Matching Heterogeneous Event Data

Yu Gao, Shaoxu Song, Xiaochen Zhu, Jianmin Wang, Xiang Lian, and Lei Zou

Abstract—Identifying events from different sources is essential to various business process applications such as provenance querying or process mining. Distinct features of heterogeneous events, including opaque names and dislocated traces, prevent existing data integration techniques from performing well. To address these issues, in this paper, (1) we propose an event similarity function by iteratively evaluating similar neighbors. (2) In addition to event nodes, we further employ the similarity of edges (indicating relationships among events) in event matching. We prove NP-hardness of finding the optimal event matching w.r.t. node and edge similarities, and propose an efficient heuristic for event matching. Experiments demonstrate that the proposed event matching approach can achieve significantly higher accuracy than state-of-the-art matching methods. In particular, by considering the event edge similarity, our heuristic matching algorithm further improves the matching accuracy without introducing much overhead.

Index Terms—Event similarity, event matching

1 Introduction

Owing to various mergers and acquisitions, information systems (e.g., Enterprise Resource Planning (ERP) and Office Automation (OA) systems), 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 8,190 business processes are implemented in these systems, among which 68.8 percent are indeed different implementations of the same business activities in different subsidiaries. For instance, in the following Example 1, we illustrate two versions of part manufacturing processes in different subsidiaries. Events denoting the same business activities commonly exist in these heterogeneous processes.

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 [21], or obtaining a more abstract global picture of business processes (workflow views [1]) in the company. Without identifying the correspondence among heterogeneous events, applications such as query and analysis over the event data may not yield meaningful results.

The event matching problem is to construct the similarity and matching relationship of events from heterogeneous sources. Manually identifying matching events is (1) obviously inefficient, and (2) could be contradictory. An automatic approach is highly demanded for matching these heterogeneous event data. Rather than manually matching events with great effort, the user could simply confirm the results returned by the automatic approaches. The major benefit to the aforesaid bus company is that the user's effort is greatly reduced in the integration project.

Different from the conventional schema matching on attributes in relational databases [7], events often appear as sequences. The event data integration is challenging due to the following features commonly observed in event data (see examples below): (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.

Example 1. Fig. 1 illustrates two example fragments of event logs $\mathcal{L}_1$ and $\mathcal{L}_2$ for part manufacturing in two different subsidiaries of a bus manufacturer, respectively. Two example traces are shown in each log, where each trace denotes a sequence of events (steps) for processing one part. 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., Functional Detection (B) and Appearance Detection (C) in $\mathcal{L}_1$), or exclusively (e.g., Production Line I (6) or Production Line II (7) in $\mathcal{L}_2$).

Note that opaque names exist in $\mathcal{L}_2$ as shown in Fig. 1b. The event E24T928AE(3) is collected from a

- Y. Gao, S. Song, X. Zhu, and J. Wang are with the Beijing Key Laboratory for Industrial Bigdata System and Application, School of Software, Tsinghua University, Beijing, China. E-mail: {gaoyu15, zhu-xc10}@mails.tsinghua.edu.cn, {sxsong, jimwang}@tsinghua.edu.cn.
- X. Lian is with Kent State University, Kent, OH 44240. E-mail: xlian@kent.edu.
- L. Zou is with Peking University, Beijing 100080, China. E-mail: zoulei@pku.edu.cn.

Manuscript received 13 May 2017; revised 13 Feb. 2018; accepted 9 Mar. 2018. Date of publication 0 . 0000; date of current version 0 . 0000.
(Corresponding author: Shaoxu Song.)
Recommended for acceptance by L. Dong.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TKDE.2018.2815695

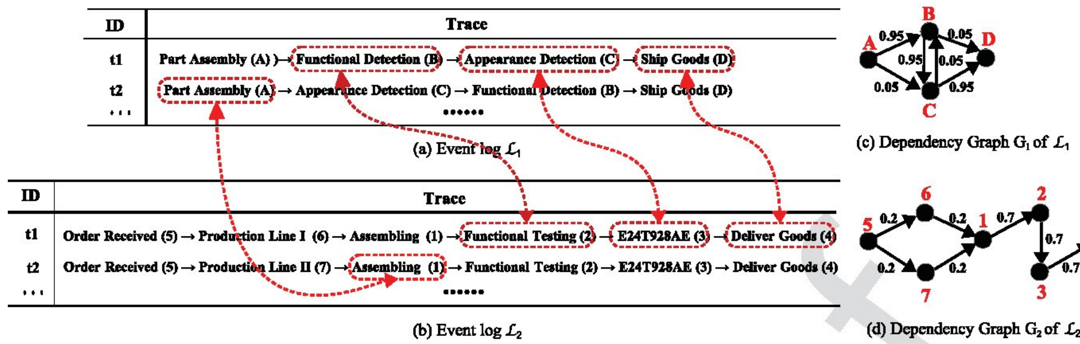


Fig. 1. Fragments of two event logs and their dependency graphs.

source whose encoding is distinct from others, which makes the event name garbled. It indeed denotes a step of "Appearance Testing", and should corresponds to Appearance Detection (C) in $\mathcal{L}_1$. The true event corresponding relation between $\mathcal{L}_1$ and $\mathcal{L}_2$ is highlighted by red dashed lines in Figs. 1a and 1b.

Dislocated matching exists between $\mathcal{L}_1$ and $\mathcal{L}_2$. Event Part Assembly (A) that appears at the beginning of traces in $\mathcal{L}_1$ corresponds to event Assembling (1), which appears in the middle of traces in $\mathcal{L}_2$, having event Production Line I (6) or event Production Line II (7) before it.

Unfortunately, existing techniques cannot effectively address the aforesaid 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) [20] as well as word stemming and the synonym relation (semantic similarity) [22] are widely used in the label similarity based approaches [5], [16], [21], [31]. 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 dependency 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. Both graph edit distance (GED) [5] for general graph data and normal distance for matching with opaque names (OPQ) [13], [14] concern a local evaluation of similar neighbors for two events. However, dislocated matching events, such as event A without preprocessor and event 1 with preprocessor in Figs. 1c and 1d, may not have highly similar neighbors (see more details below). In addition to local neighbors, another type of SimRank [12] like behavioral similarity (BHV) [21] 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., event A in Fig. 1c.

Example 2. Figs. 1c and 1d 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 Fig. 1c for instance) indicates that they appear consecutively in at least one trace (e.g., trace t2 in Fig. 1a). The numbers attached to edges represent the normalized frequencies of consecutive event pairs. For instance, 0.05 of AC means that A, C appear consecutively in 5 percent of the traces in the event log.

Since GED and OPQ concern more about the local similarity, e.g., the high similarity of (A, C) and $(5, 7)$, an event mapping $M = \{A \rightarrow 5, B \rightarrow 6, C \rightarrow 7, D \rightarrow 1\}$ will be returned by GED with distance 0.139 and OPQ with score 6.133. The true mapping $M' = \{A \rightarrow 1, B \rightarrow 2, C \rightarrow 3, D \rightarrow 4\}$ in ground truth shows a higher GED distance 0.183 (lower is better) and lower OPQ score 6.016 (higher is better) instead. BHV does not help in capturing dislocated mapping, e.g., between A and 1 with BHV similarity 0. Instead, A and 5 with no input neighbors have higher similarity 1, i.e., unable to find the dislocated matching.

1.1 Contributions

We notice that the event matching problem consists of two steps: 1) computing the pairwise similarities of events, and 2) determining the matching correspondences of events. While the preliminary version of this paper [32] focuses on computing the event similarities, summarized in Section 3, we further study the second event matching determination problem, i.e., Section 4. In particular, we indicate in Examples 6 and 7 that considering the edge similarities among events in dependency graphs is also essential in event matching. Unlike the matching with only event node similarity, the matching problem with event edge similarity is generally hard. Therefore, an efficient heuristic is studied for event matching. Our major contributions in this paper are summarized as follows.

(1) We formally define the iteratively computed, dislocated matching aware event similarity function. Please refer to the preliminary conference version of this paper [32] for the convergence analysis of iterative similarity computation (in Section 3.3), pruning

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

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 nth iteration
$l(v_1)$	the longest distance from v^X to v_1
M	event matching
$L(v)$	local neighbors of event v

(in Section 3.4) and the detailed algorithm (in Section 3.6).

(2) We study the optimal event matching problem, over the aforesaid computed event similarities. In addition to event node similarities, we show that the edge similarities among events could help in event matching. The problem of finding the optimal event matching w.r.t. node and edge similarities is proved to be NP-hard. An efficient heuristic is thus devised by gradually considering the local optimal matching of events.

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

The remainder of this paper is organized as follows. We introduce the problem in Section 2, provide a detailed analysis of existing solutions in Section 3, propose the updated solution in Section 4, and provide an empirical evaluation in Section 5.

2 Overview of Event Matching

We formalize syntax and definitions for the event matching problem. Table 1 lists the frequently used notations 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^*.

2.1 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 [13], we employ a simple graph model, namely *a dependency graph*, which consists of both dependency relations and frequencies.

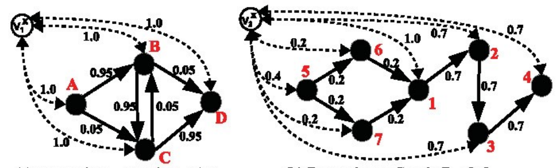
Definition 1 (Event Dependency Graph). *An* event dependency graph G *is a labeled directed graph* (V, E, f), *where each vertex in* V *corresponds to an event,* E *is an edge set, and* f *is a labeling function of normalized frequencies.*

(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.

(2) Each edge $(v, u) \in E$ *denotes that events* vu *occur consecutively at least once in the traces.* $f(v, u)$ *is the normalized frequency of consecutive events* vu, *i.e., the fraction of traces in which* vu *occur consecutively.*

(3) Otherwise, for $v \neq u$, $(v, u) \notin E$, *we have* $f(v, u) = 0$.

For any $v \in V$, *the pre-set of* v *is defined as* $\bullet v = \{u | (u, v) \in E\}$ *and the post-set of* v *is defined as* $v\bullet = \{u | (v, u) \in E\}$.



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

Fig. 2. Dependency graphs with artificial events.

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, ignoring those events before/after v in the dependency graph. 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 to 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 start (edge (v^X, v)) and a virtual end (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 artificial events and edges to dependency graphs, denoted by vertices and edges in dashed lines in Figs. 2a and 2b. 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 5 only appears in 40 percent of the traces of $\mathcal{L}_2$, which indicates $f(v_2^X, 5) = 0.4$.

2.2 Computing Pair-Wise Event Similarity

Based on the dependency graphs of two event logs, $G_1(V_1, E_1, f_1)$ and $G_2(V_2, E_2, f_2)$, the similarity on each event pair, denoted as $\mathcal{S}(v_1, v_2), v_1 \in V_1, v_2 \in V_2$, 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 (see Section 3).[2]

2.3 Determining Event Matching

Once all the pair-wise similarities are obtained between two event logs (dependency graphs), we determine the matching of events referring to the event similarities (see details in Section 4). It is worth noting that the event pairs containing

2. The proposed measure is extensible by integrating with other similarities such as typographic or linguistic similarities [22]. See the preliminary version of this paper [32].

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.

3 Computing Event Similarity

Three categories of techniques may be considered for evaluating event similarities: (1) content-based such as typographic similarities [22], (2) structure-based similarities which concern local structures such as GED [5] and OPQ [13], [14], and (3) structure-based similarities with a global view of the entire graph like SimRank [12]. 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 [12] is not effective, since it does not take edge similarities into consideration, which are the key properties of consecutive occurrence between events. (See experimental evaluation in Section 5.)

In this section, we present an adaption of SimRank like structural similarity function for matching events. Convergence and efficient pruning of unnecessary similarity updates based on early convergence and fast (one iteration) estimation of similarities are presented in the preliminary conference version of this paper [32].

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 u_1 of v_1, i.e., $u_1 \in \bullet v_1$, can find a similar $u_2 \in \bullet v_2$, i.e., predecessor of v_2. Note that this s measure is asymmetric, having $s(v_1, v_2) \neq s(v_2, v_1)$.

Next, to adapt SimRank like evaluation for event similarity, we further take edge similarities into consideration. Although the predecessors u_1 and u_2 of v_1 and v_2, respectively, have high similarity, if the frequency of (u_1, v_1) deviates far from the frequency of (u_2, v_2), the similarity of u_1 and u_2 will have less effect on the similarity of v_1 and v_2.

Following these intuitions, we define a forward similarity w.r.t. predecessors. (Backward similarity on successors can be defined similarly as discussed in Section 3.6 in the preliminary version of this paper [32].)

Definition 2 (Event Similarity). *The forward similarity of two events is*

$$\mathcal{S}(v_1, v_2) = \frac{s(v_1, v_2) + s(v_2, v_1)}{2},$$

where $s(v_1, v_2)$ and $s(v_2, v_1)$ are one-side similarities

$$s(v_1, v_2) = \frac{1}{|\bullet v_1|} \sum_{u_1 \in \bullet v_1} \max_{u_2 \in \bullet v_2} C(u_1, v_1, u_2, v_2) \mathcal{S}(u_1, u_2),$$

given that

$$C(u_1, v_1, u_2, v_2) = c \cdot \left(1 - \frac{|f_1(u_1, v_1) - f_2(u_2, v_2)|}{f_1(u_1, v_1) + f_2(u_2, v_2)}\right),$$

where c is a constant having $0 < c < 1$.

We now explain how these formulas implement our intuition. In the formula of $s(v_1, v_2)$, for each in-neighbor u_1 of v_1, we find an event u_2 with the highest similarity to u_1 among all the in-neighbors of v_2. Besides the node similarity $\mathcal{S}(u_1, u_2)$, evaluating how similar u_1 and u_2 are, we also consider the similarity of the edges (u_1, v_1) and (u_2, v_2), i.e., $C(v_1, u_1, v_2, u_2)$. Recall that edges denote the consecutive occurring relationships of events. Obviously, if (u_1, v_1) and (u_2, v_2) have similar normalized frequencies, $C(v_1, u_1, v_2, u_2)$ is close to c; otherwise close to 0, where c gives the rate of similarity decay across edges.

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 nth 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 $n \geq 1$.

3.2.1 *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. We set $\mathcal{S}^0(v_1^X, v_2)$ and $\mathcal{S}^0(v_1, v_2^X)$ to 0 for any $v_1 \in V_1, v_2 \in V_2$ that are not artificial. Similarly, for any other event pair (v_1, v_2), $\mathcal{S}^0(v_1, v_2)$ is set to 0, since there is no a priori knowledge for assigning nonzero values as initial similarities.

3.2.2 *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 nth iteration can be computed by:

$$\begin{aligned} \mathcal{S}^n(v_1, v_2) &= \frac{s^n(v_1, v_2) + s^n(v_2, v_1)}{2}, \\ s^n(v_1, v_2) &= \frac{1}{|\bullet v_1|} \sum_{u_1 \in \bullet v_1} \max_{u_2 \in \bullet v_2} C(u_1, v_1, u_2, v_2) \mathcal{S}^{n-1}(u_1, u_2). \end{aligned} \tag{1}$$

The similarities involving artificial 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). Initially, $\mathcal{S}^0(v_1^X, v_2^X)$ is assigned with 1.0, and $\mathcal{S}^0(v_1, v_2)$ is assigned with 0 for any other event pairs where $v_1 \neq v_1^X$ or $v_2 \neq v_2^X$. Consider the event pair $(A, 5)$. In the first iteration, we have $|\bullet A| = 1$, $C(v_1^X, A, v_2^X, 5) = 1.0 - |\frac{1.0-0.4}{1.0+1.4}| = 0.571$ and $\mathcal{S}^0(v_1^X, v_2^X) = 1.0$, so that $s^1(A, 5) = \frac{1}{|\bullet A|} C(v_1^X, A, v_2^X, 5) \mathcal{S}^0(v_1^X, v_2^X) = 0.571$. $s^1(5, A) = \frac{1}{|\bullet 5|} C(v_1^X, 5, v_2^X, A) \mathcal{S}^0(v_1^X, v_2^X) = 0.571$ can be got in the same way. so $\mathcal{S}^1(A, 5) = 0.5 * (s^1(A, 5) + s^1(5, A)) = 0.571$. For the event pair $(A, 1)$, we have $s^1(A, 1) = \frac{1}{|\bullet A|} C(v_1^X, A, v_2^X, 1) \mathcal{S}^0(v_1^X, v_2^X) = 1.0$ and $s^1(1, A) = \frac{1}{|\bullet 1|}(C(v_2^X, 1, v_1^X, A)\mathcal{S}^0(v_2^X, v_1^X) + C(6, 1, v_1^X, A)\mathcal{S}^0(6, v_1^X) + C(7, 1, v_1^X, A)\mathcal{S}^0(7, v_1^X)) = 0.333$, so that $\mathcal{S}^1(A, 1) = 0.5 \times (1.0 + 0.333) = 0.666$. It is notable that A and 1 have higher similarity than

A and 5, which solves the problem of dislocated matching. Indeed, by aggregating the event similarities specified in a matching (e.g., by summation in Definition 3), the true mapping M' in Example 2 has a higher (better) score 2.652 than that of M (i.e., 1.539).

The time complexity of computing forward similarity is $O(k|V_1||V_2|d_{\text{avg}})$, 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 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. It can be interpreted as trading the accuracy for efficiency.

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

$$\begin{aligned}\mathcal{S}^n(v_1, v_2) = \frac{1}{2}\Big[&\frac{1}{|\bullet v_1|}\Big(C(v_1^X, v_1, v_2^X, v_2)\mathcal{S}(v_1^X, v_2^X)\\ &+ \sum_{u_1 \in \bullet v_1 \setminus \{v_1^X\}} \max_{u_2 \in \bullet v_2} C(u_1, v_1, u_2, v_2)\mathcal{S}^{n-1}(u_1, u_2)\Big)\\ &+ \frac{1}{|\bullet v_2|}\Big(C(v_1^X, v_1, v_2^X, v_2)\mathcal{S}(v_1^X, v_2^X)\\ &+ \sum_{u_2 \in \bullet v_2 \setminus \{v_2^X\}} \max_{u_1 \in \bullet v_1} C(u_1, v_1, u_2, v_2)\mathcal{S}^{n-1}(u_1, u_2)\Big)\Big].\end{aligned}$$

For simplicity, we denote C as $C(v_1^X, v_1, v_2^X, v_2)$, A as $|\bullet v_1|$, and B as $|\bullet v_2|$. The formula is further derived.

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

Let $q = \frac{c(2AB-A-B)}{2AB}$ and $a = \frac{(A+B)}{2AB}C$. 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 $I = 0, \ldots, n-1$ denotes the number of iterations. 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 in the preliminary version of this paper [32], 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 the iteration I is, the closer the estimation values and the exact values are. The corresponding time costs are higher as well (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 5. Referring to the estimation formula, given $I = 0$, the value of $\mathcal{S}(A, 5)$ can be estimated by $\mathcal{S}^1_{\text{es}}(A, 5) = C(v_1^X, A, v_2^X, 5) = 0.571$, which is equal to the exact value of $\mathcal{S}(A, 5)$. However, for the event pair $(D, 1)$, having $h = \min(l(D), l(1)) = 3$, if we set $I = 1$, the estimated value $\mathcal{S}^3_{\text{es}}(D, 1)$ is 0.605, while the exact similarity is 0.397. This is because the estimation formula treats the similarity of events D and 1 as the similarity of their ancestors. When we set $I = 2$, the estimated similarity of event pair $(D, 1)$ is $\mathcal{S}^3_{\text{es}}(D, 1) = 0.520$, which is closer to the exact value.

4 Determining Event Matching

Once the pair-wise similarities of events are computed, in this section, we study the problem of generating an event matching. We formalize the optimal event matching problem, analyze its hardness, and present an efficient heuristic algorithm.

4.1 Problem Statement and Analysis

A matching M of events between two dependency graphs $G_1(V_1, E_1, f_1)$ and $G_2(V_2, E_2, f_2)$ is a mapping $M : V_1 \to V_2$, where no two events in V_1 are mapped to the same event in V_2, i.e., no conflict. (Without loss of generality, we assume $|V_1| \le |V_2|$.) For an event $v_1 \in V_1$, $v_2 = M(v_1)$ is called the corresponding event of v_1, and $v_1 \to v_2$ is called a matched/corresponding event pair.

Intuitively, the larger the similarities between captured events, the better the matching M will be. We may employ the following node matching score to measure the magnitude of the event similarities captured by M.

Definition 3 (Node Matching Score). *The node matching score of M is defined as:*

$$\mathcal{D}_N(M) = \sum_{v \in V_1} D_V(v, M(v)),$$

where $D_V(v, M(v)) = \mathcal{S}(v, M(v))$ *denotes the similarity between events (nodes).*

Unfortunately, the aforesaid measure treats events as a set in an event log, without considering the structure among events in the dependency graphs. Irrational matching could be generated following this measure.

Example 6 (Node Matching Score). Consider two dependency graphs G_1 and G_2 in Fig. 3 (the artificial events are omitted in matching). Let $M = \{A \to 9, B \to 2, C \to 3, D \to 4, E \to 5, F \to 7, G \to 6\}$ be a possible matching with node matching score $\mathcal{D}_N(M) = 6.2$ according to Definition 3. Such a matching is obviously irrational referring to the structure among events. As shown in Fig. 3, event F occurs before G in dependency graph G_1, while event 7 (corresponding to F in M) occurs after 6 (corresponding

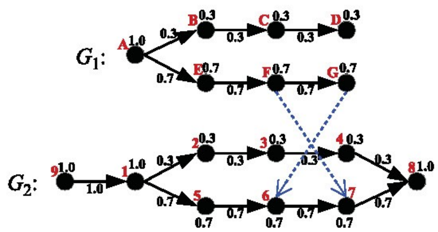


Fig. 3. Dependency graphs for matching.

to G) in G_2. Indeed, the true matching $M' = \{A \to 1, B \to 2, C \to 3, D \to 4, E \to 5, F \to 6, G \to 7\}$ has a lower node matching score $\mathcal{D}_N(M') = 6.1$.

To capture the structural information, we consider the graph matching score below, which involves similarity on edges of events, in addition to event node similarities, between two dependency graphs $G_1(V_1, E_1, f_1)$ and $G_2(V_2, E_2, f_2)$.

Definition 4 (Graph Matching Score). *The graph matching score of M is defined as:*

$$\mathcal{D}_G(M) = \sum_{v,u \in V_1} D_E(v, u, M(v), M(u)),$$

where $D_E(v, u, M(v), M(u))$

$$= \begin{cases} \mathcal{S}(v, M(v)) & \text{if } v = u \\ 1 - \frac{|f_1(v,u) - f_2(M(v),M(u))|}{f_1(v,u) + f_2(M(v),M(u))} & \text{if } v \neq u, (v,u) \in E_1 \\ 0 & \text{if } v \neq u, (v,u) \notin E_1. \end{cases}$$

For $v \neq u$, we denote $1 - \frac{|f_1(v,u) - f_2(M(v),M(u))|}{f_1(v,u) + f_2(M(v),M(u))}$ the similarity of edges (v, u) and $(M(v), M(u))$ on frequency. If $(v, u) \notin E_1$ or $((M(v), M(u)) \notin E_2$, it means $f_1(v, u) = 0$ or $f_2(M(v), M(u)) = 0$, which leads to $D_E(v, u, M(v), M(u)) = 0$.

Example 7 (Graph Matching Score). Consider again the matching $M = \{A \to 9, B \to 2, C \to 3, D \to 4, E \to 5, F \to 7, G \to 6\}$ in Example 6. For the aforesaid irrational matching $F \to 7, G \to 6$, we have $D(F, G, 7, 6) = 0$, i.e., the lowest edge similarity. Consequently, according to Definition 4, the graph matching score of M is $\mathcal{D}_G(M) = 8.2$, which is lower than $\mathcal{D}_G(M') = 12.1$ of the true matching M'.

The event matching problem is thus to find a matching with the highest node/graph matching score.

Problem 1 (Optimal Event Matching Problem). *Given two dependency graphs G_1 and G_2 with the pair-wise event similarity $\mathcal{S}$ computed, the optimal event matching problem is to find a matching M such that $\mathcal{D}_G(M)$ is maximized.*

When the simple node matching score $\mathcal{D}_N(M)$ in Definition 3 is considered, the optimal matching problem can be efficiently solved by Kuhn-Munkres algorithm (a.k.a. the Hungarian algorithm) [15] in $O(|V_2|^3)$ time (assuming $|V_1| \leq |V_2|$). That is, for each pair of events $v_1 \in V_1, v_2 \in V_2$, we define the weight of matching as $D(v_1, v_2) = \mathcal{S}(v_1, v_2)$, the similarity computed in Section 3. It is thus to find an optimal matching M between V_1 and V_2 w.r.t the pair-wise matching weights.

However, if the advanced graph matching score $\mathcal{D}_G(M)$ in Definition 4 is considered, the optimal matching problem is generally hard.

Theorem 1. *Given two dependency graphs G_1 and G_2 with the pair-wise event similarity $\mathcal{S}$ computed, and a constant k, the problem to determine whether there exists an event matching M such that $\mathcal{D}_G(M) \geq k$ is NP-complete.*

Proof. The problem is clearly in NP. Given an event matching M between the two dependency graphs, $\mathcal{D}_G(M)$ in Definition 4 can be calculated in $O(|V_1|^2)$.

To prove NP-hardness of the matching problem, we show a reduction from the subgraph isomorphism problem, which is known to be NP-complete [9]. Given two graphs $G_1(V_1, E_1)$ and $G_2 = (V_2, E_2)$, the subgraph isomorphism problem is to determine whether there is a subgraph $G_0(V_0, E_0) : V_0 \subseteq V_2, E_0 \subseteq E_2 \cap (V_0 \times V_0)$ such that $G_0 \cong G_1$, i.e., whether there exists an $m : V_1 \to V_0$ such that $(v, u) \in E_1 \Leftrightarrow (m(v), m(u)) \in E_0$.

Given two graphs $G_1(V_1, E_1)$, $G_2(V_2, E_2)$, we create two corresponding dependency graphs $G_1(V_1, E_1, f_1)$ and $G_2(V_2, E_2, f_2)$ by associating each vertex and edge appearing G_1 and G_2 with frequency 1. The similarity of any pair of vertices/edges between G_1 and G_2 is 1.

We show that there exists an $m : V_1 \to V_0$ such that $(v, u) \in E_1 \Leftrightarrow (m(v), m(u)) \in E_0$, if and only if there is an event matching $M : V_1 \to V_2$ such that $\mathcal{D}_G(M) \geq k$ where $k = |V_1| + |E_1|$.

First, if such an m exists, we consider m exactly as M. Each edge $(v, u) \in E_1$ corresponds to $(m(v), m(u)) \in E_0$, both with frequency 1. Referring to Definition 4, we have $\mathcal{D}_G(M) = |V_1| + |E_1| = k$.

Conversely, suppose that there is a matching M with $\mathcal{D}_G(M) \geq |V_1| + |E_1| = k$. Referring to $\mathcal{S}(v, M(v)) = 1$ and $1 - \frac{|f_1(v,u) - f_2(M(v),M(u))|}{f_1(v,u) + f_2(M(v),M(u))} = 1$ for any event and edge, it is a matching M with $\mathcal{D}_G(M) = |V_1| + |E_1| = k$, where each vertex (and edge) is matched. It corresponds to $m : V_1 \to V_0$ such that $(v, u) \in E_1 \Leftrightarrow (m(v), m(u)) \in E_0$. □

4.2 Heuristic Algorithm

Recognizing the hardness in Theorem 1, in this section, we propose to devise efficient heuristics. Specifically, we study the local optimal matching w.r.t. an event. Then, it is to gradually improve the matching by finding the local optimal matchings of various events.

4.2.1 Overview

Algorithm 1 presents the pseudo code of heuristic matching. To initialize, the algorithm starts from a feasible matching without conflicts in Line 1, for instance, by using the Kuhn-Munkres algorithm with node matching score as introduced in the paragraph after Problem 1.

Let M_{curr} be the current matching. In each iteration, Algorithm 1 considers the local optimal matchings w.r.t. all the events $v \in V_1$, in Line 1. Among them, it finds the local optimal matching M_{next} with the maximum graph matching score.

The iteration carries on, until the improvement from M_{curr} to M_{next} is not significant, i.e., less than a preset threshold η, $\mathcal{D}_G(M_{\text{next}}) - \mathcal{D}_G(M_{\text{prev}}) \leq \eta$. (See an evaluation on η in Fig. 11 in Section 5.3.)

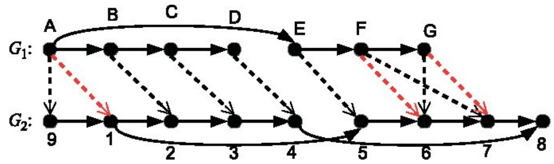


Fig. 4. Dependency graphs with matching.

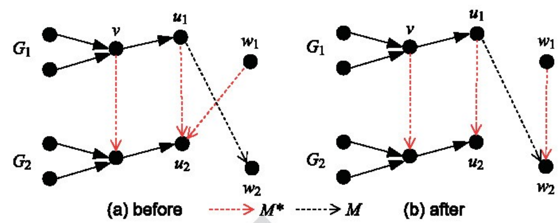


Fig. 5. Local optimal matching and conflict solving.

Algorithm 1. MATCHING(G_1, G_2, η)

Input: two dependency graphs G_1 and G_2, and a threshold η of minimum improvement
Output: event matching M

```
1: M_curr := KM(V_1, V_2)
2: repeat
3:   M_next := M_curr
4:   for each v ∈ V_1 do
5:     M_local := LOCAL(G_1, G_2, M_curr, v)
6:     if D_G(M_local) > D_G(M_next) then
7:       M_next := M_local
8:   M_prev := M_curr
9:   M_curr := M_next
10: until D_G(M_next) − D_G(M_prev) ≤ η
11: return M_curr
```

Example 8. Consider again the two dependency graphs G_1 and G_2 in Fig. 3. Let $M = \{A \to 9, B \to 2, C \to 3, D \to 4, E \to 5, F \to 7, G \to 6\}$ be the initialized matching, represented by black dashed arrows in Fig. 4. In the iteration, suppose that E$\in V_1$ is the currently considered event, which is mapped to 5$\in V_2$. By the LOCAL function (see details below) in Line 5 in Algorithm 1, a local optimal matching $M' = \{A \to 1, B \to 2, C \to 3, D \to 4, E \to 5, F \to 6, G \to 7\}$ is found, which maps F (adjacent to the current event E) to 6 (adjacent to 5) and correspondingly $G \to 7$ to avoid conflict. Similarly, event A (adjacent to E as well) is mapped to 1 (adjacent to 5). Since M' is already the optimal matching with the maximum graph matching score, no further improvement could be made in the next iteration. The algorithm terminates and returns M'.

4.2.2 Local Optimal Matching

Given a current matching M, we propose to improve M by finding the local optimal matching M^* w.r.t. an event v. In the following, we (1) define the local matching score w.r.t. v in Definition 5, (2) formalize the local optimal matching problem w.r.t. v in Problem 2, and (3) show that the problem is solvable again by the Kuhn-Munkres algorithm.

Let $L(v) = \bullet v \cup v \bullet$ denote all the local neighbors of v. We study the matching scores over v and its neighbors.

Definition 5 (Local Matching Score). *The local matching score of M over an event v is defined as:*

$$\mathcal{D}_L(M, v) = \sum_{u \in L(v)} \big(D_E(v, u, M(v), M(u)) + D_E(u, v, M(u), M(v)) + D_E(u, u, M(u), M(u))\big) + D_E(v, v, M(v), M(v))$$

where $D_E(\cdot,\cdot,\cdot,\cdot)$ is the same as in Definition 4.

Given the current matching M, the improvement of M w. r.t. event v is thus to find a matching M^* with the maximum local matching score $\mathcal{D}_L(M^*, v)$ over v.

Problem 2 (Local Optimal Matching Problem). Given two dependency graphs G_1 and G_2 with the pair-wise event similarity $\mathcal{S}$ computed, a current matching M and an event $v \in V_1$, the local optimal matching problem is to find a matching M^* over $L(v)$ such that $\mathcal{D}_G(M^*, v)$ is maximized and $M^*(v) = M(v)$.

To solve the local optimal matching problem, we again employ the Kuhn-Munkres algorithm. That is, for each pair of events $u_1 \in L(v), u_2 \in L(M(v))$, we define the weight of matching as $D_V(u_1, u_2) = \mathcal{S}(u_1, u_2) + D_E(u_1, v, u_2, M(v)) + D_E(v, u_1, M(v), u_2)$. It is thus to find an optimal node matching M^* between $L(v)$ and $L(M(v))$ with the aforesaid pair-wise matching weights.

4.2.3 Resolving Conflicts

The aforesaid local optimal matching M^* specifies only the mapping over $L(v)$. To form an improved matching of M over all the events in V_1, we simply assign $M^*(w) = M(w)$ for all $w \notin L(v) \cup \{v\}$. The problem is that some $u \in L(v) \cup \{v\}$ and aforesaid w may be mapped to the same event in V_2, i.e., conflict in matching M^*. To resolve such conflicts, we manage to modify the mapping on w.

Claim 2. *During conflict resolving, there are at most two events $u_1, w_1 \in V_1$ mapped to the same $u_2 \in V_2$, having $M^*(u_1) = M^*(w_1) = u_2$, where one event must belong to $L(v) \cup \{v\}$, say $u_1 \in L(v) \cup \{v\}$, and the other $w_1 \notin L(v) \cup \{v\}$.*

Case 1: If u_1 has $M^*(u_1) = u_2 \neq w_2 = M(u_1)$, as illustrated in Fig. 5, we assign $M^*(w_1) = w_2$ such that the conflict is resolved.

Case 2: Otherwise, referring to $|V_1| \le |V_2|$, there must exist some $w_2 \in V_2$ which is not matched in M^*, we assign $M^*(w_1) = w_2$ such that the conflict is resolved.

Proof of Claim 2. First, referring to the local optimal matching, no conflict will be introduced between events inside $L(v) \cup \{v\}$. And the mapping on events outside $L(v) \cup \{v\}$ is not changed from M to M^*. Therefore, conflicts may occur only between $u_1 \in L(v) \cup \{v\}$ and $w_1 \notin L(v) \cup \{v\}$.

During conflict resolving, Case 1 assigns $M^*(w_1) = w_2$ where $w_2 = M(u_1)$. If there exists some $v_3 \in L(v) \cup \{v\}$ having $M^*(v_3) = w_2$, the conflict still appears between $v_3 \in L(v) \cup \{v\}$ and $w_1 \notin L(v) \cup \{v\}$. On the other hand, if there does not exist $v_3 \in L(v) \cup \{v\}$ having $M^*(v_3) = w_2$, it means that no other event is mapped to w_2 except w_1 due to $M^*(u_1) = u_2 \neq w_2 = M(u_1)$ in Case 1. That is, no conflict will be introduced.

Case 2 assigns $M^*(w_1) = w_2$ on an unmatched $w_2 \in V_2$, i.e., no conflict will be introduced. □

Algorithm 2 presents the pseudo code of finding the local optimal matching M^* w.r.t. an event v as the improvement of existing matching M. It first initializes the local optimal matching over $L(v)$ by calling again the Kuhn-Munkres algorithm as presented at the end of Section 4.2.2. Each iteration resolves a conflict referring to the aforesaid two cases. While new conflicts may be introduced in iterations, as illustrated in Proposition 3, it is guaranteed to eliminate all the conflicts eventually.

Algorithm 2. LOCAL(G_1, G_2, M, v)

Input: two dependency graphs G_1 and G_2, an event matching M, and an event v
Output: the local optimal matching M^* w.r.t. event v as the improvement of M

1: $M^* := KM(L(v), L(M(v)))$
2: $M^*(w_1) := M(w_1)$ for all $w_1 \in V_1 \setminus (L(v) \cup \{v\})$
3: **repeat**
4: let $u_1 \in L(v), w_1 \notin L(v)$ be two events in conflict having $M^*(u_1) = M^*(w_1)$
5: **if** $M^*(u_1) \neq M(u_1)$ **then**
6: $M^*(w_1) := M(u_1)$
7: **else**
8: let $w_2 \in V_2$ be an unmatched event
9: $M^*(w_1) := w_2$
10: **until** there is no conflict in M^*
11: **return** M^*

Example 9. (Example 8 continued). $M = \{A \to 9, B \to 2, C \to 3, D \to 4, E \to 5, F \to 7, G \to 6\}$ be the current matching, denoted by black dashed arrows in Fig. 4. We illustrate the procedure of finding the local optimal matching for event E. Referring to the dependency graph G_1, we have $L(E) = \{A, F\}$. Similarly, for $M(E) = 5$, we have $L(5) = \{1, 6\}$. By calling the Kuhn-Munkres algorithm in Line 2 in Algorithm 2, we obtain a matching $M^* = \{A \to 1, F \to 6\}$ between $L(E)$ and $L(5)$. Line 2 further initializes M^* on remaining events in V_1 by M, i.e., $M^* = \{A \to 1, B \to 2, C \to 3, D \to 4, E \to 5, F \to 6, G \to 6\}$.

It is notable that conflict exists in M^*, having $F \to 6$, $G \to 6$. For $F \in L(E)$, we have $M^*(F) = 6 \neq 7 = M(F)$, i.e., Case 1. By assigning $M^*(G) = 7$ in Line 2, the conflict is resolved. Since no further conflict is found, the updated $M^* = \{A \to 1, B \to 2, C \to 3, D \to 4, E \to 5, F \to 6, G \to 7\}$ is returned.

4.2.4 Performance Analysis

We first show in Proposition 3 that Algorithm 2 always returns a feasible matching without conflicts, and then analyze the complexity of Algorithm 1 for event matching in Proposition 4.

Proposition 3. *Algorithm 2 returns a local optimal matching which is feasible and maximal, i.e., no conflicts and all events in V_1 are matched (assuming $|V_1| \leq |V_2|$), and runs in $O(d_{\text{avg}}^3)$ time, where d_{avg} is the average degree of all the events in the dependency graph.*

Proof. First, all the events in V_1 are mapped, according to Line 2 in Algorithm 2. A conflict assignment of u_2 is made either by $M^*(w_1) = u_2, w_1 \notin L(v) \cup \{v\}$ in initialization or by $M^*(u_1) = u_2, u_1 \in L(v) \cup \{v\}$ in Case 1. In particular, the new conflict in Case 1 is caused only by the preceding conflict in initialization. That is, once the conflict on u_2 is solved, it will not appear again. Referring to at most $O(d_{\text{avg}})$ events v'_x that may have conflicts, the iteration in Algorithm 2 runs in $O(d_{\text{avg}})$ time to resolve all the conflicts. Given the number of neighbors of an event v, $O(d_{\text{avg}})$, the Kuhn-Munkres algorithm in Line 1 in Algorithm 2 needs $O(d_{\text{avg}}^3)$ time. Consequently, Algorithm 2 has time complexity $O(d_{\text{avg}}^3)$. □

Proposition 4. *Algorithm 1 returns a feasible and maximal matching, i.e., no conflicts and all events in V_1 are matched (assuming $|V_1| \leq |V_2|$), and runs in $O(\max\{|V_2|^3, \frac{|V_1|+|E_1|}{\eta} \cdot |V_1| \cdot d_{\text{avg}}^3\})$ time, where d_{avg} is the average degree of all the events in the dependency graph.*

Proof. First, referring to the Kuhn-Munkre algorithm in Line 1 in Algorithm 1 and Proposition 3, it is easy to see the feasible and maximal matching. The Kuhn-Munkre algorithm runs in $O(|V_2|^3)$ time. In each iteration, we call LOCAL algorithm for each $v \in V_1$, with total cost $O(|V_1| \cdot d_{\text{avg}}^3)$. The threshold η in Line 1 in Algorithm 1 indicates that each iteration improves the matching score at least η. Referring to Definition 4 of matching score, we have $\mathcal{D}_G(M) \leq |V_1| + |E_1|$. That is, Algorithm 1 runs at most $\frac{|V_1|+|E_1|}{\eta}$ iterations, with total iteration cost $O(\frac{|V_1|+|E_1|}{\eta} \cdot |V_1| \cdot d_{\text{avg}}^3)$. □

4.2.5 Filtering on Edge Similarity

Recall that in addition to the event node similarity in Definition 3, the graph matching similarity in Definition 4 further considers the event edge similarity between dependency graphs. To maximize the graph matching similarity, it is not surprising that those high similarity edge pairs will make the major contribution. Intuitively, to efficiently evaluate a matching, we may consider only those edge pairs $(v_1, u_1) \in E_1$ and $(v_2, u_2) \in E_2$ with high similarity, having $D_E(v_1, u_1, v_2, u_2) > \theta$ greater than a preset threshold $\theta \in [0, 1]$. When $\theta = 0$, all the edge pairs will be considered without filtering. On the other hand, $\theta = 1$ means that no edge pair will be taken into account.

Proposition 5. *If the edge similarity threshold θ is 1, then the graph matching score in Definition 4 is equivalent to the node matching score in Definition 3, and Algorithm 1 returns the optimal solution.*

Proof. The edge similarity in Definition 4 always has $0 \leq 1 - \frac{|f_1(v,u) - f_2(M(v),M(u))|}{f_1(v,u) + f_2(M(v),M(u))} \leq 1$. Given the edge similarity threshold $\theta = 1$, only the event node similarity $\mathcal{S}(v, M(v))$ will be taken into account, i.e., equivalent to the node matching score in Definition 3. Referring to the discussion after Problem 1, the Kuhn-Munkres algorithm in Line 1 in Algorithm 1 already obtains the optimal matching w.r.t. the node matching score. □

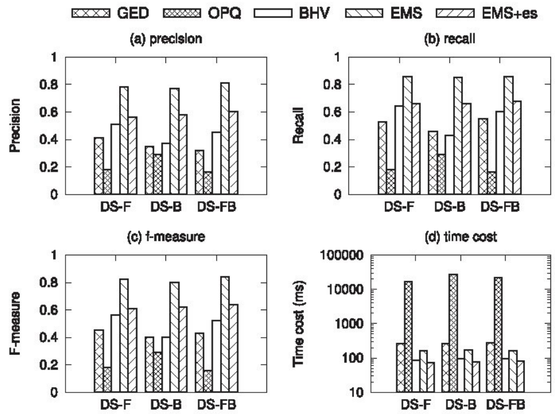


Fig. 6. Evaluating event similarity.

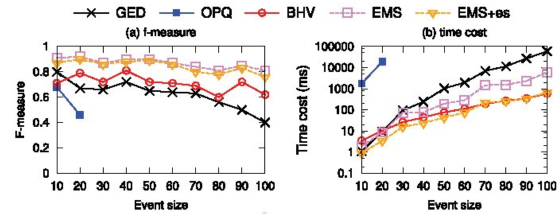


Fig. 7. Scalability on the number of events over synthetic data.

In this sense, the edge similarity filtering provides a trade-off between effectiveness (considering more precise edge similarities) and efficiency (polynomial time solvable without edge similarities). See Fig. 10 for a detailed evaluation on threshold θ.

5 Evaluation

In this section, we report an experimental evaluation on comparing our method with state-of-the-art event matching approaches, *graph edit distance* (GED) [5], *opaque name matching* (OPQ) [13], [14] and *behavioral similarity* (BHV) [21].

5.1 Experimental Settings

5.1.1 Data Sets

We employ a real data set of 103 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.

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.

The number of distinct events in the employed 103 log pairs ranges from 3 to 38, and the total number of traces is 3,000. It is worth noting that the number of distinct events in a log is often not very large in practice [30]. Real process specifications often have events less than 60, according to the recent survey [28]. Referring to the process modeling guidelines [18], business process models should be decomposed if they have more than 50 elements, so that they are easier to read and understand. Nevertheless, to evaluate the approaches over a larger number of events, we consider a synthetic dataset with up to 100 events (in Fig. 7).

To generate the synthetic dataset, an open source toolkit BeehiveZ using existing generating approaches [17], [19] is employed to generate the models and logs. First, 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. Events in two logs with the same name correspond to each other.

5.1.2 Criteria

The ground truth, i.e., the true matching of events among 103 event log pairs, is supplied by 49 subject-matter experts in MIS (Management Information Systems) departments of each subsidiary 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 $precision = \frac{|\text{truth} \cap \text{found}|}{|\text{found}|}$, $recall = \frac{|\text{truth} \cap \text{found}|}{|\text{truth}|}$, 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.

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

5.2 Evaluating Event Similarity

We first report the experimental results on computing event similarity. The compared approaches include our proposed event matching similarity (EMS) and its estimation EMS+es with $I = 5$ proposed in Section 3. The node matching score in Definition 3 is considered for matching which is equivalent to Kuhn–Munkres algorithm [15] as discussed in Section 4 (more advanced matching algorithms are evaluated in Section 5.3 below). The competitors are the existing approaches GED, OPQ and BHV.

Fig. 6 presents the average accuracy and time costs of event matching. The number of events ranges from 3 to 38 in the 103 log pairs. 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

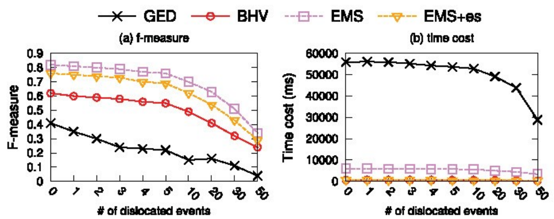


Fig. 8. Performance on handling dislocated events.

DS-B as well as DS-FB) as indicated in Example 1. Our EMS considers similarities in both directions (as indicated in Section 3.1) 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 and OPQ. It is not surprising owing to the high complexity of computing graph edit distance in GED or normal distance in OPQ. 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).

Fig. 7 reports the results of scalability on the number of events (up to 100 events).[3] As shown in Fig. 7a, 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 accuracy decrease is not as significant as other approaches, which means the EMS method is more reliable in event logs with a large number of distinct events. The time costs of all approaches increase heavily in Fig. 7b. OPQ cannot even finish the matching of events in 1000 s under large event sizes, due to the highest time complexity O(n!). Nevertheless, EMS+es always achieves the lowest time cost in all the tests with the number of events ranging from 10 to 100.

Fig. 8 evaluates the performance over various sizes of dislocated events (in the synthetic dataset of 100 events). To 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 Evaluating Event Matching

In this experiment, we illustrate the further improvement on event matching by using the graph matching algorithm proposed in Section 4. The compared approaches are

3. Real event logs, however, often have the number of events bounded by about 60, according to the recent survey [28]. Indeed, referring to the process modeling guidelines [18], workflows should be decomposed if they have more than 50 events, so that they are easier to read and understand.

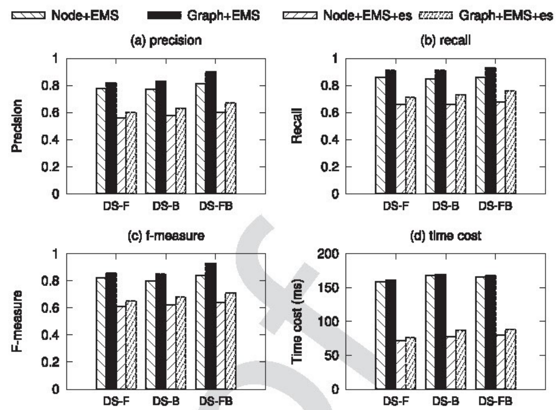


Fig. 9. Evaluating matching algorithms.

(1) Node+EMS and Node+EMS+es using the node matching in Definition 3, i.e., the best approaches presented in the aforesaid experiments in Section 5.2, and
(2) Graph+EMS and Graph+EMS+es using the advanced graph matching in Definition 4.

Given the clearly better results of the (Node+)EMS+es method in the aforesaid experiments, we omit reporting the same results of other existing methods again.

Fig. 9 presents the results by different matching algorithms. As shown in Figs. 9a, 9b, and 9c the accuracy of Graph matching is always higher than the corresponding Node matching methods in all the testbeds. Since the Graph matching further considers the edge similarity in addition to node similarity of events, the time costs of Graph matching are a bit higher in Fig. 9d.

In addition to edge frequency filtering, as presented in Section 4.2.5, we may also introduce edge similarity filtering. Fig. 10 reports the results by varying the edge similarity θ from 0 to 1. A threshold $\theta = 0$ means to consider all the edge similarity pairs between two dependency graphs. The corresponding matching accuracies are high, as well as the matching time costs. With the increase of threshold, both accuracy and time cost drop. When $\theta = 1$, as illustrated in Fig. 10a, Node and Graph matching approaches have the same accuracy. The corresponding time costs are similar as well. The results verify the analysis in Proposition 5 that graph matching is equivalent to node matching in such a case.

Fig. 11 evaluates various thresholds η of matching score improvement in the iteration of Algorithm 1. Recall that the heuristic algorithm ignores all the matching with non-significant improvement ($\leq \eta$). The larger the improvement requirement η is, the less the iterations will be. Thereby, in

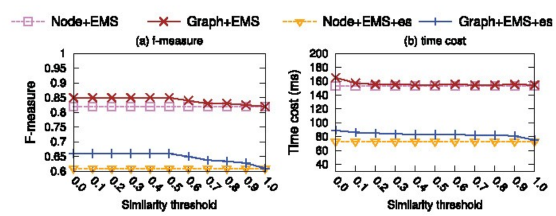


Fig. 10. Matching algorithms with various edge similarity filtering θ.

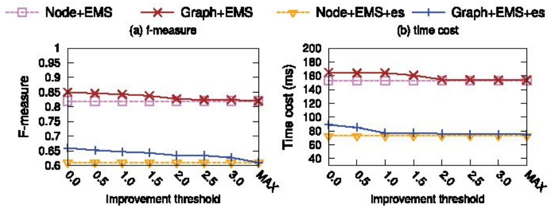


Fig. 11. Matching algorithms with various improvement requirements η in iteration.

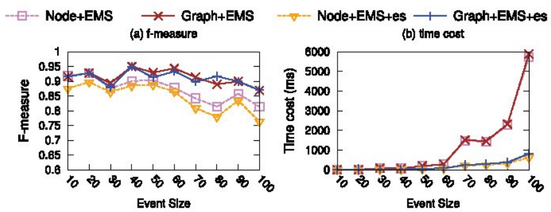


Fig. 12. Scalability of matching algorithms over synthetic data.

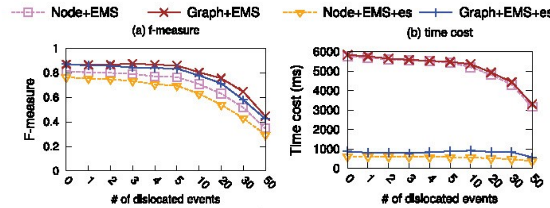


Fig. 13. Matching algorithms on handling dislocated events.

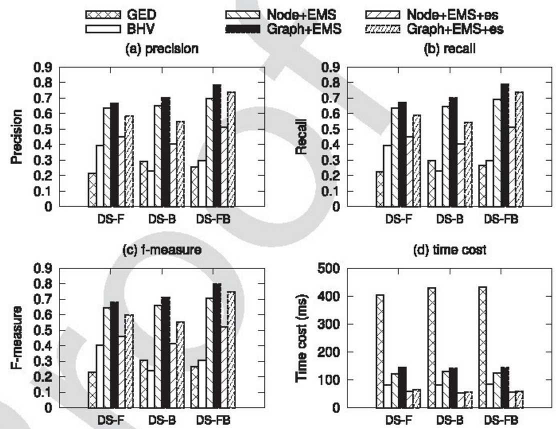


Fig. 14. Performance on matching singleton events over hospital log.

Fig. 11b, the matching time cost drops with the increase of η. Since some non-significant improvement is ignored, the corresponding matching accuracy with large η is lower as well. It is notable that an extreme large η (denoted by MAX in Fig. 11) simply ignores all the improvements by considering edge similarity, i.e., the Graph matching is equivalent to Node matching again.

In short, all the thresholds on edge similarity (in Fig. 10), and matching improvement (in Fig. 11) provide trade-off between matching effectiveness and efficiency. The effects by edge similarity and matching improvement controls are not as significant as on edge frequency. The reason is that they do not affect similarity computation which takes the majority of event matching overhead.

Fig. 12 illustrates the scalability of matching algorithms over various sizes of events. Again, the accuracies of Graph matching approaches are generally higher than those of Node matching methods in all the event sizes. Most importantly, the increase of time costs due to Graph matching is not significant compared to Node matching, especially in larger data sizes in Fig. 12b. The reason is that the computation of event similarity by EMS (or EMS+es) is the most time-consuming part in the process of events matching. Consequently, the results demonstrate that the advanced Graph matching approaches can increase the matching accuracy but without introducing significant computation overhead.

Finally, analogous to Fig. 8, we evaluate the matching algorithms on handling various sizes of dislocated events. As shown in Fig. 13, by increasing the number of dislocated events, the accuracy of all approaches drops as well as the corresponding time cost, which is similar to Fig. 8. The same relationships of Graph and Node matching results are observed again as in the aforesaid Fig. 12.

5.4 Experiments on Hospital Log

To further illustrate that the proposed solution is generic, we employ another real-world dataset, the hospital log,[4] which is publicly available. The data set consists of 1,143 distinct traces over 36 distinct events, collected by a Dutch academic hospital. We randomly sample 80 percent traces from the dataset to form a log. A total number of 200 logs are extracted. Event matching is then applied between these logs. Again, as described in Section 5.1.1, three cases of dislocated events, DS-F, DS-B and DS-FB, are considered, at the beginning, end and both sides, respectively.

Fig. 14 shows the average accuracy and time cost of our proposed approaches EMS with Node matching and Graph matching, compared to the existing methods GED and BHV. The results of OPQ are omitted owing to the extremely higher time costs over the large number of events, which is not surprising referring to Fig. 7b. Generally, the results are similar to Figs. 6 and 9 over the first dataset from the bus manufacturer. That is, our proposed EMS similarity with Graph matching can always achieve the highest accuracy. The result confirms that our proposed approach is generic over different real-world data.

5.4.1 Integrating with Typographical Similarity

It is highly possible to combine the dependency graph based evaluation with the typographical similarity of event names/labels (if available). Indeed, the combination has been studied in Definition 2 in the previous conference version [32] and omitted in this study. That is, the similarity of two events defined in Definition 2 could be $\mathcal{S}(v_1, v_2) = \alpha(s(v_1, v_2) + s(v_2, v_1))/2 + (1-\alpha)\mathcal{S}^L(v_1, v_2)$, where $\mathcal{S}^L(v_1, v_2)$ is the label similarity of events v_1 and v_2, $\alpha \in [0, 1]$ is a weight, $s(v_1, v_2)$ and $s(v_2, v_1)$ are the structural similarities computed from dependency graphs. The results in Fig. 4 in [32] show that by integrating the label similarity of events,

4. http://data.4tu.nl/repository/uuid:d9769f3d-0ab0-4fb8-803b-0d1120ffcf54

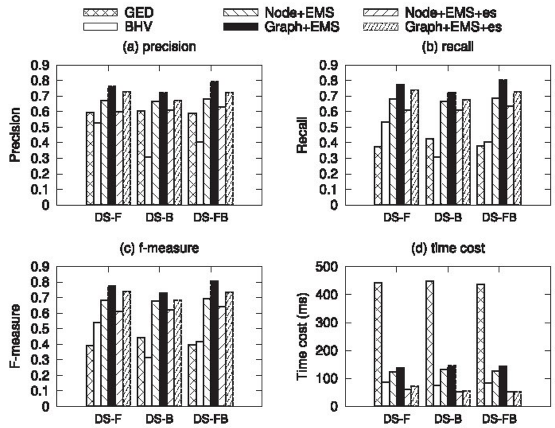


Fig. 15. Performance with typographic similarity.

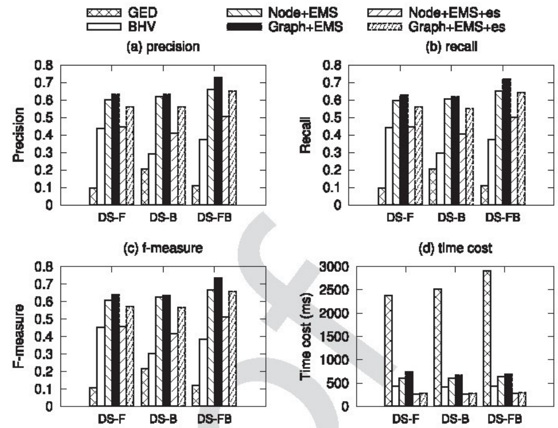


Fig. 16. Evaluation over composite events.

the matching accuracy is improved. Nevertheless, we report again the results evaluated over the second dataset. To simulate the event name differences among logs, we randomly modify (60 percent) characters. Cosine similarity with q-grams [10] is employed to compute the label similarity.

Fig. 15 presents the results by integrating structural similarities with typographic similarities (string similarity of event names). In general, the results are very similar to Fig. 14 without considering the typographic similarity. Moreover, by considering the similarity of event names, all the methods have higher matching accuracy compared to the methods without typographic similarities in Fig. 14. That is, considering event name similarity (if possible) could indeed improve the accuracy.

5.4.2 Handling Composite Events

Similarity matching of composite events (e.g., two events Check Inventory and Validate may correspond to one composite event Inventory Checking Validation) is discussed in the preliminary version of this paper [32]. Owing to the limited space, we focus on the matching of singleton events in this study. For composite events, once the similarities over composite events are identified (as in Section 4 in [32]), the matching between composite events could be similarly determined by either the existing Hungarian algorithm [15] or the edge-similarity-aware Algorithm 1 proposed in Section 4 in this study.

Fig. 16 shows that our proposal (Graph+EMS) still achieves better matching accuracy than the existing Hungarian algorithm (Node) and the graph similarity based methods (GED and BHV) in matching composite events. Indeed, the results are very similar to Fig. 14 of matching singleton events.

5.5 Discussion

As presented in the Introduction, one of the motivations of this study is to handle dislocated events. The results in Fig. 8 show that with a moderate number of dislocated events, our EMS is more effective compared to the existing graph similarity based approach (GED). It demonstrates the superiority of our proposal. However, with the further increase of dislocated events, e.g., the extremely large 50 dislocated events (a half of the total events), the structural information are insufficient for matching events. The accuracy of the proposed method drops and tends to be similar to GED.

6 Related Work

Graphs are 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 [13] by weighting vertices and edges with normalized frequencies, since it distinguishes the significance of distinct edges, and is easy to interpret. An important difference from [13] is the novel artificial node v^X introduced in the dependency graph for matching dislocated events.

Schema matching techniques [23], as a fundamental problem in many database application domains, can be employed to evaluate event similarities. Kang et al. [13], [14] study the matching on opaque data (OPQ). However, as discussed in Example 2, OPQ concerns a direct 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 neighbor distinctness. Moreover, [13], [14] need to enumerate a large number of possible matching correspondences and select the one with the highest normal distance, which is extremely time-consuming. Consequently, the time cost of [13], [14] is high as illustrated in the experimental evaluation in Section 5.

SimRank [12] like behavioral similarity [21] is employed by iteratively considering the predecessor similarities of two events. Unfortunately, this behavioral similarity (BHV) 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. Another graph based similarity is graph edit distance

(GED) [5] which falls short in matching dislocated events. As illustrated in Section 5, both the matching accuracy and time performance of GED are not as good as our proposal.

Once the pair-wise similarities between events in two logs are calculated, the event similarity relationships can be represented as a bipartite graph. To find a matching with the highest similarity, classical algorithms such as Kuhn–Munkres algorithm [15] can be directly applied. As illustrated at the end of Section 4.1, bipartite graph matching method is indeed a special case of our Optimal Event Matching Problem, where only event node similarity is considered. Experimental results in Section 5.3 demonstrate that the event matching accuracy of our proposal is higher than that of the node similarity based Kuhn–Munkres algorithm.

Subgraph isomorphism [2], [24] could be considered for graph-structure based matching. However, in the dislocated event scenario, one graph may not simply "contain" another graph, but overlap (match) only on some nodes. Hoffmann et al. [11] find the maximum common subgraph instead. Event specific information, such as event occurrence frequency, consecutive occurrence frequency or event label similarity (if available), are not considered.

Event matching with additional knowledge has also been studied. Rodriguez et al. [25] employ crowdsourcing and experts to confirm the matching. Automatic matching approaches (including our proposal) could suggest better candidates and thus are complementary to the matching with human intelligence. More complicated event patterns are also considered as distinguishing features for matching [27]. The results heavily rely on how strong the distinguishing power of the specified event patterns is.

7 Conclusions

In this paper, we first identify the unique features that often exist in heterogeneous event logs, such as opaque and dislocated 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 also propose a fast estimation of similarities with only a constant number (including 0) of iterations. For event matching, in addition to event node similarity between two dependency graphs, we further consider the similarity on edges (denoting the consecutive occurrences of events). The hardness and efficient heuristic of event matching with edge similarity are studied.

Experimental results demonstrate that our event similarity shows significantly higher accuracy than state-of-the-art matching approaches. The similarity estimation can significantly reduce time costs while keeping matching accuracy higher/comparable with existing approaches. The event matching with the consideration of edge similarity further improve the accuracy, without introducing much extra overhead.

While the dislocated events could be interpreted as missing events in a log, other event data quality issues such as erroneous events [29] or imprecise timestamps [26] also emerge in practice. Following this intuition, a promising direction is thus to enable event matching with tolerance to such noises (errors), in addition to the dislocated (missing) cases.

Acknowledgments

This work is supported in part by the National Key Research Program of China under Grant 2016YFB1001101; China NSFC under Grants 61572272 and 61202008; Tsinghua University Initiative Scientific Research Program. NSF OAC No. 1739491; Lian Start Up No. 220981, Kent State University.

References

[1] O. Biton, S. C. Boulakia, S. B. Davidson, and C. S. Hara, "Querying and managing provenance through user views in scientific workflows," in *Proc. 24th Int. Conf. Data Eng.*, 2008, pp. 1072–1081.

[2] V. Carletti, P. Foggia, A. Saggese, and M. Vento, "Introducing VF3: A new algorithm for subgraph isomorphism," in *Proc. Int. Workshop Graph-Based Representations Pattern Recognit.*, 2017, pp. 128–139.

[3] F. Casati, M. Castellanos, U. Dayal, and N. Salazar, "A generic solution for warehousing business process data," in *Proc. 33rd Int. Conf. Very Large Data Bases University Vienna*, 2007, pp. 1128–1137.

[4] F. Casati, M. Castellanos, N. Salazar, and U. Dayal, "Abstract process data warehousing," in *Proc. 23rd Int. Conf. Data Eng.*, 2007, pp. 1387–1389.

[5] R. M. Dijkman, M. Dumas, and L. García-Bañuelos, "Graph matching algorithms for business process model similarity search," in *Proc. th Int. Conf. Bus. Process Manag.*, 2009, pp. 48–63.

[6] L. Ding, S. Chen, E. A. Rundensteiner, J. Tatemura, W. Hsiung, and K. S. Candan, "Runtime semantic query optimization for event stream processing," in *Proc. 24th Int. Conf. Data Eng.*, 2008, pp. 676–685.

[7] A. Doan, A. Y. Halevy, and Z. G. Ives, *Principles of Data Integration*. San Mateo, CA, USA: Morgan Kaufmann, 2012.

[8] D. R. Ferreira and D. Gillblad, "Discovering process models from unlabelled event logs," in *Proc. 7th Int. Conf. Bus. Process Manag.*, 2009, pp. 143–158.

[9] M. R. Garey and D. S. Johnson, *Computers and Intractability: A Guide to the Theory of NP-Completeness*. New York, NY, USA: W. H. Freeman, 1979.

[10] L. Gravano, P. G. Ipeirotis, N. Koudas, and D. Srivastava, "Text joins in an rdbms for web data integration," in *Proc. 12th Int. Conf. World Wide Web*, 2003, pp. 90–101.

[11] R. Hoffmann, C. McCreesh, and C. Reilly, Between subgraph isomorphism and maximum common subgraph, in *Proc. Thirty-First (AAAI) Conf. Artif. Intell.*, San Francisco, California, USA, 2017, pp. 3907–3914.

[12] G. Jeh and J. Widom, "Simrank: A measure of structural-context similarity," in *Proc. 8th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2002, pp. 538–543.

[13] J. Kang and J. F. Naughton, "On schema matching with opaque column names and data values," in *Proc. ACM SIGMOD Int. Conf. Manag. Data*, 2003, pp. 205–216.

[14] J. Kang and J. F. Naughton, "Schema matching using interattribute dependencies," *IEEE Trans. Knowl. Data Eng.*, vol. 20, no. 10, pp. 1393–1407, Oct. 2008.

[15] H. W. Kuhn, "The hungarian method for the assignment problem," in *50 Years of Integer Programming 1958–2008 - From the Early Years to the State-of-the-Art*, Berlin, Germany: Springer, pp. 29–47. 2010.

[16] S. Melnik, H. Garcia-Molina, and E. Rahm, "Similarity flooding: A versatile graph matching algorithm and its application to schema matching," in *Proc. 18th Int. Conf. Data Eng.*, 2002, pp. 117–128.

[17] J. Mendling, J. Recker, M. Rosemann, and W. M. P. van der Aalst, "Generating correct epcs from configured C-EPCs," in *Proc. ACM Symp. Appl. Comput.*, 2006, pp. 1505–1510.

[18] J. Mendling, H. A. Reijers, and W. M. P. van der Aalst, "Seven process modeling guidelines (7PMG)," *Inf. Softw. Technol.*, vol. 52, no. 2, pp. 127–136, 2010.

[19] J. Nakatumba, M. Westergaard, and W. M. P. van der Aalst, "Generating event logs with workload-dependent speeds from simulation models," in *Proc. Adv. Inf. Syst. Eng. Workshops*, 2012, pp. 383–397.

[20] G. Navarro, "A guided tour to approximate string matching," *ACM Comput. Surv.*, vol. 33, no. 1, pp. 31–88, 2001.

[21] S. Nejati, M. Sabetzadeh, M. Chechik, S. M. Easterbrook, and P. Zave, "Matching and merging of statecharts specifications," in *Proc. 29th Int. Conf. Softw. Eng.*, 2007, pp. 54–64.

[22] T. Pedersen, S. Patwardhan, and J. Michelizzi, "Wordnet: Similarity - measuring the relatedness of concepts," in *Proc. 19th Nat. Conf. Artif. Intell., 16th Conf. Innovative Appl. Artif. Intell.*, 2004, pp. 1024–1025.

[23] E. Rahm and P. A. Bernstein, "A survey of approaches to automatic schema matching," *VLDB J.*, vol. 10, no. 4, pp. 334–350, 2001.

[24] X. Ren and J. Wang, "Exploiting vertex relationships in speeding up subgraph isomorphism over large graphs," *Proc. VLDB Endowment*, vol. 8, no. 5, pp. 617–628, 2015.

[25] C. Rodríguez, C. Klinkmüller, I. Weber, F. Daniel, and F. Casati, "Activity matching with human intelligence," in *Proc. Int. Conf. Bus. Process Manag.*, 2016, pp. 124–140.

[26] S. Song, Y. Cao, and J. Wang, "Cleaning timestamps with temporal constraints," *Proc. VLDB Endowment*, vol. 9, no. 10, pp. 708–719, 2016.

[27] S. Song, Y. Gao, C. Wang, X. Zhu, J. Wang, and P. Yu, "Matching heterogeneous events with patterns," *IEEE Trans. Knowl. Data Eng.*, vol. 29, no. 8, pp. 1695–1708, Aug. 2017.

[28] J. Wang, T. Jin, R. K. Wong, and L. Wen, "Querying business process model repositories - A survey of current approaches and issues," *World Wide Web*, vol. 17, no. 3, pp. 427–454, 2014.

[29] J. Wang, S. Song, X. Lin, X. Zhu, and J. Pei, "Cleaning structured event logs: A graph repair approach," in *Proc. 31st IEEE Int. Conf. Data Eng.*, 2015, pp. 30–41.

[30] J. Wang, S. Song, X. Zhu, and X. Lin, "Efficient recovery of missing events," *Proc. VLDB Endowment*, vol. 6, no. 10, pp. 841–852, 2013.

[31] M. Weidlich, R. M. Dijkman, and J. Mendling, "The ICoP framework: Identification of correspondences between process models," in *Proc. 22nd Int. Conf. Adv. Inf. Syst. Eng.*, 2010, pp. 483–498.

[32] X. Zhu, S. Song, X. Lian, J. Wang, and L. Zou, "Matching heterogeneous event data," in *Proc. Int. Conf. Manag. Data,*, 2014, pp. 1211–1222.

Yu Gao is working toward the ME degree in the School of Software, Tsinghua University, Beijing, China. His current research interests include event data quality and cleaning.

Shaoxu Song is an associate professor with the School of Software, Tsinghua University, Beijing, China. His research interests include data quality and complex event processing. He has published more than 20 papers in top conferences and journals such as SIGMOD, VLDB, ICDE, the *ACM Transactions on Database Systems*, the *IEEE Transactions on Knowledge and Data Engineering*, the *VLDB Journal*, etc.

Xiaochen Zhu is working toward the PhD degree in the School of Software, Tsinghua University, Beijing, China. His current research interests include event data management and schema matching.

Jianmin Wang is a professor with the School of Software, Tsinghua University. His current research interests include unstructured data management, workflow and BPM technology, benchmark for database system, information system security, and large-scale data analytics.

Xiang Lian is an assistant professor with the Department of Computer Science, Kent State University. His research interests include probabilistic data management and probabilistic RDF graphs.

Lei Zou is an associate professor with the Institute of Computer Science and Technology of Peking University. His research interests include graph database and semantic data management.

▷ **For more information on this or any other computing topic, please visit our Digital Library at** www.computer.org/publications/dlib.