding the structure-related properties (P5-P7), it can be stated that none of the measures adheres to all defined properties. Only the measures based on 1-gram (E1, C1, and J1) correctly evaluate the dissimilarity between the number of activities (P5). Furthermore, only the measures involving 2-gram (EF, E2, C2, and J2) correctly evaluate the dissimilarity between the directly-follows relations (P6). Additionally, the measures that also reflect the relative frequencies of the activities within the traces (EMR, E1, C1) correctly evaluate

the dissimilarity between the edge-types (P6).

Regarding the general similarity properties (P8-P10) it can be stated that only measures based on Jaccard similarity (JMR, J1, J2, and J3) satisfy submodularity (P8), which is characterized by the diminishing returns condition: for any sets $A \subseteq B \subseteq V$ and element $x \in V \setminus B$, the marginal gain from adding x to A is at least as large as adding it to B (for a formal proof see Kosub [60]). In contrast, the remaining measures do not exhibit submodularity. Specifically, the measures based on Levenshtein distance (LD and N-LD) fail to account for the cardinality of the feature set, which is essential for satisfying the diminishing returns property. Furthermore, measures based on cosine similarity and Euclidean similarity are inherently geometric and depend on the relative positions of feature vectors in a continuous space [61]. These measures do not define set functions over discrete subsets and do not satisfy the submodularity condition, as the returns are not guaranteed to decrease with increasing set size.

It can be further stated that all similarity measures, except those based on Levenshtein distance (LD and N-LD), account for both commonalities and differences between traces, as required by property (P9) [29]. In particular, the Jaccard similarity can be reformulated as $J(A, B) = 1 - \frac{|A \cap B|}{|A \cup B|} = \frac{|A \triangle B|}{|A \cup B|}$, where the numerator $|A \triangle B|$ captures the symmetric difference, and the denominator $|A \cup B|$ reflects the total number of distinct elements, encompassing both shared and non-shared features [60]. Similarly, cosine similarity and Euclidean distance are computed based on the relative positions of feature vectors in a continuous space, where both shared and non-shared features influence the resulting values [61]. These measures inherently incorporate both overlap and divergence in feature representation. In contrast, LD and N-LD focus solely on the edit operations required to transform one trace into another, without explicitly considering the set of shared features [15], and therefore do not satisfy (P9).

Finally, it can be stated that all similarity measures can be calculated in linear time (P10), depending on the length of the traces or the respective sets of features.

In conclusion, only Jaccard similarity measures (J1 and J2) can be considered as a proper metric (P1-P4), satisfying submodularity (P8) and recognizing dissimilarities and similarities between traces (P9). Furthermore, none of the similarity measures considered correctly evaluates all structural dissimilarities between the traces (P5-P7). In the following section, we propose a novel approach to solve this problem by incorporating different structural features into the similarity evaluation.

5. Similarity Measures Based on Graph-based Features

To address the limitations of existing trace similarity measures identified in the previous chapter, we propose the integration of high-level structural features into the comparison of trace similarity. Specifically, we introduce a novel feature extraction approach based on graph algorithms, which enables the identification of loops, skips, and choices within individual traces. In addition, we leverage instance graphs [17, 19] to detect parallelism by analyzing directly-follows relations across multiple traces. Building on the aggregation approach identified in the literature (cf. Sect. 3.1.3), we further present four aggregated similarity measures that incorporate at least two of the structural dimensions considered in the previous chapter: activities, directly-follows relations, and high-level structural features. A formal analysis of the introduced measures in this section, shows that the high-level structural features and the aggregation can indeed lead to a more complete comparison of the traces' structure.

5.1. Identifying Parallelism within Traces

As discussed in Sect. 3.1.2, parallelism between activities is commonly regarded as a key structural feature in traces representing business processes. In contrast, user behavior processes typically exhibit strictly sequential execution and generally lack parallel activity patterns (cf. Sect. 1). Therefore, parallelism is treated as an optional high-level feature and is only considered for traces that represent business processes.

According to Definition 2.9, parallel relations between activities are derived by identifying all directly-follows relations (i.e., 2-grams) within a trace $\sigma \in L$ and all causal relations within the event log L. Once parallel relations have been established, sets of parallel activities (cf. Definition 2.10) can be derived, which can then be used for the construction of trace graphs (cf. Definition 2.11).

5.2. Incorporating Graph-based Features into Similarity Measures

Identifying loops, skips, and choices within a trace requires the transformation of a trace into a trace graph $G(A, D)$ (cf. Sect. 2). This can be achieved by deriving an adjacency list from a trace. In an adjacency list, each vertex $a \in A$ is linked to a list providing information of all neighboring vertices connected by an edge $d \in D$ [37]. The transformation of a trace into an adjacency list leads to some abstraction, e.g., even though loops might be repeated more than once in a trace, based on the transformation, they are always considered to have the same frequency. This abstraction allows for a direct comparison of the trace structure in a computationally efficient manner, independent of process performance aspects, such as the number of loop repetitions [62].

Based on the adjacency list, one can subsequently derive high-level structural features, which are implicitly contained in the trace. This can be achieved by identifying the graph's edge-types (cf. Sect. 2). Two examples of trace graphs with different edge-types are shown in Fig. 1. In order to identify the edge-types and their respective lengths within a trace graph, we apply the following graph algorithms: depth-first search (DFS), breadth-first search (BFS), and lowest-common ancestor (LCA). Overall, the measurement approach consists of three steps.

5.2.1. Step 1: Identifying Edge-types

The ***DFS*** [37] is applied to detect the edge-type between the vertices of the graph. The DFS starts at the first edge between two vertices according to the adjacency list and then continues to look at all the other edges in the list. The algorithm thus considers whether the edge was visited before and which other

edges have been already visited. The edge-types are assigned accordingly as depicted in Algorithm 1.

Algorithm 1 DFS to detect graph-features including length

Input: $G(V, D)$ in the form of adjacency list $AdjList$

Output: List of tuples $(W, \mathbb{N}, e)$ with edge-type and length for all $e \in G(V, D)$

```
1: main() function
2: i ← 0                                                    ▷ initialize counter
3: numList(|v|) ← 0                          ▷ initialize list with counters of vertex visits
4: recList(|v|) ← 0          ▷ initialize list with vertices currently on recursion stack
5: for all v ∈ AdjList do
6:     if numList(v) = 0 then dfs(v)                        ▷ apply recursive DFS function
7:
8: dfs() function
9: i ← i + 1, numList(v) ← i, recList(v) ← 1
10: for all w ∈ adj(v) do                          ▷ adj(v) is the list for v ∈ AdjList
11:     if numList(w) = 0 then dfs(w)                          ▷ ⟨v, w⟩ is a sequence edge
12:     else if numList(w) > numList(v) then                   ▷ ⟨v, w⟩ is a skip edge
13:         call BFS(start = v, end = w)                ▷ derive length based on Algo. 2
14:         return (type = forward, length, ⟨v, w⟩)
15:     else if numList(w) < numList(v) then                   ▷ ⟨v, w⟩ is a choice edge
16:         call LCA(w,v)                               ▷ derive length based on Algo. 3
17:         return (type = cross, length, ⟨v, w⟩)
18:     else                                                   ▷ ⟨v, w⟩ is a loop edge
19:         call BFS(start = w, end = v)                ▷ derive length based on Algo. 2
20:         return (type = back, length, ⟨v, w⟩)
21:     recList(v) ← 0
```

5.2.2. Step 2: Identifying the Length of Edge-types

While sequences only indicate a directly-follows relation between two consecutive activities, loops, skips, and choices can span over multiple activities. For example, the trace $\sigma_1 = \langle A, B, D, A, C, D, A, D \rangle$ depicted in Fig. 1 contains a loop of length three, a skip of length one, and a choice of length one. These

Algorithm 2 BFS

Input: $G(V, D)$ in the form of adjacency list $AdjList$, start vertex a, end vertex b

Output: Minimum number of steps from a to b, list of all visited vertices

```
1: bfs() function
2: queueList ← (a, steps = 0)                          ▷ initialize queue
3: visitedList ← 0                                     ▷ initialize list of visited vertices
4: while queueList not empty do
5:     currentVertex, steps ← last entry queueList
6:     if currentVertex not in visitedList then
7:     visitedList ← currentVertex
8:     if currentVertex = b then
9:         steps + sum(parallel activities in visitedList)   ▷ adjust length
10:        return steps, visitedList
11:    for all neighboringVertex of currentVertex ∈ AdjList do
12:        queueList ← (neighboringVertex, steps + 1)
13:        if neighboringVertex not in visitedList then
14:            visitedList ← (neighboringVertex, steps)
```

Algorithm 3 LCA

Input: $G(V, D)$ in the form of adjacency list $AdjList$, vertex a, vertex b

Output: Lowest-common ancestor of a and b, and the min number of steps

```
1: remove edge ⟨a, b⟩ from AdjList
2: AdjList' ← invert(AdjList)
3: LCA() function
4: v ← startVertex(AdjList')
5: l1 = BFS(v, a)    ▷ derive list of visited vertices with min. steps based on Algo.2
6: l2 = BFS(v, b)    ▷ derive list of visited vertices with min. steps based on Algo.2
7: x ← firstCommonVertex(l1,l2)
8: return min number of steps between a and x, b and x
```

lengths are derived based on the first occurrence of the respective edge-types within the trace, i.e., the loop-edge between D and A first occurs after the

execution of $\langle A, B, D \rangle$, thus it is assigned a length of three.

Since the length of a structural construct can be an important aspect for the comparison of traces, we combine the DFS with two other graph-algorithms to derive this additional information. First, we use the ***BFS*** algorithm [37] to identify the minimum number of steps required to get from one vertex $a \in A$ to another vertex $b \in A$ following the edges in a graph. Starting from a selected vertex $a \in A$, BFS explores the neighbor vertices at the present depth before moving on to the vertices at the next depth level. This process continues until the target vertex $b \in A$ is found. The final depth indicates the minimum number of steps to get from a to b. In our implementation, we use a queue data structure to keep track of the vertices that remain to be explored (cf. Algorithm 2).

While the BFS can identify the length of loops and skips, for choices, it is necessary to first identify the start vertex of an alternative path. This can be solved based on the ***LCA*** algorithm [37]. As described by Algorithm 3, before applying the LCA the trace graph needs to be inverted such that all start vertices become the end vertices of the graph. The LCA algorithm is subsequently applied to find the lowest-common ancestor of two given vertices $a, b \in A$. The lowest-common ancestor is defined as the lowest vertex that has a and b as descendants.

5.2.3. Step 3: Aggregating the Features

To leverage the identified high-level features in trace comparison, we propose six measures, shown in Table 7. The measures are based on two distinct measurement approaches: cosine similarity based on vector-space embedding, and Jaccard similarity based on set comparison. To show that the structural properties P5-P7 can be addressed by the aggregation of different structural feature types, we consider the following combinations of feature types: (1) high-level structural features, (2) 1-grams and 2-grams, and (3) 1-grams, 2-grams and high-level structural features.

Graph-based cosine similarity (Cg) is calculated based on the identified edge-types with their length and the identified parallelism between activities with

M.	Transformation	Calculation
Cg	$\vec{\sigma_1} = \begin{matrix}(sequ.)\\(parallel,3)\\(loop,4)\end{matrix}\begin{bmatrix}1\\1\\0\end{bmatrix}$ $\vec{\sigma_2} = \begin{matrix}(sequ.)\\(parallel,3)\\(loop,4)\end{matrix}\begin{bmatrix}1\\1\\1\end{bmatrix}$	$1 - \frac{\vec{\sigma_1} \cdot \vec{\sigma_2}}{\|\vec{\sigma_1}\| \|\vec{\sigma_2}\|} = 1 - \frac{2}{\sqrt{6}}$
Jg	$F_{\sigma_1} = \{(sequ.), (parallel, \{B, C, D\})\}$ $F_{\sigma_2} = \{(sequ.), (parallel, \{B, C, D\}), (loop, 4, \langle A, E\rangle)\}$	$1 - \frac{F_{\sigma_1} \cap F_{\sigma_2}}{F_{\sigma_1} \cup F_{\sigma_2}} = \frac{1}{3}$

Table 6: Example for calculating the similarity between $\sigma_1 = \langle A, B, C, D, E\rangle$ and $\sigma_2 = \langle A, B, C, D, E, A, B, C, D, E\rangle$ with the parallel relations $R = \{B||C, C||D\}$ based on the two measures Cg and Jg.

the cardinality of the respective sets of parallel activities (cf. Definition 2.10). Both types of features are encoded based on their frequency. Alternatively, graph-based Jaccard similarity (Jg) is calculated based on the identified directly-follows relations with the respective edge-type and length and the identified sets of parallel activities. The number of edges of the type *sequence* are not considered, as they do not indicate any high-level structural feature, and are partially considered by the length of high-level structural features, as well as the directly-follows relations of traces. Furthermore, considering the number of *sequences* would bias the outcome towards the length of the traces. Nevertheless, traces that are strictly sequential, i.e., only contain edges of type *sequence*, should be considered as structurally similar. For this reason *(sequ.)* is added as an edge type (without considering length or frequency) to any trace that contains at least one edge of type sequence, i.e., an activity sequence longer than one, which is not a loop. An example for the calculation of Jg and Cg is provided in Table 6 based on the two traces depicted in Fig. 2.

The aggregated measures, denoted by $Agg()$, are computed as the sum of individual similarity measures with uniform weighting applied. For example, $Agg(C1, C2) = C1 + C2$ represents the aggregation of Cosine similarity values derived from 1-gram and 2-gram representations, as introduced in Sect. 4.2. This aggregation approach has already been applied in related studies (cf. Sect. 3.1.3).

Type	Measure	Label
Vector-space emb.	Graph-based cosine similarity	Cg
	Aggregated cosine similarity	Agg(C1,C2)
	Aggreagted graph-based cosine similarity	Agg(C1,C2,Cg)
Set comparison	Graph-based Jaccard similarity	Jg
	Aggregated Jaccard similarity	Agg(J1,J2)
	Aggreagted graph-based Jaccard similarity	Agg(J1,J2,Jg)

Table 7: Set of proposed trace similarity measures.

In the following subsection, we show that the proposed measures better reflect similarities and differences between traces based on structural features (P5-P7) than previous measures.

5.3. Formal Comparison

To evaluate the identified similarity measures (cf. Table 7) with respect to their ability to consider structural features (P5-P7), we first calculate the respective similarity values between σ_0 and $\sigma_3 - \sigma_8$ (cf. Fig. 1). The calculated similarity values are shown in Table 8. The results show that the novel measures better reflect structural similarities and differences between the traces than the measures from the literature (cf. Sect. 4.2).

Table 9 provides an overview of the proposed similarity measures with an indication of whether they adhere to the defined properties (P1-P10). With regard to the metric properties (P1-P4) it can be stated that the measures based on cosine similarity generally do not adhere to triangle-inequality (P4) [34]. Measures based on Jaccard similarity, on the other hand, generally satisfy triangle-inequality (P4) even in their aggregated form [50].

Regarding the structure-related properties (P5-P7), it can be stated that Cg, Jg, and Agg(J1,J2,Jg) correctly reflect the high-level structural features (P7), which is a clear improvement compared to the measures in the previous Sect. 4.2. Furthermore, the proposed aggregations of feature types (i.e., Agg(C1,C2), Agg(C1,C2,Cg), Agg(J1,J2), and Agg(J1,J2,Jg)) successfully reflect multiple defined structural properties. Agg(J1,J2,Jg) in fact reflects all

defined structural properties (P5-P7).

Regarding the general similarity properties (P8-P10) it can be stated that only measures based on Jaccard similarity can be considered as submodular functions (P8), while both measures based on Jaccard similarity and cosine similarity consider similarities as well as dissimilarities between the traces (P9). Regarding computational efficiency (P10), it can be stated that the additional calculations of the graph-features based on Algorithm 1 do not add to the time complexity compared to the similarity measures considered in the previous Sect. 4. Both DFS and BFS have a time complexity of $\mathcal{O}(|A|+|D|)$, while LCA can be computed in $\mathcal{O}(h)$, where h represents the length of the longest distinct path between two vertices in $G(A, D)$.

	$s(\sigma_0, \sigma_3)$	$s(\sigma_0, \sigma_4)$	$s(\sigma_0, \sigma_5)$	$s(\sigma_0, \sigma_6)$	$s(\sigma_0, \sigma_7)$	$s(\sigma_0, \sigma_8)$
Cg	0.000	0.000	0.000	0.553	0.423	0.423
Agg(C1,C2)	0.600	1.100	1.000	0.206	0.206	0.375
Agg(C1,C2,Cg)	2.449	2.949	3.162	2.018	1.888	2.474
Jg	0.000	0.000	0.000	0.667	0.667	0.667
Agg(J1,J2)	0.750	1.417	1.000	0.286	0.286	0.500
Agg(J1,J2,Jg)	0.750	1.417	1.000	0.952	0.952	1.167

Table 8: Calculated similarity values for the pairwise comparison between σ_0 and $\sigma_3 - \sigma_8$, where $s(\sigma_0, \sigma_3)$ is considered as baseline. According to the defined properties P5-P7, $s(\sigma_0, \sigma_4)$ should reflect an increase in the number of activities (P5), $s(\sigma_0, \sigma_5)$ should reflect an increase in the number directly-follows relations (P6), and $s(\sigma_0, \sigma_{6-8})$ should reflect an increase in the number of edge-types (P7). Values that are less than or equal to $s(\sigma_0, \sigma_3)$ indicate a violation of one of the defined properties P5-P7 and are highlighted in gray.

6. Evaluation

To evaluate the performance of the measures proposed in the previous section compared to the measures considered in Sect. 4 we perform an evaluation based on four synthetic event logs and four real-life event logs. The performance is evaluated in terms of the accuracy with which the measures assess the structural

	Metric			Structure				Similarity		
	P1	**P2**	**P3**	**P4**	**P5**	**P6**	**P7**	**P8**	**P9**	**P10**
Cg	yes	yes	yes	no	no	no	yes	no	yes	$\mathcal{O}(\lvert F_{i,2} \cup F_{j,2} \rvert)$
Agg(C1,C2)	yes	yes	yes	no	yes	yes	no	no	yes	$\mathcal{O}(\lvert F_{i,2} \cup F_{j,2} \rvert)$
Agg(C1,C2,Cg)	yes	yes	yes	no	yes	yes	no	no	yes	$\mathcal{O}(\lvert F_{i,2} \cup F_{j,2} \rvert)$
Jg	yes	yes	yes	yes	no	no	yes	yes	yes	$\mathcal{O}(\lvert F_{i,2} \cup F_{j,2} \rvert)$
Agg(J1,J2)	yes	yes	yes	yes	yes	yes	no	yes	yes	$\mathcal{O}(\lvert F_{i,2} \cup F_{j,2} \rvert)$
Agg(J1,J2,Jg)	yes	yes	yes	yes	yes	yes	yes	yes	yes	$\mathcal{O}(\lvert F_{i,2} \cup F_{j,2} \rvert)$

Table 9: Comparison between similarity measures based on the identified desirable properties $P1 - P10$.

similarity between traces in the respective event logs. Each trace thereby belongs to a unique class, with traces in the same class being more structurally similar to each other than to traces in a different class. To quantify the accuracy of the similarity measures, we apply three different evaluation measures. Overall, the evaluation seeks to answer two main questions: (1) can the proposed measures outperform alternative measures in terms of accuracy, and (2) can the considered high-level structural features (i.e., edge-types and parallelism) generally improve the accuracy of similarity measures.

The event logs and a Python implementation of the similarity measures and evaluation measures are available at:
https://bwsyncandshare.kit.edu/s/bNrqiXMkSebag6Q.

6.1. Evaluation Measures

Our evaluation of the similarity measures is based on three evaluation measures proposed by Back and Simonsen [14]: Nearest Neighbor, Precision@k, and Triplet. These measures are well established in the literature on similarity metric learning [63, 64]. Furthermore, Back and Simonsen [14] present a validation of these measures based on an extensive comparison of trace similarity measures. The proposed evaluation measures are deliberately selected to minimize confounding factors from downstream tasks such as process discovery [28], and trace clustering [1, 41, 2, 44]. As discussed in Sect. 3, this enables a more direct evaluation of the validity of the similarity measures with regard to content va-

lidity (i.e., the extent to which the measures capture a broad range of structural features) and criterion validity (i.e., the degree to which the measures align with predefined trace classifications).

To evaluate the accuracy of the similarity measures, the evaluation measures compare the derived similarity values of the in-class data points with the similarity values of the out-of-class data points. A similarity measure receives a positive evaluation if the similarity values between data points within a class are high and the similarity values between data points belonging to different classes are low.

Based on this idea the *Nearest Neighbor* measure calculates the ratio of data points whose nearest neighbor, i.e., the data point which is nearest based on some similarity measure, belongs to the same class and the data points whose nearest neighbor does not belong to the same class.

The *Precision@k* measure further extends this evaluation by calculating the ratio of in-class and out-of-class data points for the k-nearest data points. In accordance with [14], for this study we set $k = 10$.

The *Triplet* measure provides an even more precise evaluation of similarity by considering for each data point all the distances between in-class and out-of-class data points. Given an anchor data point x, the measure calculates for each in-class data point y the ratio of out-of-class data points z with $d(x, y) < d(x, z)$ to out-of-class data points z with $d(x, y) > d(x, z)$.

6.2. Datasets

For the empirical evaluation of the similarity measures introduced in Sect. 4 and Sect. 5, seven different event logs are considered. Table 10 provides an overview of these logs, including three synthetic event logs (SynL1, SynL2, SynL3) and four real-world event logs (WABO, PO, EyeT1, EyeT2).

These event logs are selected according to two main criteria. First, for the evaluation it is essential that the traces in the event logs can be classified according to some trace attribute. Furthermore, the traces should be equally distributed among the classes to increase the validity of the evaluation, i.e.,

	Activity	Trace	Min	Max	Class	T./C.	Class Type	Origin
SynL1	16	300	8	194	3	100	edge-type	Simulation
SynL2	16	300	8	202	6	50	edge-type and length	Simulation
SynL3	16	300	8	179	6	50	edge-type and activity	Simulation
SynL4	16	320	8	202	4	80	edge-type and activity	Simulation
WABO	465	350	1	110	5	70	process type	Administration
PO	42	300	8	324	3	100	process type	Administration
EyeT1	1194	349	27	1152	8	43-44	task type	Eye tracking
EyeT2	1194	349	27	1152	2	174-175	task complexity	Eye tracking

Table 10: Overview of the event logs including the number of activities, number of traces, the minimum and maximum length of traces, number of classes, number of traces per class, class type and their origin.

if all traces belong to the same class, the evaluation measures would yield a value of one for every similarity measure. Secondly, trace classes should be distinguishable based on their structural characteristics, i.e. activities, directly-follows relations, and edge-types, as defined by the properties P5-P7 in Sect. 4. Especially, the ability to distinguish trace classes based on edge-types (P7) is important to evaluate whether the proposed graph-based approach introduced in Sect. 5.2 can improve the performance of similarity measures. For this reason, we first look at three simulated event logs containing the desired properties, and then continue with real-world datasets.

6.2.1. SynL1, SynL2, SynL3, and SynL4.

These synthetic event logs are generated with the PLG tool [65]. The traces in the event logs can be clearly distinguished based on (a) edge types (loops, skips, or choice), (b) length of the edge types (short, or long) and (c) the set of activities ($A_1 = \{A, ..., H\}$, or $A_2 = \{K, ..., R\}$). To test different scenarios for the application of similarity measures, we apply four different classifications of the traces, resulting in four different event logs. In SynL1 the traces are classified only according to the edge types, leading to three classes: loops, skips, and choices. In each class the traces additionally differ in terms of (b) the length of the edge types and (c) the set of activities involved. This scenario is closely related to the example provided in Fig. 1, since in some cases the affiliation of

the traces with one class can only be detected based on the edge types.

In SynL2, the trace classes are additionally separated by the edge type length, leading to six classes (loops-short, loops-long, skips-short, skips-long, choices-short, choices-long). In each class the traces additionally differ in terms of (c) the set of activities involved. In this way, the similarity measures are additionally evaluated on the basis of their ability to distinguish between the length of the edge types.

In SynL3, the trace classes are separated by the edge types and the set of activities, also leading to six classes (loops-A_1, loops-A_2, skips-A_1, skips-A_2, choices-A_1, choices-A_2). In each class the traces additionally differ in terms of (b) length of the edge types. In this scenario, the effectiveness of the similarity measures is evaluated according to their ability to differentiate between edge types and distinct sets of activities.

In SynL4, an additional trace class is added to SynL1. This new class comprises traces that exhibit combinations of multiple edge types, resulting in four distinct classes: loops, skips, choices, and mixed. In each class the traces additionally differ in terms of (b) length of the edge types and (c) the set of activities involved. Compared to SynL1, SynL4 presents a more realistic scenario, enabling the evaluation of whether similarity measures can effectively identify traces that incorporate heterogeneous edge types.

6.2.2. WABO and PO.

These event logs were originally published for BPI Challenges 2015 [66] and 2019 [67]. Both data sets are derived from real-world business processes. WABO consists of five sublogs coming from five separate municipalities in the Netherlands, documenting their administration process for providing environmental permissions. PO documents a purchase order handling process of a large multinational company in the Netherlands, which can be classified into four types of flows. The classification of the traces in both cases is based on their process structure, making them suitable for our evaluation and allowing us to test whether graph-based features can also improve the similarity measurement of

traces in the context of business processes.

6.2.3. EyeT1 and EyT2.

The two event logs EyeT1 and EyeT2 contain eye tracking data, collected during an experiment on process model comprehension [23, 24] using the eye-tracking data collection tool EyeMind [68]. The data allows to analyze at which point in time during the experiment a person was looking at a specific process model element on a computer screen. This results in an event log, where each trace reflects the number and order of elements that a person looked at during a given comprehension task. Hence, the traces contain structural patterns, such as loops, skips, and choices, depending on the order in which a person looked at the different elements. 349 such traces could be recorded from 44 participants. The experiment is designed in such a way that each participant has to answer eight different comprehension tasks that can be divided into two complexity classes (low and high task complexity). In Schreiber et al. [24], the authors showed that the participants exhibit different information search and integration behaviors depending on the given tasks and its complexity, which is also reflected in the recorded traces. We therefore create the two trace logs EyeT1 and EyeT2 to evaluate how well the similarity measures can differentiate (a) between traces belonging to the eight different task types (EyeT1) and (b) between traces belonging to the two different task complexity classes (EyeT2).

6.3. Sampling Strategy

Due to the computational complexity of the evaluation measures, specifically the Triplet evaluation measure with $\mathcal{O}(|L|^3)$, we restrict the number of traces per event log. This is in line with the comparative study on traces similarity measures by Back and Simonson [14]. While the eye tracking event logs (EyeT1, EyeT2) are sufficiently small for the evaluation, the synthetic event logs (SynL1, SynL2, SynL3, SynL4) and the event logs derived from real-world business processes (WABO, PO) require some additional sampling. To prevent a sampling bias, traces are sampled randomly. Furthermore, they are sampled in such a

way that they are equally distributed among the classes, in order to ensure the effectiveness of the evaluation measures (cf. Sect. 6.2).

6.4. Results

Fig. 4 provides an overview of the three calculated evaluation measures for the similarity measures in Sect. 4.2 and the novel similarity measures proposed in Sect. 5. Additionally, three similarity measures are included in the evaluation, each computed as the cosine similarity between vector representations learned using Trace2Vec [41] (as described in Sect. 3.1.2). These representations differ in dimensionality, comprising vectors with 32 (T2V-32), 64 (T2V-64), and 128 (T2V-128) features, respectively.

Looking at the Triplet, which is the most elaborate evaluation measure included in the evaluation (cf. Sect. 6.1), it can be stated that the novel distance measures, marked in gray, consistently outperform the other similarity measures, except for the event log EyeT1. This observation is partially confirmed by Nearest Neighbour and Precision@10.

It should be noted that, in particular, the commonly applied LD and N-LD (cf. Sect. 3) are consistently outperformed by the other measures. The same holds true for the measures based on Trace2Vec (T2V-32, T2V-64, and T2V-128). It is also worth mentioning that the similarity measures based on 2-gram (C2 and J2) consistently perform well across the evaluation measures, especially for Precision@10.

It can also be stated that the aggregation of separate similarity measures leads in several cases to an improvement of the similarity evaluation (indicated by the underlined values in Fig. 4). This is particularly true for Agg(C1,C2) in the case of the two administrative processes (WABO and PO). Again looking at the Triplet, the similarity measures involving edge-type features perform in particular well with respect to the synthetic event logs (SynL1, SynL2, SynL3, SynL4) and EyeT2. This is also partially confirmed by Nearest Neighbour.

Regarding evaluation question 1, it can be stated that in terms of accuracy, the proposed measures (Cg, Agg(C1, C2), Agg(C1,C2,Cg), Jg, Agg(J1, J2),

	Nearest Neighbour							
	SynL1	SynL2	SynL3	SynL4	WABO	PO	EyeT1	EyeT2
LD	0.970	0.940	0.967	0.950	0.603	0.873	0.897	0.940
N-LD	0.970	0.933	0.967	0.953	0.600	0.910	0.937	0.971
EMR	0.963	0.917	0.927	0.965	0.200	0.710	0.785	0.871
CMR	0.977	0.957	0.987	0.978	0.203	0.333	0.957	0.971
JMR	0.977	0.957	0.983	0.978	0.194	0.730	0.954	0.971
EF	1.000	**1.000**	0.990	**1.000**	0.457	0.900	0.461	0.662
T2V-32	0.333	0.167	0.000	0.250	0.183	0.397	0.000	0.507
T2V-64	0.333	0.167	0.000	0.250	0.197	0.413	0.000	0.477
T2V-128	0.333	0.167	0.000	0.250	0.200	0.380	0.000	0.471
E1	0.930	0.900	0.913	0.941	0.629	0.860	0.954	0.974
E2	0.997	0.973	0.990	0.981	0.466	0.910	0.885	0.946
E3	0.997	0.970	0.987	0.988	0.366	0.890	0.676	0.811
C1	0.990	0.980	0.987	0.988	0.643	0.890	0.977	0.980
C2	**1.000**	0.990	0.997	0.997	0.611	0.917	0.952	0.966
C3	**1.000**	0.980	0.990	**1.000**	0.609	0.900	0.868	0.917
Cg*	**1.000**	**1.000**	0.493	**1.000**	0.226	0.577	0.172	0.653
Agg(C1,C2)	**1.000**	0.993	0.997	**1.000**	0.643	0.917	0.971	0.977
Agg(C1,C2,Cg*)	**1.000**	0.997	**1.000**	**1.000**	0.563	0.900	0.957	0.971
J1	0.403	0.290	0.333	0.316	0.646	0.863	0.966	0.980
J2	**1.000**	0.997	**1.000**	0.997	0.611	0.910	0.983	0.991
J3	**1.000**	0.993	**1.000**	**1.000**	0.606	0.920	0.986	0.994
Jg*	0.997	0.993	**1.000**	0.994	0.471	0.883	0.960	0.980
Agg(J1,J2)	**1.000**	0.997	**1.000**	0.997	**0.683**	**0.933**	**0.989**	**0.997**
Agg(J1,J2,Jg*)	**1.000**	0.997	**1.000**	**1.000**	0.563	0.923	0.957	0.971

	Precision@10							
	SynL1	SynL2	SynL3	SynL4	WABO	PO	EyeT1	EyeT2
LD	0.863	0.758	0.874	0.836	0.398	0.832	0.762	0.851
N-LD	0.875	0.800	0.892	0.861	0.400	0.850	0.865	0.917
EMR	0.871	0.782	0.831	0.853	0.200	0.757	0.621	0.736
CMR	0.941	0.884	0.926	0.932	0.199	0.299	0.903	0.937
JMR	0.946	0.878	0.931	0.930	0.198	0.790	0.901	0.937
EF	0.991	0.974	0.988	0.990	0.359	0.854	0.455	0.706
T2V-32	0.336	0.171	0.000	0.250	0.185	0.296	0.000	0.483
T2V-64	0.333	0.168	0.000	0.250	0.186	0.328	0.001	0.477
T2V-128	0.334	0.167	0.000	0.250	0.193	0.317	0.001	0.471
E1	0.808	0.730	0.830	0.824	0.433	0.857	0.862	0.916
E2	0.917	0.858	0.931	0.927	0.307	0.859	0.708	0.801
E3	0.924	0.874	0.937	0.938	0.251	0.848	0.524	0.673
C1	0.933	0.891	0.919	0.932	0.439	0.875	0.924	0.957
C2	0.962	0.938	0.962	0.967	0.393	0.876	0.842	0.900
C3	0.960	0.936	0.953	0.963	0.386	0.877	0.753	0.835
Cg*	0.993	**0.986**	0.482	0.980	0.245	0.643	0.157	0.622
Agg(C1,C2)	0.962	0.938	0.964	0.969	0.424	**0.881**	0.895	0.938
Agg(C1,C2,Cg*)	0.995	**0.986**	**0.993**	0.992	0.360	0.879	0.898	0.942
J1	0.367	0.222	0.333	0.379	0.437	0.864	0.905	0.952
J2	0.996	**0.986**	0.990	**0.995**	0.387	0.875	0.958	0.982
J3	0.987	0.968	0.982	0.991	0.379	0.880	**0.966**	**0.987**
Jg*	0.964	0.953	0.963	0.943	0.327	0.845	0.924	0.962
Agg(J1,J2)	**0.996**	**0.986**	0.990	0.967	**0.444**	0.880	0.936	0.968
Agg(J1,J2,Jg*)	0.995	**0.986**	**0.993**	0.992	0.360	**0.896**	0.898	0.942

	Triplet							
	SynL1	SynL2	SynL3	SynL4	WABO	PO	EyeT1	EyeT2
LD	0.523	0.545	0.803	0.530	0.504	0.728	0.669	0.584
N-LD	0.418	0.432	0.864	0.431	0.515	0.825	0.873	0.613
EMR	0.537	0.585	0.719	0.564	0.019	0.654	0.662	0.524
CMR	0.428	0.450	0.863	0.439	0.149	0.584	0.891	0.598
JMR	0.428	0.450	0.864	0.439	0.133	0.581	0.885	0.617
EF	0.435	0.469	0.886	0.451	0.525	0.776	0.829	0.610
T2V-32	0.461	0.460	0.153	0.462	0.469	0.292	0.065	0.375
T2V-64	0.462	0.460	0.160	0.461	0.469	0.311	0.067	0.354
T2V-128	0.462	0.464	0.162	0.466	0.473	0.309	0.071	0.350
E1	0.547	0.573	0.806	0.559	0.512	0.747	0.765	0.580
E2	0.563	0.598	0.818	0.586	0.504	0.712	0.680	0.540
E3	0.574	0.616	0.808	0.602	0.496	0.666	0.614	0.513
C1	0.422	0.456	0.870	0.445	0.524	0.853	0.890	0.565
C2	0.438	0.462	0.891	0.458	0.526	0.837	0.822	0.556
C3	0.439	0.461	0.890	0.457	0.407	0.636	0.762	0.533
Cg*	**0.816**	**0.917**	0.724	**0.814**	0.500	0.573	0.533	0.577
Agg(C1,C2)	0.434	0.462	0.886	0.456	**0.529**	**0.857**	0.862	0.561
Agg(C1,C2,Cg*)	0.653	0.700	**0.910**	0.666	0.526	0.817	0.863	0.582
J1	0.278	0.282	0.631	0.311	0.524	0.799	0.847	0.607
J2	0.419	0.469	0.854	0.437	0.526	0.827	0.915	0.627
J3	0.428	0.468	0.873	0.445	0.407	0.635	**0.919**	0.629
Jg*	0.618	0.637	0.782	0.623	0.468	0.692	0.903	**0.663**
Agg(J1,J2)	0.416	0.463	0.849	0.429	**0.529**	0.839	0.878	0.619
Agg(J1,J2,Jg*)	0.653	0.700	**0.910**	0.666	0.526	0.840	0.863	0.582

Figure 4: The tables show the calculated evaluation measures for the similarity measures on the basis of eight different event logs. The proposed similarity measures based on the formal comparison in Sect. 5 are marked in gray. Values that indicate best-performing similarity evaluations are marked in bold, performance improvement due to aggregations are underlined.

Agg(J1,J2,Jg)) outperform the other trace similarity measures with respect to the synthetic event logs (SynL1, SynL2, SynL3, and SynL4) based on all three evaluation measures, except for SynL4 based on Precision@10. Furthermore, the aggregation of 1-gram and 2-gram (Agg(C1,C2) and Agg(J1,J2)) consistently outperform the other trace similarity measures on the business process event logs (WABO and PO) across all four evaluation measures, except for PO based on Precision@10.. For the user behavior processes (EyeT1 and EyeT2) the measures J3 and Jg (excluding parallelism) achieved the best performance with respect to the Triplet evaluation measure.

Regarding evaluation question 2, it can be stated that the high-level features (edge-types and parallelism) lead to improvements in the similarity evaluation in several cases. For the synthetic event logs (SynL1, SynL2, SynL3, and SynL4) this is especially true with regard to Agg(J1,J2,Jg) when considering Triplet. Furthermore, for the business process PO the additional consideration of high-level features (including parallelism) could even improve similarity evaluation when calculating Agg(J1,J2,Jg). Only for the user behavior processes (EyeT1 and EyeT2) no improvement could be observed, even though Jg (excluding parallelism) yielded the highest performance with respect to the Triplet evaluation measure for EyeT2.

7. Discussion

The results confirm the comparative analyses in Sect. 4.2 and Sect. 5.3. The evaluation across all event logs shows that the aggregation of distinct similarity measures can substantially improve the similarity evaluation between the traces. The evaluation further shows that the consideration of high-level structural features can substantially improve the similarity evaluation between traces in several cases, and therefore provides a validation of the measurement approach proposed in Sect. 5.

Furthermore, the identified properties P1-P10 (cf. Table 3) help to better evaluate trace similarity measures. More specifically, P5-P7 provide insights

on the content validity of similarity measures, i.e., to which extent they reflect different structural features. This aspect has been largely neglected by existing studies on trace similarity measures.

The empirical evaluation in Sect. 6 shows that the structural distinction between traces on the basis of activities (P5), directly-follows relations (P6), and high-level structural features (P7) appears to be particularly valuable for (unstructured) behavioral processes, as well as (structured) business processes. In practice the proposed similarity measures could help to differentiate more nuanced structural similarities and differences between traces, which, subsequently might improve the performance of downstream tasks, such as anomaly detection and trace clustering. Moreover, the proposed aggregated measures potentially provide more transparency, as similarities and differences can be attributed to specific structural features. However, it should be noted that the aspect of transparency has not been empirically examined within the scope of this study.

From a research perspective, our study suggests that an assessment of trace similarity measures should not merely rely on empirical comparison, but should also consider formal properties of these measures to gain a better understanding of what trace features are respectively considered.

Integrating high-level structural features into the similarity comparison between traces opens up a variety of new opportunities for future research. It allows investigating the role of structural features in more detail and could therefore potentially improve a variety of process mining techniques, such as trace clustering, conformance checking, event abstraction, event log sampling, change point detection, and variety analysis (cf. Sect. 1).

The empirical evaluation further shows that there is not a single similarity measure dominating the others. Rather, it is the case that event logs require different similarity measures according to their structural features. One way to solve this issue is to apply aggregation and assign different weights to the distinct measures, focusing on different features. This can, for example, be solved by using domain knowledge to assign appropriate weights. Another approach would be to automatically assign weights based on feature learning [63, 41, 64].

When analyzing user behavior through data, e.g., based on eye-tracking, edge types can provide better differentiation between traces that exhibit various behavioral patterns, such as repetitive actions, skipping, or choices. This differentiation can, for example, be useful in identifying whether a user is searching for information or integrating it [23, 24].

7.1. Limitations

The evaluation stresses some limitations of the applied evaluation measures. Nearest Neighbour and Precision@10 only consider a small number (respectively two and ten) of the most similar data points within a class, thereby ignoring the similarity between the rest of the class's data points. This also explains the relative similarity of the outcomes in comparison to Triplet, which offers greater distinction among the considered similarity measures (cf. Fig. 4).

Moreover, due to the computational complexity of the evaluation measure Triplet (with $\mathcal{O}(|L|^3)$) the evaluation contains only event logs with a relatively small number of traces. This introduces some limitation regarding the generalizability of the results. However, this concern is partially mitigated by the diversity of event logs included in the evaluation, spanning a wide range of structural trace characteristics based on synthetic processes, eye-tracking, and real-world business processes.

Additionally, there exist some validity risk from the applied sampling of the synthetic event logs (SynL1, SynL2, SynL3) and the event logs derived from real-world business processes (WABO, PO). This risk is mitigated by the employed sampling strategy, i.e., traces were randomly sampled in such a way that they are equally distributed among the trace classes.

A further potential limitation arises from the extraction of the edge types based on the transformation of the traces into a graph representation, which inevitably leads to some abstraction of the process behavior. So far we have, for example, not considered relative frequencies of the occurring sequences in the traces, which could lead to some bias in the pairwise trace comparison. In future work, graph-based features could be weighted according to their relative

frequency, thereby emphasizing the stochastic properties of the traces [6].

Similarly, aggregating activities into sets of parallel activities based on causal relations (cf. Definition 2.10) introduces an abstraction of the trace structure. For example, when considering the trace $\sigma_1 = \langle A, B, C, D, E \rangle$ with $R = \{B||C, C||D\}$ (as shown in Fig. 2), $\{B, C, D\}$ is defined as a set of parallel activities within σ_1, even though no direct parallel relation exists between B and D, i.e., the information regarding the causality between B and D is lost. Similarly, the causality between B and D is ignored for the traces $\sigma_2 = \langle A, C, B, D, E \rangle$ and $\sigma_3 = \langle A, B, D, C, E \rangle$. Addressing this limitation would require refining the notion of a vertex in the trace graph (cf. Definition 2.11) to represent more complex parallel structures than simple sets of activities. Nevertheless, this abstraction is arguably reasonable, in order to detect additional high-level structural features, such as loops, skips, and choices, which is for example not possible based on instance graphs [17, 19].

Furthermore, deriving parallelism from causal relations between activities is sensitive to noise in the event log, which may introduce incorrect directly-follows relations. A common mitigation strategy is to apply data pre-processing techniques to filter out noisy traces before deriving parallelism between activities [17, 19]. An alternative approach would be to consider partially ordered traces, assuming correctly recorded timestamps [58].

Finally, certain similarity measures proposed in the literature, specifically optimal alignments [14] and generic edit distance [44], were excluded from the comparison due to their computational infeasibility when applied to the eye-tracking data. This limitation arises from the considerable length of these traces and the increased number of distinct activities involved.

8. Conclusion and Future Work

In this paper, we show the relevance of different structural features for the measurement of the similarity between two traces. We propose a novel approach to extract additional high-level structural features from the traces, re-

flecting loops, skips, choices, and parallelism. We further propose a set of novel similarity measures, which incorporate these structural features. The formal comparison in Sect. 5.3 and the evaluation in Sect. 6 show that these features can improve the similarity measurement, without an increase in computational time complexity.

As future work, we plan to further extend the proposed similarity measures, such that they can incorporate additional process perspectives, such as resources or data. For this purpose, the suggested graph-based approach could be employed on event knowledge graphs as detailed in [69], which encompass different process perspectives beyond their structural properties.

Furthermore, it would be interesting to investigate how the identified structural trace features can improve the training of neural networks and thus improve subsequent tasks, such as similarity measurement [41].

Declaration of Generative AI and AI-assisted technologies in the writing process. During the preparation of this work, the authors used ChatGPT 4o mini in order to polish sentences and ensure correct spelling. After using this service, the authors reviewed and edited the content as needed and take full responsibility for the content of the publication.

References

[1] M. Song, C. W. Günther, W. M. P. van der Aalst, Trace clustering in process mining, in: Business Process Management Workshops: BPM 2008 International Workshops, Milano, Italy, September 1-4, 2008. Revised Papers 6, Springer, 2009, pp. 109–120.

[2] P. Delias, M. Doumpos, E. Grigoroudis, P. Manolitzas, N. Matsatsinis, Supporting healthcare management decisions via robust clustering of event logs, Knowledge-Based Systems 84 (2015) 203–213.

[3] M. Boltenhagen, T. Chatain, J. Carmona, Generalized alignment-based trace clustering of process behavior, in: International Conference on Ap-

plications and Theory of Petri Nets and Concurrency, Springer, 2019, pp. 237–257.

[4] F. Zandkarimi, J.-R. Rehse, P. Soudmand, H. Hoehle, A generic framework for trace clustering in process mining, in: 2020 2nd International Conference on Process Mining (ICPM), IEEE, 2020, pp. 177–184.

[5] A. Adriansyah, J. Munoz-Gama, J. Carmona, B. F. van Dongen, W. M. van der Aalst, Alignment based precision checking, in: International conference on business process management, Springer, 2012, pp. 137–149.

[6] S. J. Leemans, W. M. P. van der Aalst, T. Brockhoff, A. Polyvyanyy, Stochastic process mining: Earth movers' stochastic conformance, IS 102 (2021) 101724.

[7] R. J. C. Bose, W. M. P. van der Aalst, Abstractions in process mining: A taxonomy of patterns, in: International Conference on Business Process Management, Springer, 2009, pp. 159–175.

[8] N. Tax, N. Sidorova, R. Haakma, W. M. P. van der Aalst, Mining local process models, Journal of Innovation in Digital Ecosystems 3 (2) (2016) 183–196.

[9] C. Diamantini, L. Genga, D. Potena, W. van der Aalst, Building instance graphs for highly variable processes, Expert Systems with Applications 59 (2016) 101–118.

[10] M. Kabierski, H. L. Nguyen, L. Grunske, M. Weidlich, Sampling what matters: relevance-guided sampling of event logs, in: 2021 3rd International Conference on Process Mining (ICPM), IEEE, 2021, pp. 64–71.

[11] A. Burattin, H. A. López, L. Starklit, Uncovering change: A streaming approach for declarative processes, in: International Conference on Process Mining, Springer, 2022, pp. 158–170.

[12] C. O. Back, S. Debois, T. Slaats, Entropy as a measure of log variability, Journal on Data Semantics 8 (2019) 129–156.

[13] C. Schreiber, A. Abbad-Andaloussi, Structural process variety and standardization, in: 2024 6th International Conference on Process Mining (ICPM), IEEE, 2024, pp. 153–160.

[14] C. O. Back, J. G. Simonsen, Comparing trace similarity metrics across logs and evaluation measures, in: International Conference on Advanced Information Systems Engineering, Springer, 2023, pp. 226–242.

[15] V. I. Levenshtein, Binary codes capable of correcting deletions, insertions, and reversals, in: Soviet physics doklady, Vol. 10, Soviet Union, 1966, pp. 707–710.

[16] S. B. Needleman, C. D. Wunsch, A general method applicable to the search for similarities in the amino acid sequence of two proteins, Journal of molecular biology 48 (3) (1970) 443–453.

[17] B. F. Van Dongen, W. M. P. van der Aalst, Multi-phase process mining: Building instance graphs, in: Conceptual Modeling–ER 2004: 23rd International Conference on Conceptual Modeling, Shanghai, China, November 8-12, 2004. Proceedings 23, Springer, 2004, pp. 362–376.

[18] F. Zerbato, R. Seiger, G. Di Federico, A. Burattin, B. Weber, Granularity in process mining: Can we fix it?, in: International Workshop on BPM Problems to Solve Before We Die, CEUR-WS, 2021, pp. 40–44.

[19] C. Diamantini, L. Genga, D. Potena, Behavioral process mining for unstructured processes, Journal of Intelligent Information Systems 47 (2016) 5–32.

[20] F. Taymouri, M. La Rosa, M. Dumas, F. M. Maggi, Business process variant analysis: Survey and classification, Knowledge-Based Systems 211 (2021) 106557.

[21] K. Holmqvist, M. Nyström, R. Andersson, R. Dewhurst, H. Jarodzka, J. Van de Weijer, Eye tracking: A comprehensive guide to methods and measures, oup Oxford, 2011.

[22] P. Fournier-Viger, J. C.-W. Lin, R. U. Kiran, Y. S. Koh, R. Thomas, A survey of sequential pattern mining, Data Science and Pattern Recognition 1 (1) (2017) 54–77.

[23] C. Schreiber, A. Abbad-Andaloussi, B. Weber, On the cognitive effects of abstraction and fragmentation in modularized process models, in: International Conference on Business Process Management, Springer, 2023, pp. 359–376.

[24] C. Schreiber, A. Abbad-Andaloussi, B. Weber, On the cognitive and behavioral effects of abstraction and fragmentation in modularized process models, Information Systems 125 (2024) 102424.

[25] M. Franceschetti, A. Abbad-Andaloussi, C. Schreiber, H. A. López, B. Weber, Exploring the cognitive effects of ambiguity in process models, in: International Conference on Business Process Management, Springer, 2024, pp. 493–510.

[26] W. Van der Aalst, Process Mining: Data Science in Action, Springer, Berlin, Heidelberg., 2016.

[27] B. Pentland, L. Ping, W. Kremser, T. Hærem, The dynamics of drift in digitized processes, MIS quarterly.

[28] R. J. C. Bose, W. M. P. van der Aalst, Trace clustering based on conserved patterns: Towards achieving better process models, in: Business Process Management Workshops: BPM 2009 International Workshops, Ulm, Germany, September 7, 2009. Revised Papers 7, Springer, 2010, pp. 170–181.

[29] M. Becker, R. Laue, A comparative survey of business process similarity measures, Computers in Industry 63 (2) (2012) 148–167.

[30] R. Dijkman, M. Dumas, L. García-Bañuelos, Graph matching algorithms for business process model similarity search, in: Business Process Management: 7th International Conference, BPM 2009, Ulm, Germany, September 8-10, 2009. Proceedings 7, Springer, 2009, pp. 48–63.

[31] A. Burattin, Streaming process mining, in: Process Mining Handbook, Vol. 349, Springer Cham, 2022.

[32] W. M. P. van der Aalst, Object-centric process mining: dealing with divergence and convergence in event data, in: Software Engineering and Formal Methods: 17th International Conference, SEFM 2019, Oslo, Norway, September 18–20, 2019, Proceedings 17, Springer, 2019, pp. 3–25.

[33] J. N. van Detten, P. Schumacher, S. J. Leemans, A framework for advanced case notions in object-centric process mining, in: International Conference on Process Mining, Springer, 2024, pp. 402–414.

[34] J. Han, M. Kamber, J. Pei, Data Mining: Concepts and Techniques, Morgan Kaufmann Publishers, 2012.

[35] W. M. P. van der Aalst, Process mining: a 360 degree overview, in: Process Mining Handbook, Springer, 2022, pp. 3–34.

[36] W. Van der Aalst, T. Weijters, L. Maruster, Workflow mining: Discovering process models from event logs, IEEE transactions on knowledge and data engineering 16 (9) (2004) 1128–1142.

[37] T. H. Cormen, C. E. Leiserson, R. L. Rivest, C. Stein, Introduction to algorithms, MIT press, 2022.

[38] D. Krantz, D. Luce, P. Suppes, A. Tversky, Foundations of measurement, Vol. I: Additive and polynomial representations, New York Academic Press, 1971.

[39] J. Mendling, Metrics for process models: empirical foundations of verification, error prediction, and guidelines for correctness, Vol. 6, Springer Science & Business Media, 2008.

[40] G. R. Marczyk, D. DeMatteo, D. Festinger, Essentials of research design and methodology, Vol. 1, John Wiley & Sons, 2005.

[41] P. De Koninck, S. vanden Broucke, J. De Weerdt, act2vec, trace2vec, log2vec, and model2vec: Representation learning for business processes, in: Business Process Management: 16th International Conference, BPM 2018, Sydney, NSW, Australia, September 9–14, 2018, Proceedings 16, Springer, 2018, pp. 305–321.

[42] G. Wang, X. Zhang, S. Tang, H. Zheng, B. Y. Zhao, Unsupervised clickstream clustering for user behavior analysis, in: Proceedings of the 2016 CHI conference on human factors in computing systems, 2016, pp. 225–236.

[43] R. Dijkman, M. Dumas, B. Van Dongen, R. Käärik, J. Mendling, Similarity of business process models: Metrics and evaluation, Information Systems 36 (2) (2011) 498–516.

[44] R. J. C. Bose, W. M. P. van der Aalst, Context aware trace clustering: Towards improving process mining results, in: proceedings of the 2009 SIAM International Conference on Data Mining, SIAM, 2009, pp. 401–412.

[45] R. Dewhurst, M. Nyström, H. Jarodzka, T. Foulsham, R. Johansson, K. Holmqvist, It depends on how you look at it: Scanpath comparison in multiple dimensions with multimatch, a vector-based approach, Behavior research methods 44 (2012) 1079–1100.

[46] N. C. Anderson, F. Anderson, A. Kingstone, W. F. Bischof, A comparison of scanpath comparison methods, Behavior research methods 47 (2015) 1377–1392.

[47] A. Schoknecht, T. Thaler, P. Fettke, A. Oberweis, R. Laue, Similarity of business process models—a state-of-the-art analysis, ACM CSUR 50 (4) (2017) 1–33.

[48] N. Tax, N. Sidorova, R. Haakma, W. M. P. van der Aalst, Event abstraction for process mining using supervised learning techniques, in: Proceedings of

SAI Intelligent Systems Conference (IntelliSys) 2016: Volume 1, Springer, 2018, pp. 251–269.

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

[50] M. Kunze, M. Weidlich, M. Weske, Behavioral similarity–a proper metric, in: Business Process Management: 9th International Conference, BPM 2011, Clermont-Ferrand, France, August 30-September 2, 2011. Proceedings 9, Springer, 2011, pp. 166–181.

[51] G. M. Tavares, R. S. Oyamada, S. B. Junior, P. Ceravolo, Trace encoding in process mining: A survey and benchmarking, Engineering Applications of Artificial Intelligence 126 (2023) 107028.

[52] J. Pflug, S. Rinderle-Ma, Process instance similarity: Potentials, metrics, applications, in: On the Move to Meaningful Internet Systems: OTM 2016 Conferences, Springer, 2016, pp. 136–154.

[53] F. Emmert-Streib, M. Dehmer, Y. Shi, Fifty years of graph matching, network alignment and network comparison, Information sciences 346 (2016) 180–197.

[54] M. Leemans, W. M. P. van der Aalst, Discovery of frequent episodes in event logs, in: International symposium on data-driven process discovery and analysis, Springer, 2014, pp. 1–31.

[55] A. Abbad Andaloussi, J. Buch-Lorentsen, H. A. López, T. Slaats, B. Weber, Exploring the modeling of declarative processes using a hybrid approach, in: International conference on conceptual modeling, Springer, 2019, pp. 162–170.

[56] A. Abbad Andaloussi, F. Zerbato, A. Burattin, T. Slaats, T. T. Hildebrandt, B. Weber, Exploring how users engage with hybrid process artifacts based on declarative process models: a behavioral analysis based on

eye-tracking and think-aloud, Software and Systems Modeling 20 (2021) 1437–1464.

[57] P. Bera, P. Soffer, J. Parsons, Using eye tracking to expose cognitive processes in understanding conceptual models, MIS quarterly 43 (4) (2019) 1105–1126.

[58] S. J. Leemans, S. J. van Zelst, X. Lu, Partial-order-based process mining: a survey and outlook, Knowledge and Information Systems 65 (1) (2023) 1–29.

[59] L. Yujian, L. Bo, A normalized levenshtein distance metric, IEEE transactions on pattern analysis and machine intelligence 29 (6) (2007) 1091–1095.

[60] S. Kosub, A note on the triangle inequality for the jaccard distance, Pattern Recognition Letters 120 (2019) 36–38.

[61] A. Levy, B. R. Shalom, M. Chalamish, A guide to similarity measures and their data science applications, Journal of Big Data 12 (1) (2025) 188.

[62] H. Zha, J. Wang, L. Wen, C. Wang, J. Sun, A workflow net similarity measure based on transition adjacency relations, in: Computers in Industry, Vol. 61, Elsevier, 2010.

[63] A. Bellet, A. Habrard, M. Sebban, A survey on metric learning for feature vectors and structured data, in: arXiv preprint arXiv:1306.6709, 2013.

[64] M. Kaya, H. Ş. Bilge, Deep metric learning: A survey, in: Symmetry, Vol. 11, MDPI, 2019.

[65] A. Burattin, Plg2: Multiperspective process randomization with online and offline simulations., in: BPM (Demos), Citeseer, 2016, pp. 1–6.

[66] B. F. van Dongen, Bpi challenge 2015, in: 4TU.ResearchData. dataset, 2015.

[67] B. F. van Dongen, Bpi challenge 2019, in: 4TU.ResearchData. dataset, 2019.

[68] A. Abbad-Andaloussi, D. Lübke, B. Weber, Conducting eye-tracking studies on large and interactive process models using eyemind, SoftwareX 24 (2023) 101564.

[69] D. Fahland, Process mining over multiple behavioral dimensions with event knowledge graphs, in: Process Mining Handbook, Springer, 2022, pp. 274–319.