Matching Heterogeneous Event Data

Xiaochen Zhu[§] Shaoxu Song[§] Xiang Lian[†] Jianmin Wang[§] Lei Zou[‡]

[§]KLiss, MoE; TNList; School of Software, Tsinghua University, China
zhu-xc10@mails.tsinghua.edu.cn {sxsong, jimwang}@tsinghua.edu.cn
[†]University of Texas - Pan American, USA lianx@utpa.edu
[‡]Peking University, China zoulei@pku.edu.cn

ABSTRACT

Identifying duplicate events are essential to various business process applications such as provenance querying or process mining. Distinct features of heterogeneous events including opaque names, dislocated traces and composite events, prevent existing data integration from techniques performing well. To address these issues, in this paper, we propose an event similarity function by iteratively evaluating similar neighbors. We prove the convergence of iterative similarity computation, and propose several pruning and estimation methods. To efficiently support matching composite events, we devise upper bounds of event similarities. Experiments on real and synthetic datasets demonstrate that the proposed event matching approaches can achieve significantly higher accuracy than the state-of-the-art matching methods.

Categories and Subject Descriptors

H.2.5 [**Database Management**]: Heterogeneous Databases

Keywords

Schema matching; event matching

1. INTRODUCTION

Owing to various mergers and acquisitions, information systems (e.g., Enterprise Resource Planning (ERP) and Office Automation (OA) systems), probably developed independently in different branches or subsidiaries in large-scale corporations, keep on generating heterogeneous event logs. We surveyed a major bus manufacturer who recently started a project on integrating their event data in the OA systems of 31 subsidiaries. These OA systems have been built independently on 5 distinct middleware products in the past 20 years. More than 8190 business processes are implemented in these systems, among which 68.8% are indeed different implementations of the same business activities in different subsidiaries. For instance, in the following Example 1, we illustrate two versions of turbine order processing processes in 31 subsidiaries. Duplicate events commonly exist in these heterogeneous processes for the same business activities.

The company has started to integrate these heterogeneous event data into a unified business process data warehouse [3, 4], where different types of analyses can be performed, e.g., querying similar complex procedures or discovering interesting event patterns in different subsidiaries (complex event processing, CEP [6]), comparing business processes in different subsidiaries to find common parts for process simplification and reuse [19], or obtaining a more abstract global picture of business processes (workflow views [2]) in the company. Without exploring the correspondence among heterogeneous events, applications such as query and analysis over the event data may not yield any meaningful results.

The event matching problem is to construct the similarity and matching relationship of events from heterogeneous sources. Although there are 49 system integrators (subject-matter experts) employed by the bus manufacturer in the project of OA system integration, given the large number of more than 100,000 event activities, manually checking the correspondences of all possible event pairs is highly impractical for two reasons: 1) manual checking is obviously inefficient, and 2) manual checking results could be inaccurate and contradictory. An automatic approach is highly demanded for matching these heterogeneous event data.

Different from conventional schema matching on attributes in relational databases [7], there are strong relationships of consecutive occurrence among events. The event data integration is challenging due to the following features commonly observed in the general event data.

(1) Event names could be *opaque*, due to various encoding, syntax or language conventions in heterogeneous systems.

(2) Event traces might be *dislocated*. Only a part (e.g., the beginning) of a trace 1 corresponds to a distinct part (e.g., the end) of another trace 2.

(3) *Composite* event may exist. That is, one event in a source corresponds to several decomposed ones in another source, known as composite event in CEP [6].

Example 1. *Figure 1 illustrates two fragments of event logs $\mathcal{L}_1$ and $\mathcal{L}_2$ for turbine order processing in two different subsidiaries of the bus manufacturer, respectively. Two example traces are shown in each fragment, and each trace denotes a sequence of events (steps) for processing one order. An event log consists of many traces, among which the sequences of events may be different, since some of the events can be executed concurrently (e.g.* Ship Goods(E) *and* Email

SIGMOD'14, June 22–27, 2014, Snowbird, UT, USA.

http://dx.doi.org/10.1145/2588555.2588570.

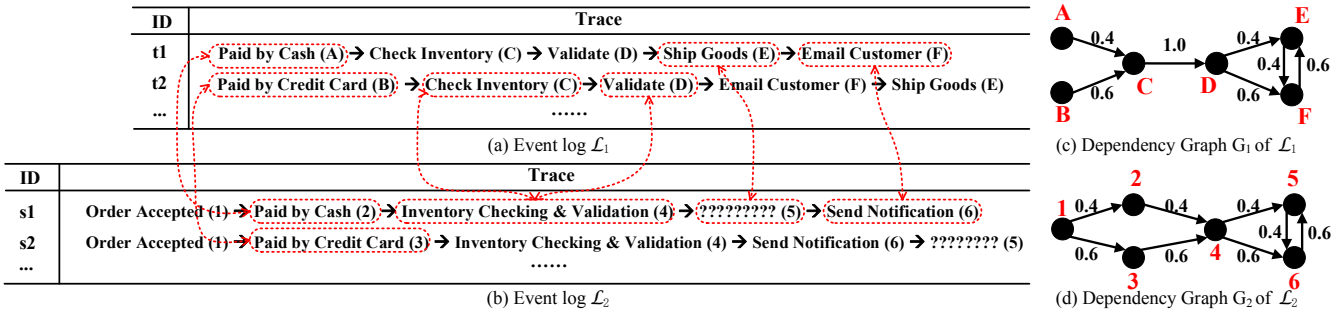


Figure 1: Fragments of two event logs and their dependency graphs

Customer(F) in $\mathcal{L}_1$), or exclusively (e.g., Paid by Cash(2) or Paid by Credit Card(3) in $\mathcal{L}_2$).

Note that opaque names exist in $\mathcal{L}_2$ as shown in Figure 1(b). The event "????????(5)" is collected from a database whose encoding is distinct from others, which makes the event name garbled. According to expert investigation, the original name of "????????(5)" should be "Delivery(5)", and the true event corresponding relation between $\mathcal{L}_1$ and $\mathcal{L}_2$ is highlighted by red dashed lines in Figures 1(a) and (b).

Dislocated matching exists between the first traces in $\mathcal{L}_1$ and $\mathcal{L}_2$. Event Paid by Cash(A) that appears at the beginning of traces in $\mathcal{L}_1$ corresponds to event Paid by Cash(2), which appears in the middle of traces in $\mathcal{L}_2$, having another event Order Accepted(1) before it.

Moreover, two events Check Inventory(C) and Validate(D) in $\mathcal{L}_1$ correspond to one composite event Inventory Checking & Validation(4). For simplicity, we use ABCDEF to denote event names in $\mathcal{L}_1$, while 123456 are events in $\mathcal{L}_2$.

Unfortunately, existing techniques cannot effectively address the aforesaid three challenges in event matching. A straightforward idea of matching events is to compare their names (a.k.a. event labels). String edit distance (syntactic similarity) [13] as well as word stemming and the synonym relation (semantic similarity) [20] are widely used in the label similarity based approaches [14, 19, 5, 23]. As shown in Example 1, such a *typographical similarity* cannot address the identified Challenge 1, i.e., opaque event labels.

Structural similarity may be considered besides the typographical similarity. The idea is to construct a graph for describing the relationships among events, e.g., the frequency of appearing consecutively in an event log [8]. Once the graphical structure is obtained, graph matching techniques can be employed to identify the event (behavioral/structural) similarities. Unfortunately, existing graph matching techniques cannot handle well the dislocated matching of events[1], i.e., the aforesaid Challenge 2. Graph edit distance (GED) [5] for general graph data and normal distance for matching with opaque names (OPQ) [11] concern a local evaluation of similar neighbors. However, as illustrated in Example 1, dislocated matching events may have distinct neighbors (see more details below). Rather than local neighbors, another type of Simrank [10] like behavioral similarity (BHV) [19] considers a global evaluation via propagating similarities in the entire graph in multiple iterations. Unfortunately, directly applying the global propagation does not help in matching dislocated events that do not have any predecessor, e.g., Paid by Cash(A) in Figure 1.

[1]According to our survey on 5642 processes with redundancy (68.8% of 8190) of the bus manufacturer, more than 44% of them involve dislocated event traces.

Example 2. *Figures 1(c) and 1(d) capture the statistical and structural information of $\mathcal{L}_1$ and $\mathcal{L}_2$, respectively (see Definition 1 for constructing G_1 and G_2). Each vertex in the directed graph denotes an event, while an edge between two events (say AC in Figure 1(c) for instance) indicates that they appear consecutively in at least one trace (e.g., trace 1 in Figure 1(a)). The numbers attached to edges represent the normalized frequencies of consecutive event pairs. For instance, 0.4 of AC means that A, C appear consecutively in 40% of the traces of the event log.*

Since GED and OPQ concern more about the local similarity, e.g., the high similarity of (A, C) and $(1, 2)$, an event mapping $M = \{A \to 1, B \to 3, C \to 2, D \to 4, E \to 5, F \to 6\}$ will be returned by GED with distance 0.16 and OPQ with score 10.7. The true mapping $M' = \{A \to 2, B \to 3, C/D \to 4, E \to 5, F \to 6\}$ in ground truth shows a higher GED distance 0.233 (lower is better) and lower OPQ score 10 (higher is better) instead. BHV does not help in capturing dislocated mapping, e.g., between A and 2 with BHV similarity 0. Instead, A and 1 with no input neighbors have higher similarity 1, i.e., unable to find the dislocated matching.

When matching composite events, i.e., Challenge (3), computing efficiency becomes a major issue. For instance, one may be interested in composite events $\{C, D\}$, $\{D, E\}$ and $\{E, F\}$ in Figure 1(c). Given a candidate set of n composite events, there may exist 2^n possible combinations, for each of which we need to evaluate the corresponding event similarities. Unfortunately, structural similarities such as GED are already costly to compute for one combination.

Contributions. In this paper, we propose a similarity function that supports dislocated matching and opaque events, by iteratively computing neighbor similarities. Our major contributions in this paper are summarized as follows.

(1) We formally define the similarity function, which is iteratively computed by incorporating other similarity criteria in the iteration. Intuitively, as discussed in Example 2, directly applying OPQ that concerns a local evaluation of similar neighbors is ineffective in matching dislocated events with possibly distinct neighbors. In contrast, the proposed iteration based similarity function concerns the global evaluation via propagating similarities, and thus overcomes the impact of local neighbor distinctness.

(2) We prove the convergence of the proposed iteration based similarity function and devise a pruning technique

Table 1: Frequently used notations

Symbol	Description
$v \in V$	an event v in event set V
$G(V, E, f)$	an event dependency graph
v^X	an artificial event
$\bullet v, v\bullet$	the pre/post set of an event
$\mathcal{S}^n(v_1, v_2)$	the similarity between events v_1 and v_2 after the n^{th} iteration
$l(v_1)$	the longest distance from v^X to v_1
U	a composite event
$\mathcal{U}$	a collection of composite event candidates

for efficient computations. Moreover, we introduce a fast estimation with a constant number of iterations. In particular, by varying the number of maximum iterations from 0 to larger, it provides a trade-off between matching accuracy and computing time.

(3) We study the complexity bound of the composite event matching problem. The problem of finding the optimal composite event matching is proved to be NP-hard. Owing to the large number of possible composite event matching candidates, applying existing OPQ is inefficient (as discussed in related work in Section 6 and observed in the experiments in Section 5). Therefore, we propose greedy heuristics for efficiently matching composite events, identify event pairs that can avoid updating similarities between greedy steps, and design upper bounds of similarities for efficient pruning.

(4) We report an extensive experimental evaluation on both real and synthetic datasets. The results demonstrate that our proposed matching methods can achieve higher accuracy than the state-of-the-art methods.

The remainder of this paper is organized as follows. Section 2 gives an overview of the event matching framework. Section 3 illustrates the computation of event similarity between two event logs. Section 4 extends the matching approach to capture composite event similarities. Section 5 reports the experimental evaluation. Finally, Section 6 discusses related work and Section 7 concludes this paper.

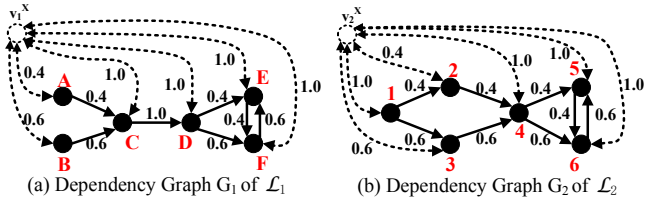
2. OVERVIEW OF EVENT MATCHING

We formalize syntax and definitions for the event matching problem. Table 1 lists the frequently used notation in this paper. Let V be a set of events, i.e., events that can be recorded in a log. A trace is a finite sequence of events from V. An event log $\mathcal{L}$ is a multi-set of traces from V^*.

Capturing structural information. Detecting correspondences on raw logs is difficult, since the event names could be "opaque". Other than typographic similarity, we can exploit the structural information for matching events.

Following the same line of [11], we employ a simple graph model, namely *dependency graph*, which consists of both dependency relations and frequencies.

Definition 1 (Event Dependency Graph)**.** *An* event dependency graph G *is a labeled directed graph* (V, E, f)*, where each vertex in* V *corresponds to an event,* E *is an edge set, and* f *is a labeling function of normalized frequencies that*



(a) Dependency Graph G_1 of $\mathcal{L}_1$ (b) Dependency Graph G_2 of $\mathcal{L}_2$

Figure 2: Dependency graphs with artificial events

(1) for each $v \in V$*,* $f(v, v)$ *is the normalized frequency of event* v*, i.e., the fraction of traces in* $\mathcal{L}$ *that contain* v*, and (2) for each edge* $(v_1, v_2) \in E$*,* $f(v_1, v_2)$ *is the normalized frequency of two consecutive events* $v_1 v_2$*, i.e., the fraction of traces in which* $v_1 v_2$ *occur consecutively at least once.*
For any $v \in V$*, the pre-set of* v *is defined as* $\bullet v = \{v' | (v', v) \in E\}$ *and the post-set of* v *is defined as* $v\bullet = \{v' | (v, v') \in E\}$*.*

With the presence of dislocated matching, any event in an event log can be a starting/ending event. That is, a trace can start/end with any event v within it by ignoring those events before/after v. Based on this intuition, we extend the dependency graph G by adding an artificial event v^X and several artificial edges as follows.

(1) An artificial event v^X is added into V, which denotes the virtual beginning/end of all traces in an event log.

(2) For each event $v \in V$ except v^X, we add two artificial edges (v, v^X) and (v^X, v), i.e., each event can be a virtual starting event (edge (v^X, v)) and a virtual ending event (edge (v, v^X)). Moreover, we associate $f(v^X, v) = f(v, v^X) = f(v)$ based on the intuition that a trace can start/end with event v at all the locations where v occurs.

Example 3. *In order to support dislocated matching, we add an artificial events and edges into dependency graphs, denoted by vertices and edges in dashed lines in Figures 2(a) and 2(b). As the virtual beginning/end of all traces,* v_1^X *and* v_2^X *connect to all the events in* V_1 *and* V_2*, respectively. The weight on each artificial edge is assigned by the normalized frequency of the occurrence of each event. For instance, since event* C *appears in all the traces of* $\mathcal{L}_1$*, we have* $f(v_1^X, C) = 1.0$*. Event* A *only appears in 40% traces of* $\mathcal{L}_1$*, which indicates* $f(v_1^X, A) = 0.4$*.*

Filtering with minimum frequency control. Intuitively, edges in the dependency graph with low frequencies may not help much in matching events since they have few statistical information. We can filter out those edges whose frequencies are lower than a predefined threshold. The average degree of a dependency graph may decrease after filtering out those edges, which can accelerate the computation of similarity. Hence, the minimum edge frequency control could be interpreted as a trade-off between accuracy and efficiency.

Computing pair-wise similarities. Based on the dependency graphs of statistical information in two event logs, the similarity between each event pair, denoted $\mathcal{S}(v_1, v_2)$, can be computed. Motivated by the unique features of heterogeneous events as indicated in the introduction, we propose a similarity measure by iteratively utilizing structural information in Section 3. In particular, the proposed measure is extensible by integrating with other similarities such as typographic or linguistic similarities [20].

Selecting matching correspondences. Once all the pair-wise similarities are gained, there are various existing approaches to capture the corresponding events. It is worth noting that the event pairs containing either v_1^X or v_2^X should be omitted since these two events are introduced artificially and do not actually exist in event logs. As it is not the focus of this study, we briefly outline the approaches of selecting correspondences from pair-wise similarities in Section 6.

Matching composite events. Most existing graph matching approaches are restricted to finding 1:1 correspondences [14], i.e., one event should be mapped exactly to another. Such a strong restriction fails to detect the matching of composite events. Owing to heterogeneous settings, a single event in one process could be divided into several events in another process. Therefore, the m:n correspondences, which are called complex matches in [23], should also be addressed.

When matching composite events with various candidate combinations, it needs to frequently compute the (composite) event similarities. We devise upper bounds of the proposed similarity function for efficient pruning in Section 4.

3. EVENT MATCHING SIMILARITY

Three categories of techniques may be considered for evaluating event similarities: 1) content-based such as typographic similarities [20], 2) structure-based similarities which concern local structures such as GED [5] and OPQ [11], and 3) structure-based similarities with a global view of the entire graph like Simrank [10]. Unfortunately, as illustrated in the introduction, content based similarities often fail to perform owing to opaque event names, while GED and OPQ cannot handle dislocated matching well. On the other hand, the widely used Simrank [10] is not directly applicable to event data for two reasons: 1) Simrank concerns only the pure structural similarity, while the utilization of other criteria such as typographic similarities is not supported, even when they are available; 2) Simrank does not take edge similarities into consideration, which are the key properties of consecutive occurrence between events.

In this section, we present an adaption of Simrank like structural similarity function for matching events, and discuss its convergence. Efficient pruning of unnecessary similarity updates based on early convergence and fast (one iteration) estimation of similarities are devised.

3.1 Structural Similarity Function

Intuitively, following the same line of Simrank, an event, say $v_1 \in V_1$, is similar to event $v_2 \in V_2$, if they frequently share similar predecessors (in-neighbors). We use $s(v_1, v_2)$ to denote how often a predecessor v_1' of $v_1, v_1' \in \bullet v_1$, can find a similar $v_2' \in \bullet v_2$, predecessor of v_2. Note that this s measure is asymmetric, i.e., $s(v_1, v_2) \neq s(v_2, v_1)$.

Next, to adapt Simrank like evaluation for event similarity, we further take edge similarities into consideration. Although the predecessors v_1' and v_2' of v_1 and v_2, respectively, have high similarity, if the frequency of (v_1', v_1) deviates far from the frequency of (v_2', v_2), the similarity of v_1' and v_2' will have less effect on the similarity of v_1 and v_2.

Finally, although it supports matching events with opaque names, we take label similarities (typographic similarities of event names) into account, in case they are available.

Following these intuitions, we define a forward similarity.

Definition 2. *The* forward similarity *of two events is*

$$\mathcal{S}(v_1, v_2) = \alpha(s(v_1, v_2) + s(v_2, v_1))/2 + (1 - \alpha)\mathcal{S}^L(v_1, v_2),$$

where $\mathcal{S}^L(v_1, v_2)$ is the label similarity of v_1 and v_2, $\alpha \in [0, 1]$ is a weight, $s(v_1, v_2)$ and $s(v_2, v_1)$ are one-side similarities

$$s(v_1, v_2) = \frac{1}{|\bullet v_1|} \sum_{v_1' \in \bullet v_1} \max_{v_2' \in \bullet v_2} C(v_1, v_1', v_2, v_2')\mathcal{S}(v_1', v_2'),$$

given that $C(v_1, v_1', v_2, v_2') = c*(1-\frac{|f(v_1,v_1')-f(v_2,v_2')|}{f(v_1,v_1')+f(v_2,v_2')})$, where c is a constant that has $0 < c < 1$.

We now explain how these formulas implement our intuition. In the formula of $s(v_1, v_2)$, for each in-neighbor v_1' of v_1, we find an event v_2' with the highest similarity to v_1' among all the in-neighbors of v_2. Besides the node similarity $S(v_1', v_2')$ which evaluates how similar v_1' and v_2' are, we also consider the similarity of the edges (v_1', v_1) and (v_2', v_2) which denote the consecutive occurring relationships of events, by $C(v_1, v_1', v_2, v_2')$. Obviously, if (v_1', v_1) and (v_2', v_2) have similar normalized frequencies, $C(v_1, v_1', v_2, v_2')$ is close to c; otherwise close to 0, where c gives the rate of similarity decay across edges. To support label similarity, we use a weighted aggregation of the structural similarity $\frac{1}{2}(s(v_1, v_2) + s(v_2, v_1))$ and the label similarity $\mathcal{S}^L(v_1, v_2)$.

3.2 Iterative Computation

To compute $\mathcal{S}(v_1, v_2)$ from predecessors, we present an iteration method by iteratively applying the formulas in Definition 2. Let $\mathcal{S}^n(v_1, v_2)$ denote the forward similarity of (v_1, v_2) after the n^{th} iteration. The computation has two steps: the initialization step which assigns $\mathcal{S}^0(v_1, v_2)$ for every event pair (v_1, v_2), and the iteration step which computes the value of $\mathcal{S}^n(v_1, v_2)$ by using $\mathcal{S}^{n-1}(v_1, v_2)$ according to Definition 2, when $n \geq 1$.

1) Initialization. For the artificial events v_1^X and v_2^X, the initial similarities $\mathcal{S}^0(v_1^X, v_2^X)$ is set to 1.0 since both of them are defined as the virtual beginning and ending of traces. $\mathcal{S}^0(v_1^X, v_2)$ (and $\mathcal{S}^0(v_1, v_2^X)$) is set to 0 if one event is real but the other is artificial. For any other real event pair (v_1, v_2), $\mathcal{S}^0(v_1, v_2)$ is set to 0, since there is no priori knowledge for assigning nonzero values as initial similarities.

2) Iteration. In each iteration, we refresh $\mathcal{S}$ for each event pair (v_1, v_2) using the similarities of their neighbors in the previous iteration. For instance, according to Definition 2, $\mathcal{S}^n$ which denotes the forward similarity of (v_1, v_2) after the n^{th} iteration can be computed by:

$$\mathcal{S}^n(v_1, v_2) = \alpha(s^n(v_1, v_2) + s^n(v_2, v_1))/2 + (1 - \alpha)\mathcal{S}^L(v_1, v_2),$$
$$s^n(v_1, v_2) = \frac{1}{|\bullet v_1|} \sum_{v_1' \in \bullet v_1} \max_{v_2' \in \bullet v_2} C(v_1, v_1', v_2, v_2')\mathcal{S}^{n-1}(v_1', v_2'). \quad (1)$$

The similarities between artificial events and real events (e.g. $\mathcal{S}(v_1^X, v_2)$, $\mathcal{S}(v_1, v_2^X)$ and $\mathcal{S}(v_1^X, v_2^X)$) are not updated during the iteration. The algorithm stops when the difference between $\mathcal{S}^n(v_1, v_2)$ and $\mathcal{S}^{n-1}(v_1, v_2)$ for all event pairs (v_1, v_2) is less than a predefined threshold.

Example 4 (Example 2 continued)**.** *At the beginning of the iteration, $\mathcal{S}^0(v_1^X, v_2^X)$ is assigned with 1.0, and $\mathcal{S}^0(v_1, v_2)$ is*

assigned with 0 for any other event pairs where $v_1 \neq v_1^X$ *and* $v_2 \neq v_2^X$. *Consider the event pair* $(A, 1)$. *Let* $\alpha = 1$, *on the* 1^{st} *iteration, we have* $s^1(A,1) = \frac{1}{|\bullet A|} C(v_1^X, A, v_2^X, 1)\mathcal{S}^0(v_1^X, v_2^X) = 0.457$ *and* $s^1(1,A) = \frac{1}{|\bullet 1|} C(v_1^X, 1, v_2^X, A)\mathcal{S}^0(v_1^X, v_2^X) = 0.457$, *so that* $\mathcal{S}^1(A,1) = 0.5 * (s^1(A,1) + s^1(1,A)) = 0.457$. *For the event pair* $(A, 2)$, *we have* $s^1(A,2) = \frac{1}{|\bullet A|} \max(C(v_1^X, A, v_2^X, 2)\mathcal{S}^0(v_1^X, v_2^X), C(v_1^X, A, 1, 2)\mathcal{S}^0(v_1^X, 1)) = 0.8$ *and* $s^1(2, A) = \frac{1}{|\bullet 2|}(C(v_2^X, 2, v_1^X, A)\mathcal{S}^0(v_2^X, v_1^X) + C(1, 2, v_1^X, A)\mathcal{S}^0(1, v_1^X)) = 0.4$, *so that* $\mathcal{S}^1(A,2) = 0.5 \times (0.8 + 0.4) = 0.6$. *It is notable that* A *and* 2 *have higher similarity than* A *and* 1, *which solves the problem of dislocated matching. Consequently, we can compute an average similarity of all the event pairs in the true mapping* M' *in Example 2, i.e., 0.551, which is higher than the average similarity 0.506 of* M.

The time complexity of computing forward similarity is $O(k|V_1||V_2|d_{avg})$, where k is the number of iterations and d_{avg} is the average degree of all the events in the dependency graph. When the density of the dependency graph as well as the numbers of iterations is high (i.e., d_{avg} and k are high), the iterative computation is time-consuming.

3.3 Convergence

We show that the iterative computation converges to limits satisfying the similarity function in Definition 2.

Theorem 1. *For all* $v_1 \in V_1, v_2 \in V_2, \lim_{n\to\infty} \mathcal{S}^n(v_1, v_2) = \mathcal{S}(v_1, v_2)$.

Proof. **Existence**. Firstly, the monotonicity of $\mathcal{S}^n(v_1, v_2)$ can be simply proved by induction.

Monotonicity: $0 \leq \mathcal{S}^{n-1}(v_1, v_2) \leq \mathcal{S}^n(v_1, v_2) \leq 1$, for all $v_1 \in V_1, v_2 \in V_2, n \geq 1$.

Basis: Let $n = 1$. According to formula (1), there is $s^1(v_1, v_2) = \frac{1}{|\bullet v_1|} \sum_{v_1' \in \bullet v_1} \max_{v_2' \in \bullet v_2} C(v_1, v_1', v_2, v_2')\mathcal{S}^0(v_1', v_2')$. That is, $s^1(v_1, v_2) \in (0, 1]$ *iff* $v_1^X \in \bullet v_1$ and $v_2^X \in \bullet v_2$; otherwise, $s^1(v_1, v_2)$ must be 0. Similarly, we have $s^1(v_2, v_1) \in [0, 1]$, and thus $\mathcal{S}^1(v_1, v_2) \in [0, 1]$. Since the similarity $\mathcal{S}^n(v_1, v_2)$ will be updated only if both v_1 and v_2 are real events, we have $\mathcal{S}^0(v_1, v_2) = 0$ and $\mathcal{S}^1(v_1, v_2) \in [0, 1]$, that is, $0 \leq \mathcal{S}^0(v_1, v_2) \leq \mathcal{S}^1(v_1, v_2) \leq 1$. Therefore, the monotonicity holds for $n = 1$.

Induction: Assume that the monotonicity holds for $n = k$, i.e., $0 \leq \mathcal{S}^{k-1}(v_1, v_2) \leq \mathcal{S}^k(v_1, v_2) \leq 1$. According to formula (1), we have $s^{k+1}(v_1, v_2) = \frac{1}{|\bullet v_1|} \sum_{v_1' \in \bullet v_1} \max_{v_2' \in \bullet v_2} C(v_1, v_1', v_2, v_2') \times \mathcal{S}^k(v_1', v_2') \geq \frac{1}{|\bullet v_1|} \sum_{v_1' \in \bullet v_1} \max_{v_2' \in \bullet v_2} C(v_1, v_1', v_2, v_2') \times \mathcal{S}^{k-1}(v_1', v_2') = s^k(v_1, v_2)$. It thus follows $\mathcal{S}^{k+1}(v_1, v_2) \geq \mathcal{S}^k(v_1, v_2)$, and implies that $0 \leq \mathcal{S}^{n-1}(v_1, v_2) \leq \mathcal{S}^n(v_1, v_2) \leq 1$ also holds for $n = k + 1$.

As both basis and induction have been performed, by mathematical induction, the monotonicity holds for all $n \geq 1$.

Therefore, for any event pair (v_1, v_2), the sequence $\mathcal{S}^n(v_1, v_2)$ is bounded and non-decreasing along with the increase of n. According to the Completeness Axiom of calculus, each sequence $\mathcal{S}^n(v_1, v_2)$ has a limit $\mathcal{S}(v_1, v_2) \in [0, 1]$.

Uniqueness. We prove the uniqueness of the limit of $\mathcal{S}(v_1, v_2)$ by contradiction. Suppose that there are two distinct limits of forward similarity, denoted by $\mathcal{S}_1$ and $\mathcal{S}_2$. For each event pair (v_1, v_2), let $\Delta(v_1, v_2) = \mathcal{S}_1(v_1, v_2) - \mathcal{S}_2(v_1, v_2)$ be the difference of two limits, and $\Delta_M = \max_{v_1 \in V_1, v_2 \in V_2} |\Delta(v_1, v_2)|$ be the maximum absolute value of any $\Delta(v_1, v_2)$. For artificial events v_1^X and v_2^X, we have $\Delta(v_1^X, v_2) = \Delta(v_1, v_2^X) = 0$ because their similarity is never updated during an iteration. Otherwise, for two real events v_1 and v_2, we have:

$$
\begin{aligned}
\Delta_M =& \mathcal{S}_1(v_1, v_2) - \mathcal{S}_2(v_1, v_2) \\
=& \frac{\alpha}{2}\Big(\frac{1}{|\bullet v_1|} \sum_{v_1' \in \bullet v_1} \big(\max_{v_2' \in \bullet v_2} C(v_1, v_1', v_2, v_2')\mathcal{S}_1(v_1', v_2') \\
& - \max_{v_2' \in \bullet v_2} C(v_1, v_1', v_2, v_2')\mathcal{S}_2(v_1', v_2')\big) \\
& + \frac{1}{|\bullet v_2|} \sum_{v_2' \in \bullet v_2} \big(\max_{v_1' \in \bullet v_1} C(v_1, v_1', v_2, v_2')\mathcal{S}_1(v_1', v_2') \\
& - \max_{v_1' \in \bullet v_1} C(v_1, v_1', v_2, v_2')\mathcal{S}_2(v_1', v_2')\big)\Big) \\
\leq& \frac{\alpha}{2}\Big(\frac{1}{|\bullet v_1|} \sum_{v_1' \in \bullet v_1} \big(\max_{v_2' \in \bullet v_2} (C(v_1, v_1', v_2, v_2')\mathcal{S}_1(v_1', v_2') \\
& - C(v_1, v_1', v_2, v_2')\mathcal{S}_2(v_1', v_2'))\big) \\
& + \frac{1}{|\bullet v_2|} \sum_{v_2' \in \bullet v_2} \big(\max_{v_1' \in \bullet v_1} (C(v_1, v_1', v_2, v_2')\mathcal{S}_1(v_1', v_2') \\
& - C(v_1, v_1', v_2, v_2')\mathcal{S}_2(v_1', v_2'))\big)\Big) \\
\leq& \frac{\alpha}{2} c\Big(\frac{1}{|\bullet v_1|}|\bullet v_1|\Delta_M + \frac{1}{|\bullet v_2|}|\bullet v_2|\Delta_M\Big) = c\alpha\Delta_M
\end{aligned}
$$

Note that, $\Delta_M \leq c\alpha\Delta_M$ and Δ_M is an absolute number. If it holds $\Delta_M > 0$, we have $1 \leq c\alpha$, which is in contradiction with $0 \leq c\alpha \leq 1$. It concludes $\Delta_M = 0$, and thus, $\mathcal{S}_1(v_1, v_2) = \mathcal{S}_2(v_1, v_2)$.[2] □

3.4 Pruning

Next, we identify that the similarities of some node pairs are guaranteed to converge in a certain number (say h) of iterations. These similarities do not need to be updated in the n^{th} iterations, where $n > h$.

Intuitively, for a node v_1 (often the source node) with only one predecessor, artificial v_1^X, we can show that the similarity of v_1 to any node will be fixed (converged) after one iteration according to the similarity definition. Consequently, in the next iteration, a node whose predecessors are all converged (like v_1) is guaranteed to converge too.

Let $l(v)$ denote the longest distance from v^X to v (could be ∞ if loops exist from v^X to v).

Proposition 2. *For any two events* $v_1 \in V_1$ *and* $v_2 \in V_2$, $\mathcal{S}^{h+1}(v_1, v_2) - \mathcal{S}^h(v_1, v_2) = 0$, *for all* $h \geq \min(l(v_1), l(v_2))$.

Proof. This proposition can be proved by induction.

Basis: Obviously, when $\min(l(v_1), l(v_2)) = 1$, $\mathcal{S}^h(v_1, v_2)$ will never change after the first iteration.

Induction: Assume that the proposition holds for all v_1 and v_2 satisfying $\min(l(v_1), l(v_2)) = k$. For all v_1 and v_2 satisfying $\min(l(v_1), l(v_2)) = k + 1$, the similarity $\mathcal{S}^h(v_1', v_2')$ of their input neighbors will never change after the k^{th} iteration due to the assumption. Therefore, $\mathcal{S}^h(v_1, v_2)$ will never change after the $(k + 1)^{th}$ iteration.

By mathematical induction, the proposition holds for all $\min(l(v_1), l(v_2)) \geq 1$. □

Based on this proposition, we first compute $l(v_1)$ and $l(v_2)$ for all the events, so that there is no need to compute those similarities for event pairs (v_1, v_2) with $\min(l(v_1), l(v_2)) < k$

[2]Due to the limit of space, we include the complete proof in http://ise.thss.tsinghua.edu.cn/sxsong/doc/ematch.pdf

at the k^{th} iteration. In addition, we can say for sure that the iterative computation will stop after the n^{th} iteration, where $n = \min(\max_{v_1 \in V_1} l(v_1), \max_{v_2 \in V_2} l(v_2))$. It is worth noting that if G_1 and G_2 have loops, the early convergence property still holds for those node pairs (v_1, v_2) where all the ancestors of v_1 and v_2 are not involved in loops.

Example 5. *Considering the event pair $(A, 1)$ where $h = \min(l(A), l(1)) = 1$. According to formula (1), the value of $\mathcal{S}^n(A, 1)$ will only be affected by the value of $\mathcal{S}^{n-1}(v_1^X, v_2^X)$. Since $\mathcal{S}^n(v_1^X, v_2^X)$ never changes, the value of $\mathcal{S}^n(A, 1)$ will not change after the 1^{st} iteration. Likewise, the value of $\mathcal{S}^n(C, 2)$ will not change after the 2^{nd} iteration, since $\mathcal{S}(A, 1)$ and $\mathcal{S}(A, 2)$ converged at the 1^{st} iteration. Therefore, we prune the computation of $\mathcal{S}^n(A, 1)$ and $\mathcal{S}^n(B, 1)$ where $n \geq 1$, the computation of $\mathcal{S}^n(C, 2)$ and $\mathcal{S}^n(C, 3)$ where $n \geq 2$ and the computation of $\mathcal{S}^n(D, 4)$ where $n \geq 3$.*

3.5 Estimation

We further improve the efficiency by introducing an estimation of each $\mathcal{S}(v_1, v_2)$ with fewer iterations, e.g., even with only one iteration. Thereby, the estimation has an $O(|V_1||V_2|)$ time complexity in an extreme case that only one iteration is conducted, or conduct more iterations to make the estimated similarity closer to the exact similarity, which can be interpreted as trading the accuracy for efficiency.

First, we rewrite the formula of $\mathcal{S}^n(v_1, v_2)$ as follows:

$$\begin{aligned}\mathcal{S}^n(v_1, v_2) =& \frac{\alpha}{2}\Big[\frac{1}{|\bullet v_1|}\Big(C(v_1^X, v_1, v_2^X, v_2)\mathcal{S}(v_1^X, v_2^X) + \\ & \sum_{v_1' \in \{\bullet v_1\} \setminus \{v_1^X\}} \max_{v_2' \in \bullet v_2} C(v_1, v_1', v_2, v_2')\mathcal{S}^{n-1}(v_1', v_2')\Big) \\ & + \frac{1}{|\bullet v_2|}\Big(C(v_1^X, v_1, v_2^X, v_2)\mathcal{S}(v_1^X, v_2^X) + \\ & \sum_{v_2' \in \{\bullet v_2\} \setminus \{v_2^X\}} \max_{v_1' \in \bullet v_1} C(v_1, v_1', v_2, v_2')\mathcal{S}^{n-1}(v_1', v_2')\Big)\Big].\end{aligned}$$

In the formula above, we assume that each $C(v_1, v_1', v_2, v_2')$ achieves the maximum value c, and we replace the similarity of v_1' and v_2' by the similarity of v_1 and v_2. For simplicity, we denote C as $C(v_1, v_1', v_2, v_2')$, A as $|\bullet v_1|$, and B as $|\bullet v_2|$. The formula is further derived.

$$\begin{aligned}\mathcal{S}^n(v_1, v_2) \approx \mathcal{S}^n_{\text{es}}(v_1, v_2) =& \frac{\alpha c(2AB - A - B)}{2AB}\mathcal{S}^{n-1}_{\text{es}}(v_1, v_2) \\ & + \frac{\alpha(A + B)}{2AB}C + (1 - \alpha)\mathcal{S}^L(v_1, v_2).\end{aligned}$$

Let $q = \frac{\alpha c(2AB-A-B)}{2AB}$ and $a = \frac{\alpha(A+B)}{2AB}C + (1-\alpha)\mathcal{S}^L(v_1, v_2)$. It follows

$$\begin{aligned}\mathcal{S}^n_{\text{es}}(v_1, v_2) =& q\mathcal{S}^{n-1}_{\text{es}}(v_1, v_2) + a \\ q\mathcal{S}^{n-1}_{\text{es}}(v_1, v_2) =& q^2\mathcal{S}^{n-2}_{\text{es}}(v_1, v_2) + aq \\ & \vdots \\ q^{n-I-1}\mathcal{S}^{I+1}_{\text{es}}(v_1, v_2) =& q^{n-I}\mathcal{S}^I_{\text{es}}(v_1, v_2) + aq^{n-I-1},\end{aligned}$$

where $0 \leq I \leq n-1, I \in \mathbb{N}$. By eliminating the corresponding items on the left and the right sides, it implies

$$\mathcal{S}^n_{\text{es}}(v_1, v_2) = q^{n-I}\mathcal{S}^I_{\text{es}}(v_1, v_2) + a(1 + q + q^2 + \cdots + q^{n-I-1}).$$

By summing the geometric sequence, $\mathcal{S}^n_{\text{es}}(v_1, v_2)$ is given by

$$\mathcal{S}^n_{\text{es}}(v_1, v_2) = q^{n-I}\mathcal{S}^I_{\text{es}}(v_1, v_2) + \frac{a(1 - q^{n-I})}{1 - q}. \quad (2)$$

According to the early convergence proposed in Section 3.4, n should not be greater than $h = \min(l(v_1), l(v_2))$ (or n could be ∞, if $l(v_1)$ or $l(v_2)$ is ∞). Thereby, the estimation of $\mathcal{S}(v_1, v_2)$ is $\mathcal{S}^h_{\text{es}}(v_1, v_2)$. Noting that I is a constant number of iterations of exact computation before estimation, $\mathcal{S}^I_{\text{es}}(v_1, v_2)$ can be replaced by the exact value $\mathcal{S}^I(v_1, v_2)$. It provides a trade-off between accuracy and time. The larger I is, the closer the estimation values and the exact values are, which costs more time to do the iterations (see the experiments in Section 5 for the effect of varying I). In addition, I should be no greater than h according to early convergence.

Example 6. *Referring to the estimation formula, given $I = 0$ and $\alpha = 1$. The value of $S(A, 1)$ can be estimated by $\mathcal{S}^1_{\text{es}}(A, 1) = C(v_1^X, A, v_2^X, 1)c = 0.6$, which is equal to the exact value of $\mathcal{S}(A, 1)$. However, for the event pair $(C, 4)$, the estimated value is 0.409 while the exact value is 0.587. This is because the estimation formula treats the similarity of event C and 4 as the similarity of their ancestors. However, if we set $I = 10$, the estimated similarity of event pair $(C, 4)$ is 0.557, which is closer to the exact value.*

3.6 Algorithm

Finally, Algorithm 1 summarizes the procedure of iterative computation and estimation of forward similarity. All the similarities are initialized in Line 1. From Line 2 to Line 5, we first conduct I rounds of iterations according to formula (1). We consider the early convergence in Line 4, by skipping the iterative computations for those similarities that are early converged. Then, the estimation is conducted from Line 6 to Line 8 based on formula (2). Algorithm 1 has the lowest time complexity $O(|V_1||V_2|)$ when $I = 0$, and is equivalent to the exact algorithm if we set $I > \min(l(v_1), l(v_2))$, for all $v_1 \in V_1$ and $v_2 \in V_2$.

Algorithm 1 ExactEstimationTradeOff(G_1, G_2, I)

Input: two dependency graphs G_1 and G_2, and a constant I of iterations
Output: the 1:1 matching similarity $\mathcal{S}$

1: $\mathcal{S}^0(v_1, v_2) := 1$ for v_1^X and v_2^X, otherwise $\mathcal{S}^0(v_1, v_2) := 0$
2: **for** each i from 1 to I **do**
3: **for** each $v_1 \in V_1 \setminus \{v_1^X\}$,$v_2 \in V_2 \setminus \{v_2^X\}$ **do**
4: **if** $i \leq \min(l(v_1), l(v_2))$ and $\mathcal{S}^i(v_1, v_2)$ is not converged **then**
5: $\mathcal{S}^i(v_1, v_2) := \alpha(s^{i-1}(v_1, v_2) + s^{i-1}(v_2, v_1))/2 + (1 - \alpha)\mathcal{S}^L(v_1, v_2)$
6: **for** each $v_1 \in V_1 \setminus \{v_1^X\}$,$v_2 \in V_2 \setminus \{v_2^X\}$ **do**
7: **if** $I < h = \min(l(v_1), l(v_2))$ **then**
8: $\mathcal{S}_{\text{es}}(v_1, v_2) := q^{h-I}\mathcal{S}^I(v_1, v_2) + \frac{a(1-q^{h-I})}{1-q}$
9: **return** $\mathcal{S}_{\text{es}}$

Similarities in two directions

The similarity function in Definition 2 considers only the predecessors (in-neighbors, e.g., events A and B of event C in Figure 1), known as *forward similarity*. Symmetrically, we may also investigate successors (out-neighbors, e.g., events 2 and 3 are out-neighbors of event 1), namely *backward simi-*

larity. As shown in the experiments, by aggregating the forward and backward similarities together (e.g., by average), we can successfully address the matching with dislocations.

4. MATCHING COMPOSITE EVENTS

As mentioned earlier, an event in one source could possibly correspond to multiple events in another source, known as composite events [6]. For the set V of events appearing in an event log $\mathcal{L}$, let $U \subseteq V$ denote a composite event that consists of a subset of singleton events from V. We consider a collection of composite event candidates $\mathcal{U} = \{U \subseteq V\}$, where different candidates may overlap.[3] We call $\mathcal{W} \subseteq \mathcal{U}$ a *subset of non-overlapping composite events*, i.e., $\forall U_1, U_2 \in \mathcal{W}, U_1 \cap U_2 = \emptyset$.

Let $\mathcal{W}_1, \mathcal{W}_2$ be two sets of composites from the candidate sets $\mathcal{U}_1$,$\mathcal{U}_2$ for $\mathcal{L}_1$ and $\mathcal{L}_2$, respectively. To evaluate the similarities with composite events, i.e., $\mathcal{S}(\mathcal{W}_1, \mathcal{W}_2)$, we can simply treat each composite event as one node (in constructing the dependency graph) and thus directly apply the similarity function in Definition 2.

It is easy to see the large number of possible combinations of non-overlapping composite events in each event log. Different combinations may yield distinct similarity results. Intuitively, the higher the similarities in $\mathcal{S}(\mathcal{W}_1, \mathcal{W}_2)$, the better the composite events are matched. We denote $avg(\mathcal{S}(\mathcal{W}_1, \mathcal{W}_2)) = \frac{\sum_{W_1 \in \mathcal{W}_1, W_2 \in \mathcal{W}_2} \mathcal{S}(W_1, W_2)}{|\mathcal{W}_1| \cdot |\mathcal{W}_2|}$ the average of all the pair-wise similarities between composite events in $\mathcal{W}_1, \mathcal{W}_2$.

Problem 1 (Composite event matching)**.** *Given two sets of candidate composite events $\mathcal{U}_1$ and $\mathcal{U}_2$ for two event logs $\mathcal{L}_1$ and $\mathcal{L}_2$, respectively, the optimal composite event matching problem is to find two subsets of non-overlapping composite events $\mathcal{W}_1 \subseteq \mathcal{U}_1$ and $\mathcal{W}_2 \subseteq \mathcal{U}_2$ such that the average composite event similarity $avg(\mathcal{S}(\mathcal{W}_1, \mathcal{W}_2))$ is maximized.*

Unfortunately, the problem is highly non-trivial.

Theorem 3. *The problem of finding the optimal composite event matching is* NP*-hard.*

Proof idea. To show the NP-hardness, we build a reduction from the maximum set packing problem, which is one of Karp's 21 NP-complete problems [12]. Given a universe V and a family $\mathcal{U}$ of subsets of V, a packing is a subfamily $\mathcal{W} \subseteq \mathcal{U}$ of sets such that all sets in $\mathcal{W}$ are pairwise disjoint. The maximum set packing problem asks for the maximum weighted aggregation of pairwise disjoint sets, which is analogous to a set of composite events with the maximum similarity in the matching problem. □

Hence, in the following, we focus on greedy heuristics that iteratively select one composite event in a step to form the results. Note that in each step of selecting one composite event, similarities of some node pairs may not be affected and thus have no need to be recomputed. Moreover, to enable pruning among possible combinations, we devise the upper bounds of similarities.

4.1 Heuristics

Consider all the remaining candidate composite events that can be further selected $\mathcal{C}_1 = \{U \in \mathcal{U}_1 \mid U \cap W = \emptyset, W \in \mathcal{W}_1\}$, i.e., those U not overlapping with any composite event W in $\mathcal{W}_1$. As mentioned above, for a $U \in \mathcal{C}_1$, we can directly obtain the dependency graph w.r.t. $\mathcal{W}_1 \cup \{U\}$ by updating the graph of $\mathcal{W}_1$.

In each step, we greedily select a composite event $U_1 \in \mathcal{C}_1$ of $\mathcal{L}_1$ that can maximize the average similarity, i.e.,

$$U_1 = \arg\max_{U \in \mathcal{C}_1} avg(\mathcal{S}(\mathcal{W}_1 \cup \{U\}, \mathcal{W}_2)) - avg(\mathcal{S}(\mathcal{W}_1, \mathcal{W}_2)).$$

Similarly, we obtain a $U_2 \in \mathcal{C}_2$ of $\mathcal{L}_2$. The one, either $U_1 \in \mathcal{C}_1$ or $U_2 \in \mathcal{C}_2$, with higher average similarity improvement will be confirmed to add to the corresponding $\mathcal{W}_1$ or $\mathcal{W}_2$.

Since the similarity improvement could be negative, a threshold δ of minimum similarity improvement can be specified, e.g. $\delta = 0$ with the requirement of similarity increase $avg(\mathcal{S}(\mathcal{W}_1 \cup \{U_1\}, \mathcal{W}_2)) - avg(\mathcal{S}(\mathcal{W}_1, \mathcal{W}_2)) > 0$ or $avg(\mathcal{S}(\mathcal{W}_1, \mathcal{W}_2) \cup \{U_2\}) - avg(\mathcal{S}(\mathcal{W}_1, \mathcal{W}_2)) > 0$.

Example 7. *Suppose that composite event candidates are $\mathcal{U}_1 = \{\{A\}, \{B\}, \{C\}, \{D\}, \{E\}, \{F\}, \{C, D\}, \{E, F\}\}$ for $\mathcal{L}_1$ on the dependency graph shown in Figure 1(c) and δ is* 0.005*. We first compute the pairwise similarities of singleton events of G_1 and G_2, $\mathcal{S} = \mathcal{S}(\{A, B, C, D, E, F\}, \{1, 2, 3, 4, 5, 6\})$, having $avg(\mathcal{S}) =$ 0.502 (let $\alpha = 1$). Then, we construct two new dependency graphs $G_1^{\{C,D\}}$ and $G_1^{\{E,F\}}$ by merging events CD and EF in G_1, respectively. The pairwise event similarities of each candidate $\mathcal{S}^{\{C,D\}} = \mathcal{S}(\{A, B, CD, E, F\}, \{1, 2, 3, 4, 5, 6\})$ and $\mathcal{S}^{\{E,F\}} = \mathcal{S}(\{A, B, C, D, EF\}, \{1, 2, 3, 4, 5, 6\})$ are computed, with average similarities $avg(\mathcal{S}^{\{C,D\}}) =$ 0.508 and $avg(\mathcal{S}^{\{E,F\}}) =$ 0.478. The composite event candidate $\{C, D\}$ with higher $avg(\mathcal{S}^{\{C,D\}}) - avg(\mathcal{S}) > \delta$ is accepted. Since $\{C, D\}$ and $\{E, F\}$ do not overlap, we further test merging both CD and EF in G_1 by computing $\mathcal{S}^{\{C,D\},\{E,F\}} = \mathcal{S}(\{A, B, CD, EF\}, \{1, 2, 3, 4, 5, 6\})$. However, the average similarity $avg(\mathcal{S}^{\{C,D\},\{E,F\}}) = 0.478 < avg(\mathcal{S}^{\{C,D\}})$ decreases. $\{C, D\}$ is returned in composite event matching.*

4.2 Identifying Unchanged Similarities

Since introducing a new composite event U changes the structure of the dependency graph, the similarity of existing events (not involved in U) may change. A straightforward idea is to recompute all the similarities from scratch. We note, however, if there does not exist a path from U to v, the similarities on v will not change in this iteration.

Let U be the currently selected composite event in a greedy step. We denote $AN(v)$ all the ancestors of a node v w.r.t. prerequisites.

Proposition 4. *If $AN(v) \cap U = \emptyset$, all the similarities $\mathcal{S}(v, u)$ will be unchanged after integrating U.*

Proof. According to formula (1), the similarity of $\mathcal{S}(v, u)$ will only be affected by the similarities of $AN(v)$ and $AN(u)$. Obviously, if $AN(v) \cap U = \emptyset$, the pairwise similarities of the ancestors of v and u remain unchanged, thus $\mathcal{S}(v, u)$ will not change after integrating U. □

Therefore, we only need to update the similarities whose ancestors are changed w.r.t. the composite event U.

4.3 Pruning via Similarity Upper Bounds

Although a greedy heuristic is employed, it is still costly in each greedy step to compute the similarity for selecting each possible composite event candidate. In the following, we show that we can compute an upper bound of similarity for any pair of events between two dependency graphs. Such an upper bound enables pruning of composite event candidates,

[3]Candidates of composite events can either be manually identified or automatically discovered [6].

whose similarity upper bound is lower than some currently computed results.

The idea is that the similarity is not only increasing (Theorem 1) between two iterations, but also increases within a certain bound w.r.t. the iteration number. For those event pairs that converge in a constant number of iterations (Proposition 2), we can develop a certain bound of similarity. A general upper bound for any event pair is also derived.

Lemma 5. *Let $v_1 \in V_1$ and $v_2 \in V_2$ be two events, c the upper bound of function C. The difference of similarity between the n^{th} and $(n-1)^{th}$ iterations ($n \geq 1$) holds:*

$$0 \leq \mathcal{S}^n(v_1, v_2) - \mathcal{S}^{n-1}(v_1, v_2) \leq (\alpha c)^n$$

Proof. $0 \leq \mathcal{S}^n(v_1, v_2) - \mathcal{S}^{n-1}(v_1, v_2)$ has been proved by the monotonicity, and we only need to prove $\mathcal{S}^n(v_1, v_2) - \mathcal{S}^{n-1}(v_1, v_2) \leq c^n$. It can be proved by induction.
Basis: Let $n = 1$. $\mathcal{S}^1(v_1, v_2) - \mathcal{S}^0(v_1, v_2)$

$$\begin{aligned}=&\frac{\alpha}{2}\Big(\frac{1}{|\bullet v_1|}\sum_{v_1' \in \bullet v_1}\max_{v_2' \in \bullet v_2} C(v_1', v_1, v_2', v_2)\mathcal{S}^0(v_1', v_2')\\ &+\frac{1}{|\bullet v_2|}\sum_{v_2' \in \bullet v_2}\max_{v_1' \in \bullet v_1} C(v_1', v_1, v_2', v_2)\mathcal{S}^0(v_1', v_2')\Big).\end{aligned}$$

Since $0 \leq \mathcal{S}^0(v_1', v_2') \leq 1$, we have

$$\mathcal{S}^1(v_1, v_2) - \mathcal{S}^0(v_1, v_2) \leq \frac{\alpha}{2}\Big(\frac{1}{|\bullet v_1|}|\bullet v_1|c + \frac{1}{|\bullet v_2|}|\bullet v_2|c\Big) = \alpha c.$$

The lemma holds for $n = 1$.
Induction: Assume that the lemma holds for $n = k$, i.e., $\mathcal{S}^k(v_1, v_2) - \mathcal{S}^{k-1}(v_1, v_2) \leq (\alpha c)^k$. When $n = k + 1$, according to Definition 2, we have

$$\begin{aligned}&\mathcal{S}^{k+1}(v_1, v_2) - \mathcal{S}^k(v_1, v_2)\\ \leq&\frac{\alpha}{2}\Big(\frac{c}{|\bullet v_1|}\sum_{v_1' \in \bullet v_1}\max_{v_2' \in \bullet v_2}(\mathcal{S}^k(v_1', v_2') - \mathcal{S}^{k-1}(v_1', v_2'))\\ &+\frac{c}{|\bullet v_2|}\sum_{v_2' \in \bullet v_2}\max_{v_1' \in \bullet v_1}(\mathcal{S}^k(v_1', v_2') - \mathcal{S}^{k-1}(v_1', v_2'))\Big)\\ \leq&\frac{\alpha}{2}\Big(\frac{c}{|\bullet v_1|}|\bullet v_1|(\alpha c)^k + \frac{c}{|\bullet v_2|}|\bullet v_2|(\alpha c)^k\Big) = (\alpha c)^{k+1}.\end{aligned}$$

Thus, the conclusion also holds for $n = k + 1$.

By mathematical induction, Lemma 5 holds for $n \geq 1$. □

We are now ready to compute an upper bound of similarity between any two events.

Proposition 6. *Let v_1, v_2 be two events, c be the upper bound of function C, and $\mathcal{S}^k(v_1, v_2)$ be the similarity after k^{th} iteration. The upper bound of similarity $\mathcal{S}(v_1, v_2)$ is:*

$$\mathcal{S}_U(v_1, v_2) = \mathcal{S}^k(v_1, v_2) + \frac{(\alpha c)^k}{1 - \alpha c}.$$

Proof. According to Lemma 5, we have

$$\begin{aligned}\mathcal{S}^n(v_1, v_2) =&\mathcal{S}^k(v_1, v_2) + (\mathcal{S}^{k+1}(v_1, v_2) - \mathcal{S}^k(v_1, v_2))\\ &+ (\mathcal{S}^{k+2}(v_1, v_2) - \mathcal{S}^{k+1}(v_1, v_2)) + \cdots\\ &+ (\mathcal{S}^n(v_1, v_2) - \mathcal{S}^{n-1}(v_1, v_2))\\ \leq&\mathcal{S}^k(v_1, v_2) + (\alpha c)^{k+1} + (\alpha c)^{k+2} + \cdots + (\alpha c)^n\\ =&\mathcal{S}^k(v_1, v_2) + \frac{(\alpha c)^k - (\alpha c)^n}{1 - \alpha c}.\end{aligned}$$

When $n \to \infty$, $(\alpha c)^n \to 0$, we have $\mathcal{S}(v_1, v_2) \leq \mathcal{S}^k(v_1, v_2) + \frac{(\alpha c)^k}{1-\alpha c}$. □

According to Proposition 2, if there exists a longest path with finite length h from the artificial node v^X to v, the similarity on v will always converge in h steps. Rather than the general bound in Proposition 6, we can develop an upper bound of such events as follows.

Corollary 7. *For any two events $v_1 \in V_1$ and $v_2 \in V_2$, let $h \geq \min(l(v_1), l(v_2))$. The upper bound of similarity $\mathcal{S}(v_1, v_2)$ is:*

$$\mathcal{S}_U(v_1, v_2) = \mathcal{S}^k(v_1, v_2) + \frac{(\alpha c)^k - (\alpha c)^h}{1 - \alpha c}.$$

Finally, Algorithm 2 presents the procedure of greedy merge. At the beginning, the event similarity $\mathcal{S}$ of original dependency graph G_1 and G_2 is computed. From Line 2 to Line 12, we use the greedy strategy to determine the best solution of node aggregation. We first gain all the possible candidates $\mathcal{U}$. For each candidate, we reconstruct the dependency graphs by merging events (or composite events). Then, we compute $\mathcal{S}'$ for reconstructed G_1' and G_2' in Line 6. It is worth noting that the pruning methods proposed in Sections 4.2 and 4.3 can be utilized in this step. After all the candidates are processed, we compare the maximum average similarity $avg(\mathcal{S}^{max})$ among all the $\mathcal{S}'$ in this iteration with the maximum average similarity $avg(\mathcal{S})$ of last iteration. If the increment is higher than a predefined threshold δ, the candidate that produces $\mathcal{S}^{max}$ is confirmed as a proper merging candidate, and the iteration will go on. Otherwise, we return $\mathcal{S}$ as the best similarity since the average similarity cannot be improved significantly.

Algorithm 2 CompositePrune(G_1, G_2)

Input: two dependency graph G_1 and G_2
Output: the m:n matching similarity $\mathcal{S}$
1: $\mathcal{S}$:= compute event similarity of G_1 and G_2
2: **for** i from 1 to infinite **do**
3: $\mathcal{U}$:=all possible candidate pairs
4: **for** each candidate $\mathcal{W}$ in $\mathcal{U}$ **do**
5: G_1', G_2' := reconstructed G_1 and G_2 by merging events
6: $\mathcal{S}'$:= compute event similarity of G_1' and G_2'
7: **if** $avg(\mathcal{S}') > avg(\mathcal{S}^{max})$ **then**
8: $\mathcal{S}^{max}$:=$\mathcal{S}'$,$G_1^{max} := G_1'$,$G_2^{max} := G_2'$
9: **if** $avg(\mathcal{S}^{max}) - avg(\mathcal{S}) < \delta$ **then**
10: break
11: **else**
12: $\mathcal{S} := \mathcal{S}^{max}$,$G_1 := G_1^{max}$,$G_2 := G_2^{max}$
13: **return** $\mathcal{S}$

Example 8 (Example 7 continued)**.** *Suppose that we now consider the composite event candidate $U = \{E, F\}$ in Figure 1 (c). For those events in V_1, we have $AN(A) \cap U = AN(B) \cap U = AN(C) \cap U = AN(D) \cap U = \emptyset$. Hence, for event $v_1 \in \{A, B, C, D\}$, we have $\mathcal{S}^{\{C,D\}}(v_1, v_2) = \mathcal{S}(v_1, v_2)$ according to Proposition 4.*

Now suppose that we are computing $\mathcal{S}^{\{E,F\}}$ after $avg(\mathcal{S}^{\{C,D\}}) = 0.508$ is gained. When the 15^{th} iteration is done, the upper bound of average similarity $avg(\mathcal{S}_U^{\{E,F\}})$ is lower than 0.504, which satisfies $avg(\mathcal{S}_U^{\{E,F\}}) < avg(\mathcal{S}^{\{E,F\}})$. Therefore, we can immediately stop the computation of $\mathcal{S}^{\{E,F\}}$.

5. EVALUATION

In this section, we report an experimental evaluation on comparing our method with the state-of-the-art event matching approaches, *graph edit distance* (GED) [5], *opaque name matching* (OPQ) [11] and *behavioral similarity* (BHV) [19].

5.1 Implementation and Settings

Data sets. We employ a real data set of 149 event log pairs, which are extracted from 10 different functional areas in the OA systems of two subsidiaries of a bus manufacturer. Each event log pair denotes two event logs doing the same or similar works in two subsidiaries, respectively. The matching relationships in event log pairs are manually identified.

The data set is divided into two groups, where the first group with 103 event log pairs does not contain composite events, while the remaining 46 event logs in the second group have composite events.

To study the performance on dislocations, we categorize the dataset into 3 testbeds w.r.t. matching positions. The first one, namely DS-F, consists of 23 event log pairs where the dislocated events appear at the end of traces between two logs. In the second testbed, namely DS-B with 22 event log pairs, those dislocated events locate in the beginning of traces between two logs. Finally, DS-FB may involve dislocated events at both the beginning and the end of traces.

A state-of-the-art string similarity measure, cosine similarity with q-grams [9], is employed to compute the label similarity. Candidates of composite events are obtained by grouping singleton events that always appear consecutively, following the convention of SEQ pattern in CEP [6].

Besides real data sets, we also generate a synthetic data set to evaluate the scalability of our approaches. Firstly, we generate 10 groups of random process specifications by varying event sizes ranging from 10 to 100. Each event size contains 20 distinct process specifications. For each process specification, we randomly generate 2 event logs, which form an event log pair. Therefore, we have 20 event log pairs on each distinct event size. Obviously, the events in two logs with the same name correspond to each other. All these models and logs are generated by an open source toolkit BeehiveZ[4] using existing generating approaches [18, 15].

Criteria. After pair-wise similarities of events are computed, we use the maximum total similarity selection method [17] to select event correspondences. The ground truth, i.e., the true mapping of events among 149 event log pairs, is supplied by 49 subject-matter experts in MIS (Management Information Systems) departments of each subsidiaries of the bus manufacturer during a long-period deliberation. Let found denote the matching correspondences produced by event matching approaches. We use the f-measure of precision and recall to evaluate the accuracy of event matching, given by $precision = \frac{|\mathsf{truth} \cap \mathsf{found}|}{|\mathsf{found}|}$, $recall = \frac{|\mathsf{truth} \cap \mathsf{found}|}{|\mathsf{truth}|}$, and $f\text{-}measure = 2 \cdot \frac{precision \cdot recall}{precision + recall}$. A larger f-measure indicates a higher matching accuracy.

Besides the accuracy performance, we also evaluate time costs of matching approaches.

[4]http://code.google.com/p/beehivez/

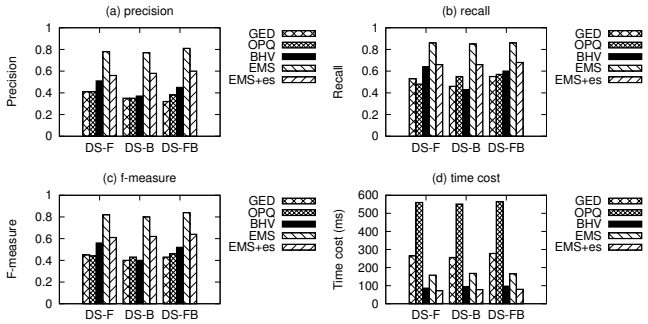


Figure 3: Performance on matching singleton events

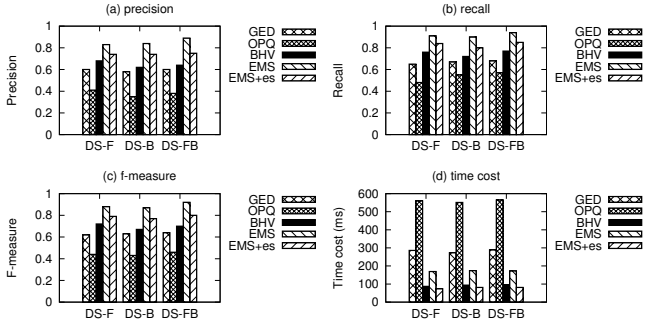


Figure 4: Integrating with typographic similarity

Our programs are implemented in Java and all experiments were performed on a PC with Intel(R) Core(TM) i7-2600 3.40GHz CPU and 8 GB memory.

5.2 Evaluating Event Similarity

We first report the experimental results on computing singleton event similarity. The compared approaches include our proposed event matching similarity (EMS) and the estimation EMS+es with $I = 5$, as well as existing approaches GED, OPQ and BHV.

Figure 3 presents the average accuracy and time costs of event matching. In order to observe the performance in the scenario of opaque events, we do not employ other similarities such as typographic similarities on event names, and rely on the structural similarity of dependency graphs only. First, the accuracy of our proposed EMS is higher than all the existing methods in all the testbeds. The rationale is that GED and OPQ concern local similarity, while dislocated events often have distinct neighbors and prevent these two approaches performing well as explained in Example 2. Moreover, BHV performs better than GED and OPQ on testbed DS-F, where the correspondences of events at the beginning of traces can be addressed by the forward similarity of BHV. However, BHV's accuracy is much lower on testbed DS-B compared with DS-F, since it only considers one-direction similarity and cannot handle well the dislocated events at the beginning of traces (in DS-B as well as DS-FB) as indicated in Example 1. Our EMS considers similarities in both directions (as indicated in Section 3.6) and employs the artificial event to reduce the impact of distinct neighbors of dislocated events. Consequently, EMS outperforms BHV on all the testbeds.

The corresponding time cost of EMS is no more than the double of BHV's and significantly lower than that of GED

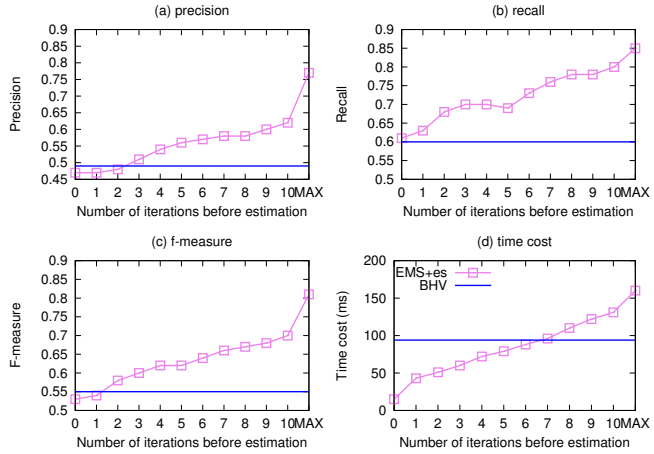


Figure 5: Trade-off in estimation

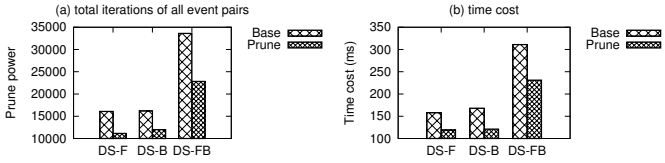


Figure 6: Prune power of early convergence

and OPQ. It is not surprising owing to the high complexity of computing graph edit distance or normal distance. Most importantly, the similarity estimation approach (EMS+es, with 5 iterations) shows the lowest time cost among all the evaluated approaches. Although the improvement in terms of time by EMS+es is not great compared with BHV, the accuracy of EMS+es outperforms BHV significantly (especially in DS-B and DS-FB), which is also observed in the following experiments such as Figure 4.

Figure 4 illustrates the results by integrating structural similarities with typographic similarities (cosine similarity of event names). In general, the results are very similar to Figure 3 without considering the typographic similarity. The major difference is that the accuracies of all approaches are improved except OPQ, since OPQ does not benefit from label similarity. The results demonstrate that our proposed similarity measure can improve the matching accuracy, no matter with or without typographic similarities.

Figure 5 evaluates the trade-off in performance by estimation. As shown in the figure, when the number of fixed iterations is small, e.g., 0, the estimation f-measure is comparable with the state-of-the-art BHV and shows about an order of magnitude improvement in time costs. On the other hand, for a large number of iterations, such as 10 or MAX (precise measure without estimation), the improvement of f-measure is up to 0.25, i.e., from 0.55 to 0.8.

Figure 6 evaluates the performance of the pruning method proposed in Section 3.4. Note that when pruning is applied, each pair of events may need a different number of iterations (each iteration performs a computation of formula (1)). Thereby, we observe the total number of iterations w.r.t. all event pairs, i.e., the total times of calculating formula (1). As illustrated in Figure 6(a), the total number of iterations for all event pairs can be significantly reduced by pruning, and the corresponding time costs are reduced in Figure 6(b).

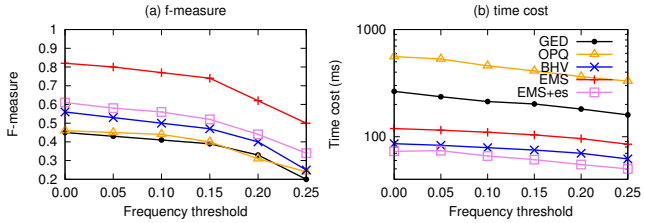


Figure 7: Trade-off by frequency filtering

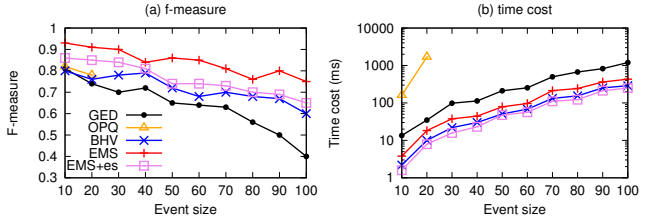


Figure 8: Scalability on the number of events

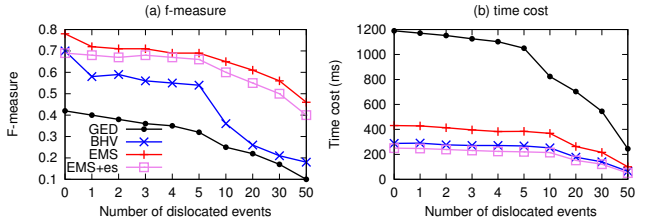


Figure 9: Performance on handling dislocated events

To evaluate the minimum frequency control proposed in Section 2, we gradually remove the edges with frequency lower than a threshold (varying from 0.05 to 0.25) from dependency graphs. According to the result reported in Figure 7(a), the accuracy decreases along with the increase of pruning ratio, due to the absence of more statistic and structural information. Nevertheless, the time cost also decreases as shown in Figure 7(b), since the average degree of dependency graph is decreasing, which makes the computation faster. Hence, we can utilize the minimum frequency control to trade the accuracy for computation efficiency.

Figure 8 reports the results of scalability on the number of events (up to 100 events).[5] As shown in Figure 8(a), the accuracy of all the approaches decreases along with the increase of event size. It is not surprising since more choices of events lead to a higher chance of mismatching. Remarkably, the falling speed of our EMS is slower than other approaches, which means the EMS method is more reliable in event logs with a large number of distinct events. The time cost of all approaches increases heavily in Figure 8(b). OPQ cannot even finish the matching of events more than 30, due to the highest time complexity $O(n!)$. Nevertheless, EMS+es always achieves the lowest time cost when the number of events is increasing from 10 to 100.

Figure 9 evaluates the performance over various sizes of dislocated events (in the synthetic dataset of 100 events). To

[5]Real event logs, however, often have the number of events bounded by about 60, according to the recent survey [22]. Indeed, referring to the process modeling guidelines [16], workflows should be decomposed if they have more than 50 events, so that they are easier to read and understand.

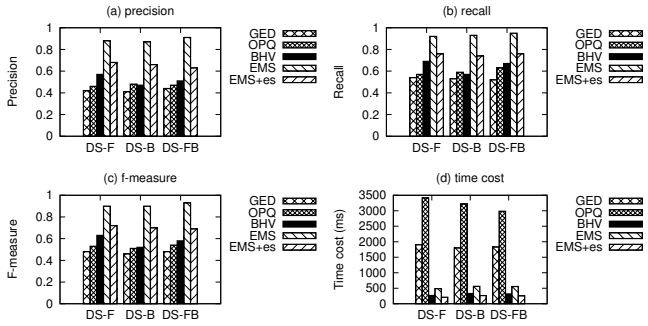


Figure 10: Matching composite events

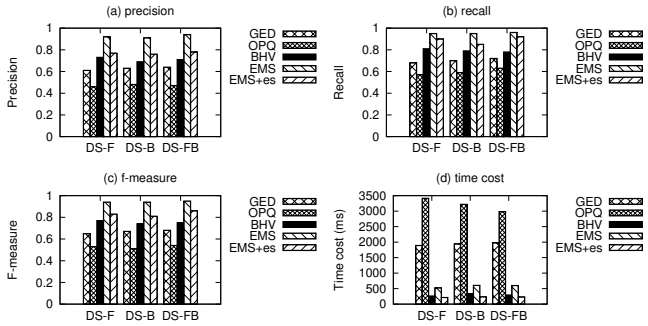


Figure 11: Performance on matching composite events by integrating with typographic similarity

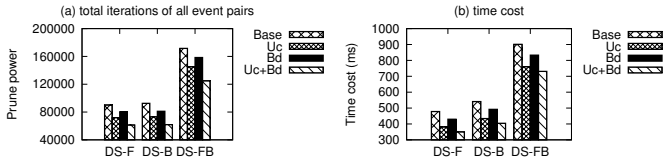


Figure 12: Prune power of unchanged similarities and upper bounds of similarities

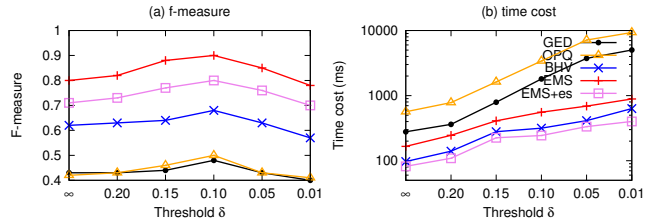


Figure 13: Performance on varying threshold δ

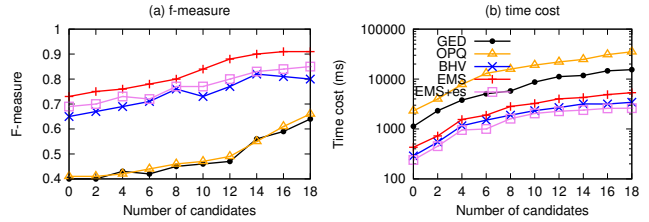


Figure 14: Performance on varying candidate sizes

simulate the different sizes of dislocated events presented in Example 1, we synthetically remove the first m events of each trace in one event log for every event log pair. By increasing m, i.e., the number of dislocated events, the accuracy of all the approaches drops. In particular, BHV's accuracy drops fast, with performance as poor as GED when the dislocated event size is large. Our proposed EMS shows the highest and relatively steady accuracy. These results verify again the superiority and demonstrate scalability of EMS in handling a larger number of dislocated events.

5.3 Matching Composite Events

In Figures 10 and 11, we report the results on matching composite events, without and with typographic similarity, respectively. In general, the results are very similar to Figures 3 and 4 of matching singleton events. That is, our similarity such as EMS can achieve higher accuracy, while the time cost of computation with estimation is much lower. A major difference observed in matching composite events is the significantly higher time costs of GED and OPQ. As mentioned in the introduction, we need to frequently compute the similarities of events for various combinations of candidate composite events, which is not affordable for similarity measures with high computational costs (such as GED and OPQ). Nevertheless, our proposed similarity measures can keep time costs low. Especially, EMS+es shows 1-2 orders of magnitude time cost improvement while the accuracy still outperforms GED.

Figure 12 reports the effectiveness of pruning methods, i.e., identifying unchanged similarities (Uc) in Section 4.2 and pruning with similarity upper bounds (Bd) in Section 4.3. Again, we first observe the total number of iterations (times of computing formula (1)) for all event pairs. As illustrated, both these approaches show an improvement of time performance, and the combination of two pruning methods can further reduce the time cost.

Figure 13 evaluates the effect of δ in Algorithm 2. According to Algorithm 2, a lower δ prefers more composite events. As illustrated in Figure 13(a), the f-measure first grows up along with the decrease of δ, since the algorithm does discover some true matching of composite events. However, the f-measure starts to drop when δ decreases lower than 0.10, since many false positive composite events are involved. That is, a moderately large threshold δ is preferred, in order to achieve a high matching accuracy. It is not surprising that the time cost significantly grows along with the decrease of δ in Figure 13(b), since Algorithm 2 needs to conduct the iterative computation of similarity for all the composite events discovered.

Finally, as shown in Figure 14, by considering more candidates, we can identify the matching of more composite events and thus increase the matching accuracy. The corresponding time costs, however, increase significantly fast.

6. RELATED WORK

A graph is often employed to represent the structural information among events. While vertices usually denote events, the edges in the graph are associated with various semantics exploited from event logs in different perspectives. Ferreira et al. [8] used a graphical form of Markov transition matrix whose edges are weighted by the conditional probability of one event directly followed by another. However, the conditional probability cannot tell the significance of the edge. In this paper, we employ the dependency graph proposed in [11] by weighting vertices and edges with nor-

malized frequencies, since it distinguishes the significance of distinct edges, and is easy to interpret. An important difference from [11] is the novel artificial node v^X introduced in the dependency graph for matching dislocated events.

Schema matching techniques [21], as a fundamental problem in many database application domains, can be employed to evaluate event similarities. Kang et al. [11] studied the matching on opaque data. However, as discussed in Example 2, OPQ concerns a local evaluation of similar neighbors, while dislocated matching events may have distinct neighbors which prevents OPQ performing well. In contrast, our proposed iterative similarity function concerns the global evaluation via propagating similarities and thus overcome the effect of local neighbor distinctness. Moreover, [11] needs to enumerate all the possible matching correspondences and select the one with the highest normal distance, which is extremely time-consuming. Consequently, the performance of [11] is poor in the experimental evaluation in Section 5.

A SimRank [10] like behavioral similarity [19] is employed by iteratively considering the predecessor similarities of two events. Unfortunately, this behavioral similarity fails to consider the distinct feature of dislocated events. Therefore, as illustrated in the experimental evaluation in Section 5, our proposed similarity measure with the consideration of dislocation shows higher matching accuracy. Weidlich et al. [23] studies matching composite events due to the different levels of granularity of business process [1]. However, it uses label similarity of events to judge m:n matching, which is non-effective on opaque event names. Another graph based similarity is graph edit distance [5] which falls short in matching dislocated events. Moreover, another defeat of graph edit distance is its high computation cost. When matching composite events with various combinations of candidates, the large matching time costs become a major issue, which is also observed in the experimental evaluation in Section 5.

7. CONCLUSIONS

In this paper, we first identify the unique features that often exist in heterogeneous event logs, such as **opaque, dislocated** and **composite** events. Since possibly opaque event names prevent most existing typographic or linguistic similarities from performing well, we focus on the structural information for matching. In particular, an iterative similarity function is introduced with the consideration of dislocation issues. We prove that the iterative computation of the proposed similarity function converges. Efficient pruning with the identification of early convergence is developed. We also propose a fast estimation of similarities with only a constant number (including 0) of iterations. To efficiently match composite events, we devise upper bounds of similarities for pruning. Experimental results demonstrate that our event similarity shows significantly higher accuracy than state-of-the-art matching approaches. Moreover, the similarity estimation can significantly reduce time costs while keeping matching accuracy higher/comparable with existing approaches. Since it is not clear about the difference of substituting $\mathcal{S}(v_1, v_2)$ and $\mathcal{S}(v_1', v_2')$, thus far, we do not get any theoretical bound of estimation. It is interesting to investigate the bound of estimation as a future study.

Acknowledgment. This work is supported in part by China NSFC under Grants 61325008, 61202008, 61370055, and National Grand Fundamental Research 973 Program of China under Grant 2012-CB316200.

8. REFERENCES

[1] Z. Bao, S. B. Davidson, and T. Milo. Labeling workflow views with fine-grained dependencies. *PVLDB*, 5(11):1208–1219, 2012.

[2] O. Biton, S. C. Boulakia, S. B. Davidson, and C. S. Hara. Querying and managing provenance through user views in scientific workflows. In *ICDE*, pages 1072–1081, 2008.

[3] F. Casati, M. Castellanos, U. Dayal, and N. Salazar. A generic solution for warehousing business process data. In *VLDB*, pages 1128–1137, 2007.

[4] F. Casati, M. Castellanos, N. Salazar, and U. Dayal. Abstract process data warehousing. In *ICDE*, pages 1387–1389, 2007.

[5] R. M. Dijkman, M. Dumas, and L. García-Bañuelos. Graph matching algorithms for business process model similarity search. In *BPM*, pages 48–63, 2009.

[6] L. Ding, S. Chen, E. A. Rundensteiner, J. Tatemura, W.-P. Hsiung, and K. S. Candan. Runtime semantic query optimization for event stream processing. In *ICDE*, pages 676–685, 2008.

[7] A. Doan, A. Halevy, and Z. Ives. *Principles of data integration*. Morgan Kaufmann, 2012.

[8] D. R. Ferreira, D. Gillblad, and D. Gillblad. Discovering process models from unlabelled event logs. In *BPM*, pages 143–158, 2009.

[9] L. Gravano, P. G. Ipeirotis, N. Koudas, and D. Srivastava. Text joins in an rdbms for web data integration. In *WWW*, pages 90–101, 2003.

[10] G. Jeh and J. Widom. Simrank: a measure of structural-context similarity. In *KDD*, pages 538–543, 2002.

[11] J. Kang and J. F. Naughton. On schema matching with opaque column names and data values. In *SIGMOD Conference*, pages 205–216, 2003.

[12] R. M. Karp. Reducibility among combinatorial problems. In R. E. Miller and J. W. Thatcher, editors, *Complexity of Computer Computations*, pages 85–103. Plenum Press, 1972.

[13] V. I. Levenshtein. Binary codes capable of correcting deletions, insertions, and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966.

[14] S. Melnik, H. Garcia-Molina, E. Rahm, and E. Rahm. Similarity flooding: A versatile graph matching algorithm and its application to schema matching. In *ICDE*, pages 117–128, 2002.

[15] J. Mendling, J. Recker, M. Rosemann, and W. M. P. van der Aalst. Generating correct epcs from configured c-epcs. In *SAC*, pages 1505–1510, 2006.

[16] J. Mendling, H. A. Reijers, and W. M. P. van der Aalst. Seven process modeling guidelines (7pmg). *Information & Software Technology*, 52(2):127–136, 2010.

[17] J. Munkres. Algorithms for the assignment and transportation problems. *Journal of the Society for Industrial & Applied Mathematics*, 5(1):32–38, 1957.

[18] J. Nakatumba, M. Westergaard, and W. M. P. van der Aalst. Generating event logs with workload-dependent speeds from simulation models. In *CAiSE Workshops*, pages 383–397, 2012.

[19] S. Nejati, M. Sabetzadeh, M. Chechik, S. M. Easterbrook, and P. Zave. Matching and merging of statecharts specifications. In *ICSE*, pages 54–64, 2007.

[20] T. Pedersen, S. Patwardhan, and J. Michelizzi. Wordnet: : Similarity - measuring the relatedness of concepts. In *AAAI*, pages 1024–1025, 2004.

[21] E. Rahm and P. A. Bernstein. A survey of approaches to automatic schema matching. *VLDB J.*, 10(4):334–350, 2001.

[22] J. Wang, T. Jin, R. Wong, and L. Wen. Querying business process model repositories - a survey of current approaches and issues. *World Wide Web*, 17(3):427–454, 2014.

[23] M. Weidlich, R. M. Dijkman, and J. Mendling. The icop framework: Identification of correspondences between process models. In *CAiSE*, pages 483–498, 2010.