Discovering Social Networks Instantly: Moving Process Mining Computations to the Database and Data Entry Time

Alifah Syamsiyah(✉), Boudewijn F. van Dongen, Wil M.P. van der Aalst

Eindhoven University of Technology
A.Syamsiyah@tue.nl, B.F.v.Dongen@tue.nl, W.M.P.v.d.Aalst@tue.nl

Abstract. Process mining aims to turn event data into insights and actions in order to improve processes. To improve process performance it is crucial to get insights into the way people work and collaborate. In this paper, we focus on discovering social networks from event data. To be able to deal with large data sets or with an environment which requires repetitive discoveries during the analysis, and still provide results instantly, we use an approach where most of the computation is moved to the database and things are precomputed at data entry time. Differently from traditional process mining where event data is stored in file-based system, we store event data in relational databases. Moreover, the database also has a role as an engine to compute the intermediate structure of social network during insertion data. By moving computation both in location (to database) and time (to recording time), the discovery of social networks in a process context becomes truly scalable. The approach has been implemented using the open source process mining toolkit ProM. The experiments reported in this paper demonstrate scalability while providing results instantly.

Keywords: Social network · Process mining · Relational database · Repetitive discovery

1 Introduction

Consider the following example: a production process in a company is conducted by several branches. Each branch has a dedicated group of responsible resources, and a resource may collaborate with other resources from the other branch. Based on the report from company's business analysts, some branches are well-performing, but in some other branches there may be performance problems. In order to solve this problem, the production manager needs information about resource performance in problematic batches; how each resource collaborates with others, how they maintain collaboration with good branches, whether there are under-utilized resources, etc. These questions can be addressed by social network analysis in a process mining context.

Suppose that the production manager instructs the company's business analysts to give a report in each month for checking the progress since the beginning of the year. In this case, the business analysts need to repeatedly look at results based on event logs that grow over time. For the report in January, they look at results based on logs recorded in

January. For the report in February, they analyze results based on logs in January plus February, and so on.

Process mining is a research discipline where the aim is to improve organization's process given the information from the so called *event logs*. The application of social network analysis in process mining was firstly studied in [16, 17]. The combination of these two research disciplines has proven to be valuable: novel insights in business processes performance can be obtained by combining both.

In traditional process mining, we import an event log when we want to perform an analysis and need to do an end-to-end computation at analysis time. This approach suffers from two main problems. First of all, we need to load all event data into main memory to do the computation. For larger event logs the data needs to be partitioned to perform the computation. Second, the computation may be very time consuming. Hence, it may take some time to get results if all of this is done on-demand. Third, in an environment where some repetitive discoveries are required, as illustrated in the example, unnecessary reloading and mining the prior data must be performed.

In this paper, we propose a solution to tackle these problems using mature and established relational database technology. We first move event data to a relational database, define a notion of *intermediate structures* in social networks, pre-compute the intermediate structure inside the database, and set up a connection between the database and process mining tools. By this technique, we easily pass the intermediate structures to existing social network algorithms. The algorithms will run normally and not be aware that the intermediate structures are retrieved from the database. They run even faster because some computations have been already done inside the database. Furthermore, the intermediate structures are always kept alive, i.e., they are always updated when a new data is inserted, hence this technique is well-applied for the needs of repetitive discovery for event data that grows over time.

While many discovery techniques can be identified, this paper focuses on the social network analysis particularly the *handover of work network*. In handover of work network, we analyze how a task is passed over from one resource to the other in the context of a process instance. From the handover of work metric we can get insights into how job allocation is done, whether a resource is always busy compared to others, or whether unnecessary handovers of work happen. In spite of that, the work trivially extends to other types of social networks as long as we can identify an intermediate structure used by the technique which can be updated when inserting new events into the database.

This paper is organized as follows. In Section 2, we present some related work. In Section 3, we give general overview of process mining and the computation of handover of work network. Then, in Section 4, we explain how to enable social network discovery in database. In Section 5 we introduce the step-by-step to mine social networks instantly. We explain the implementation and experimental results in Section 6 and 7 respectively. Finally, Section 8 concludes the paper.

2 Related Work

How social network analysis and process mining can be combined was first shown in [17]. This work gave a foundation of social network metrics in the context of process mining. Based on a log data consisting resource information, one can mine organizational perspective using notions such as handover of work, working together, and subcontracting. As an extension of this work, the authors presented concrete metrics and demonstrated them using a case study in [16].

Visualizing social networks, especially with huge volume of data, is a challenging task. The work in [9] supports social network analysis using chord diagram where nodes are represented by arcs and chords represent interactions among nodes. Each element is defined formally and the results showed that this approach can support investigation of new insights from the social network better than the traditional approach.

To deal with large event logs, the work in [19] presented a streaming-based framework that defines online cooperative network discovery in a generic way. It considers the organizational perspective using real time, online, and, infinite event streams. Stream representation of event logs has also been adapted in [1]. The method uses time-based window model in a finite streams of events. It first transforms the event log into a finite stream of events, then it divides the stream into windows.

Differently from streaming-based technique, the work in [5] employed a hierarchical clustering to deal with large event logs. By clustering the users and considering the working together metric, it discovers a community structure in social networks.

A study in [10] examined multiple online social network at a big data scale. It studied the structural perspective of social network. The data showed that social networks are structurally different from the web network. Social networks have a much higher fraction of symmetric links and also exhibit much higher levels of local clustering.

Social network analysis based on Hadoop was investigated in [20]. This work introduced a framework called big cloud-parallel data mining but only focused on huge scale telecom data. Moreover, social network analysis based on the MapReduce framework was investigated in [14]. However, this work only discussed about social influences in social network analysis.

The use of relational databases in process mining has been investigated earlier. For example, the ontology-based approach in [4] provides on-demand access to the data in the database using query unfolding and rewriting techniques in Ontology-based Data Access [3, 11]. Furthermore, RXES presented in [18] introduced the relational representation of XES.

Building on top of RXES, the work in [12] introduced an approach to discover resource assignment constraints by means of SQL queries. The queries can be customized in order to analyze the interplay of different perspectives, specifically to discover the influence of resources on the control flow of the process. However, this technique does not handle live event data, the focus is on static data that has been imported in a database.

As an improved version of RXES, DB-XES was introduced in [13]. DB-XES defined a basic schema which resembles the standard structure of event data, i.e., the XES standard [8]. To enable a specific family of discovery techniques, this basic schema was extended with directly follows relations (DFR). Moreover it introduced a technique to precompute the DFR inside DB-XES. Using experiments on real-life data, it was shown

that storing event data and DFR in DB-XES not only leads to a significant reduction in memory use of the process mining tool, but can even speed up the analysis of process discovery. However, this technique is limited to the control flow perspective only.

To the best of our knowledge, this is the first work that considers the organizational perspective in process mining while moving computation to the database and recording time such that it is able to discover social networks instantly.

3 Preliminaries

3.1 Process Mining

Process mining is a research discipline that sits between machine learning and data mining on the one hand and process modeling and analysis on the other hand. The goal of process mining is to turn event data into insights and actions in order to improve processes. Three main tasks in process mining are *process discovery, conformance checking,* and *enhancement* [15].

Process mining requires an *event log*, i.e., a set of *traces* of *events*, as the input. Each event has properties called *event attributes*, such as *timestamp* when the event is executed, *activity name* which represents the task name, and *resource* who conduct the event. In this paper, we focus on two particular relations between events, namely the *Directly Follows Relation (DFR)* and the *Causality Relation (CR)*. DFR is a pair of two consecutive events happened in a trace, and CR is a pair of events (x, y) where x is sometimes directly followed by y but y is never followed by x. Formally, they are defined as follows.

Definition 1 (Event Log). *Let E be a set of events. An event log $L \subseteq E^*$ is a set of event sequences (called traces) such that each event appears precisely once in precisely one trace.*

Definition 2 (Event Attributes). *Let E be a set of events and let X be a set of attribute names. For any event $e \in E$ and name $x \in X$: $\#_x(e)$ is the value of attribute x for event e, $\#_x(e) = \perp$ if there is no value. $\{$activity, resource, timestamp $\} \in X$ are standard event attributes, such that:*

- *$\#_{activity}(e)$ is the activity name of event e.*
- *$\#_{resource}(e)$ is the resource who executes event e.*
- *$\#_{timestamp}(e)$ is the timestamp when event e is executed.*

Note that each event is unique and appears only once in the event log. There may be many event sequences that follow the same sequence of activities. However, these are all distinguishable and events in these sequences may have different timestamps, resources, etc.

Definition 3 (Directly Follows Relation). *Let $L \subseteq E^*$ be an event log. x is directly followed by y, denoted $x >_L y$, if and only if there is a trace $\sigma = \langle e_1, e_2, ..., e_n \rangle \in L$ and $1 \leq i < n$ such that $\#_{activity}(e_i) = x$ and $\#_{activity}(e_{i+1}) = y$.*

Definition 4 (Causality Relation). *x is in causality relation with y, denoted $x \rightarrow_L y$, if and only if y sometimes directly follows x but never the other way around, i.e., $x >_L y$ and $y \not> _L x$*

3.2 Social Network Analysis

Social network analysis is an approach for investigating social structures where the context of the social actor, or the relationships between actors are considered [6]. Social networks are applicable to a wide range of substantive domains, ranging from the analysis of concepts within mental models to the study of war between nations. Network methods can also be applied to intrapersonal networks, as well as developmental phenomena such as the structure of individual life histories [2].

In process mining context, social network analysis is harnessed to mine organizational relation. Resource utilization, performance issues, and hierarchical structure within an organization are some examples of the social network usage in process mining. In this context, we derive social networks from the information in the event log. In the following we list some examples of social network analysis in process mining [16]:

- *Subcontracting*. The main idea is to count the number of times individual r_2 executed an activity in-between two activities executed by individual r_1. This may indicate that work was subcontracted from r_1 to r_2.
- *Working-together*. Working-together metric counts how often two resources are working together in the same case. For example, resource r_1 and r_2 work together in twenty cases, while r_1 and r_3 only work together in two cases. In this example, the relation between r_1 and r_2 is stronger than r_1 with r_3.
- *Reassignment*. Reassignment metric measures how often a resource r_1 reassigns the work to another resource r_2. This arises if there is reassign status in the lifecycle extension (one of the standard extension in [8]) stated in the log. This metric provides insight about hierarchical relation of an organization. If r_1 frequently delegates work to r_2 but not vice versa, it is likely that r_1 is in a higher hierarchy than r_2.

3.3 Handover of Work

Handover of work is one of the social network analysis. Within a trace (i.e., process instance), there is handover of work from individual r_1 to individual r_2 if there are two subsequent activities where the first is completed by r_1 and the second by r_2. In [16], there are three kinds of refinement applied to handover of work metrics. First of all, one can differentiate with respect to the degree of causality, e.g., the length of handover. It means that we can consider not only direct succession but also indirect succession. Second, we can ignore multiple transfers within one process instance or not. Third, we can consider arbitrary transfers of work or only consider those where there is a causal dependency.

In this paper we focus on causality relations with degree one, i.e., the directly follows relation. We differentiate the metrics into four categories: (1) *absolute handover of work*, is the basic metric where arbitrary transfers of work with length one are considered, (2) *boolean handover of work*, is the metric for arbitrary transfers of work and ignores multiple transfers within one trace, (3) *absolute causal handover of work*, is similar to (1) but takes into account the causality relation, and (4) *boolean causal handover of work*, is similar to (2) but plus causality.

Suppose we have log $L = \{\langle a_{Alif},\ b_{Berli},\ c_{Charlie},\ d_{Dania},\ e_{Eliaz}\rangle, \langle a_{Alif},\ b_{Berli},\ d_{Dania},\ c_{Charlie},\ e_{Eliaz}\rangle\}$, with x_y denotes that an activity x is executed by a resource y. Based on this log, we mine four types of handover of work as depicted in Figure 1. The figure shows that the structure of absolute and boolean handover of work is the same, however later we will see that the handover values in the two networks are different. Moreover, the difference between non-causal (Figure 1a) and causal (Figure 1b) handover of work is in the edges between *Charlie* and *Dania*. Handover of work which takes into account the causality relation does not have those edges since activity c and d are not in causality relation.

(a) Absolute/boolean handover of work (b) Absolute/boolean causal handover of work

Fig. 1: Handover of work networks for log L

Formally, the four categories of handover of work are formalized as follows [16].

Definition 5 ($\triangleright, \trianglerighteq$)**.** *Let $L \subseteq E^*$ be a log, A be a set of attributes and $R = \{r \mid \exists_{\sigma \in L}\ \exists_{e \in \sigma}\ r = \#_{resource}(e)\}$ be a set of resources. For $a_1, a_2 \in A, r_1, r_2 \in R$, and $\sigma = \langle e_1, e_2, ..., e_n\rangle \in L$:*

- $r_1 \triangleright_\sigma r_2 = \exists_{1 \le i < |\sigma|}\ \#_{resource}(e_i) = r_1\ \wedge\ \#_{resource}(e_{i+1}) = r_2$
- $|r_1 \triangleright_\sigma r_2| = \sum_{1 \le i < |\sigma|} \begin{cases} 1, & \textit{if } \#_{resource}(e_i) = r_1\ \wedge\ \#_{resource}(e_{i+1}) = r_2 \\ 0, & \textit{otherwise} \end{cases}$
- $r_1 \trianglerighteq_\sigma r_2 = \exists_{1 \le i < |\sigma|}\ \#_{resource}(e_i) = r_1\ \wedge\ \#_{resource}(e_{i+1}) = r_2\ \wedge$
  $\#_{activity}(e_i) \rightarrow_L \#_{activity}(e_{i+1})$
- $|r_1 \trianglerighteq_\sigma r_2| = \sum_{1 \le i < |\sigma|} \begin{cases} 1, & \textit{if } \#_{resource}(e_i) = r_1\ \wedge\ \#_{resource}(e_{i+1}) = r_2\ \wedge \\ & \#_{activity}(e_i) \rightarrow_L \#_{activity}(e_{i+1}) \\ 0, & \textit{otherwise} \end{cases}$

Definition 6 (Handover of Work Metrics). *Let L be a log and R be a set of resources. For $r_1, r_2 \in R$, we define:*

- $r_1 \triangleright_L r_2 = \Big(\sum_{\sigma \in L}\ |r_1 \triangleright_\sigma r_2|\Big) \Big/ \Big(\sum_{\sigma \in L}(|\sigma| - 1)\Big)$
  as Absolute Handover of Work.
- $r_1 \triangleright'_L r_2 = \Big(\sum_{\sigma \in L\ \wedge\ r_1 \triangleright_\sigma r_2}\ 1\Big) \Big/ \Big(\sum_{\sigma \in L} 1\Big)$
  as Boolean Handover of Work.
- $r_1 \trianglerighteq_L r_2 = \Big(\sum_{\sigma \in L}\ |r_1 \trianglerighteq_\sigma r_2|\Big) \Big/ \Big(\sum_{\sigma \in L}(|\sigma| - 1)\Big)$
  as Absolute Causal Handover of Work.
- $r_1 \trianglerighteq'_L r_2 = \Big(\sum_{\sigma \in L\ \wedge\ r_1 \trianglerighteq_\sigma r_2}\ 1\Big) \Big/ \Big(\sum_{\sigma \in L} 1\Big)$
  as Boolean Causal Handover of Work.

If we apply log L from the previous example to the metrics above, we get the following handover values: *Charlie*$\triangleright_L$*Dania* $= \frac{1}{8}$, *Charlie*$\triangleright'_L$*Dania* $= \frac{1}{2}$, *Charlie* $\trianglerighteq_L$*Dania* $= 0$, and *Charlie* $\trianglerighteq'_L$*Dania* $= 0$.

4 Enabling Social Network Discovery in Database

As mentioned before, process mining needs the so-called input *event log*. This log is a file-based system and imported to process mining tools every time we do process mining analysis. Existing process mining techniques can handle event log that fits into computer's memory fast. However, the process becomes slower when the log size exceeds the memory since swapping to disk needs to be performed. It becomes more challenging for repetitive discovery where we need to repeatedly mine event data that grows over time.

In this paper, we utilize relational database technology to enable social network discovery. The database is used both for storing event data and computing social network metrics. We choose database technology as the data storage since it is persistent and we can always retrieve back the data when needed. Furthermore, the metrics computed inside the database are designed to be aware of new addition in data, hence there is no need to reload the data as in the case of traditional repetitive discovery. Aside from database, any other persistent storages such as Hadoop and Google File System could also be exploited.

We use a database schema called *DB-XES* as presented in [13]. On top of this schema, we add handover of work tables for each type of the metrics introduced in Definition 6, namely tables *Absolute_HoW, Boolean_HoW, Absolute_Causal_HoW,* and *Boolean_Causal_HoW*. With these tables, we enable social network discovery through DB-XES. Formally, the handover of work tables are defined as follows.

Definition 7 (Handover of Work Tables). *Let $L \subseteq E^*$ be an event log, $N = \{n \mid \exists_{\sigma \in L}\ \exists_{e \in \sigma}\ n = \#_{activity}(e)\}$ is the set of activity names, and $R = \{r \mid \exists_{\sigma \in L}\ \exists_{e \in \sigma}\ r = \#_{resource}(e)\}$ is the set of resources. We define:*

- $\#_{activity}(e) >_L \#_{activity}(e')$ *if and only if* $\exists_{\langle e_1,\ldots,e_n\rangle \in L}\ \exists_{1 \le i < n}\ e = e_i\ \wedge\ e' = e_{i+1}$
- $f(r_1, r_2, e_1, e_2) := r_1 = \#_{resource}(e_1) \wedge\ r_2 = \#_{resource}(e_2) \wedge \#_{activity}(e_1) >_L \#_{activity}(e_2)$
- $g(n_1, n_2, r_1, r_2, e_1, e_2) := n_1 = \#_{activity}(e_1) \wedge\ n_2 = \#_{activity}(e_2) \wedge r_1 = \#_{resource}(e_1) \wedge r_2 = \#_{resource}(e_2) \wedge \#_{activity}(e_1) >_L \#_{activity}(e_2)$
- Absolute_HoW $\in R \times R \nrightarrow \mathbb{N}$ *where:*
  - *dom*(Absolute_HoW) $= \{(r_1, r_2) \in R \times R \mid \exists_{e_1,e_2}\ f(r_1, r_2, e_1, e_2)\}$
  - Absolute_HoW$(r_1, r_2) = \sum_{\langle e_1,\ldots,e_n\rangle \in L}\ |\{i \in \{1, \ldots, n-1\} \mid f(r_1, r_2, e_i, e_{i+1})\}|$
- Boolean_HoW $\in R \times R \nrightarrow \mathbb{N}$ *where:*
  - *dom*(Boolean_HoW) $= \{(r_1, r_2) \in R \times R \mid \exists_{e_1,e_2}\ f(r_1, r_2, e_1, e_2)\}$
  - Boolean_HoW$(r_1, r_2) = |\{\langle e_1, \ldots, e_n\rangle \in L \mid \exists_{1 \le i < n}\ f(r_1, r_2, e_i, e_{i+1})\}|$
- Absolute_Causal_HoW $\in N \times N \times R \times R \nrightarrow \mathbb{N}$ *where:*
  - *dom*(Absolute_Causal_HoW) $= \{(n_1, n_2, r_1, r_2) \in N \times N \times R \times R \mid \exists_{e_1,e_2}\ g(n_1, n_2, r_1, r_2, e_1, e_2)\}$
  - Absolute_Causal_HoW$(n_1, n_2, r_1, r_2) = \sum_{\langle e_1,\ldots,e_n\rangle \in L}\ |\{i \in \{1, \ldots, n-1\} \mid\ g(n_1, n_2, r_1, r_2, e_i, e_{i+1})\}|$

– Boolean_Causal_HoW $\in N \times N \times R \times R \nrightarrow \mathbb{N}$ *where:*
  - *dom*(Boolean_Causal_HoW) = $\{(n_1, n_2, r_1, r_2) \in N \times N \times R \times R \mid \exists_{e_1,e_2}\ g(n_1, n_2, r_1, r_2, e_1, e_2)\}$
  - Boolean_Causal_HoW(n_1, n_2, r_1, r_2) = $|\{\langle e_1, ..., e_n\rangle \in L \mid \exists_{1 \leq i < n}\ g(n_1, n_2, r_1, r_2, e_i, e_{i+1})\}|$

There are some differences between the handover of work metrics (Definition 6) and the handover of work tables (Definition 7). First, the handover of work tables only store the numerator of the handover of work metrics. Second, the handover of work tables only incorporate the directly follows relations, while the handover of work metrics incorporate causality relations. This design choice is preferred because of the nature of intermediate structures which is explained in the next section.

5 Mining Social Networks Instantly

To analyze event data, process mining algorithms typically create an *intermediate structure*, which is an abstraction of event data in a structured way, e.g., the directly follows relation, a prefix-automaton, etc. [13]. In the context of handover of work, the four tables mentioned in Definition 7 are such intermediate structures.

Defining an intermediate structure is not trivial. We have to consider which operation is more suitable to be handled by executing SQL queries over relational databases, which operation can be executed on the fly during analysis. Such design choices heavily influence the performance. Since the intermediate structure reflects the abstraction of *event* data, it should cover operations related to *events*, e.g., a new insertion of event, the order of events, events removal, etc. In other words, the intermediate structure should accommodate all changes in the event data.

In the handover of work case, operations related to events are captured in the numerator of the metrics. Moreover, the causality relations can be derived from the directly follows relations. Therefore, the handover of work tables only store the numerator values and count based on the directly follows relations. As defined in Definition 7, the intermediate structure in handover of work is *a pair of resources (r_1, r_2) (possibly with the corresponding activity names) where r_1 directly handed over the work to r_2, followed by the frequency of how often this pair appears in the log*.

In this paper, the computation of the intermediate structures is done inside DB-XES, unlike traditional process mining which compute these intermediate structures only when a social network is constructed. This migration obviously will accelerate the analysis time since the intermediate structures are now available beforehand. Besides, this approach is well-suited for analysis in increasing data since new insertions in data are automatically captured by the intermediate structures. Figure 2 describes four main steps to mine social networks instantly: (a) *initialization*, (b) *update*, (c) *retrieving the intermediate structure*, and (d) *mining the social network*.

Initialization. The initialization step is done before process mining analysis. During this step, we create intermediate structures from the last snapshot of DB-XES, i.e., we incorporate all elements that are currently stored in DB-XES. For each type of handover of work, we create a SQL query and execute it against DB-XES. For example, to obtain

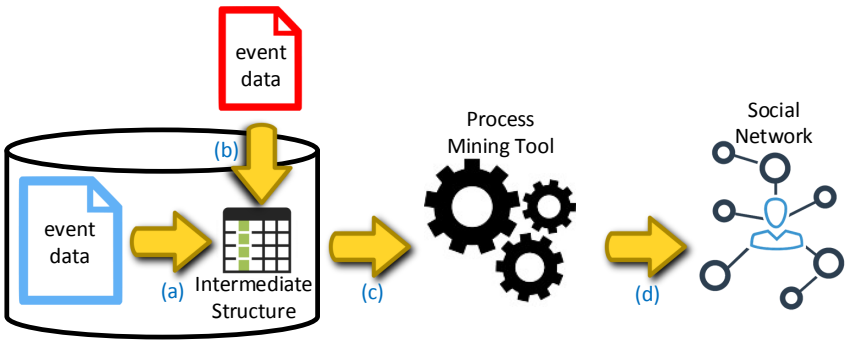


Fig. 2: The steps for mining social networks instantly

the intermediate structure for *absolute handover of work*, the SQL query extracts all pairs of resources whose events happened one after another, and then counts how often it takes place in a particular log (as pointed in the SQL query below; we omit some parts in < ... > for readability). Note that in order to create the intermediate structure for *boolean (causal) handover of work*, we need a temporary intermediate structure. These types of handover of work ignore multiple transfers within one trace, hence we need a temporary intermediate structure to keep track whether a pair of resources has been observed in the same trace. After the initialization step is done, the intermediate structures for four handover of work metrics are ready to be accessed by process mining tools at any time for further analysis.

```
SELECT id, resource1, resource2, count(*) as freq
FROM (  /* get pairs of consecutive events */
        SELECT <...>
        FROM trace_has_event as t1
        INNER JOIN trace_has_event as t2
        ON t1.trace_id = t2.trace_id
        WHERE t1.sequence = t2.sequence - 1
     ) as pairs_of_events,
     attribute as a1, attribute as a2,
     event as event1, event as event2, log_has_trace
/* join condition to take resource values
(resource1 of event1 and resource2 of event2)*/
WHERE <...>
GROUP BY id, resource1, resource2
```

Update. One key advantage of mining social networks within the database is the availability of intermediate structures before analysis time. However, this pre-computation will be useless if it is not aware of changes in the event data. Therefore, the update step becomes crucial in order to keep the intermediate structure up-to-date with the latest view of DB-XES. Note that having live intermediate structures will avoid unnecessary reloading the increasing data in repetitive discovery. To this end, we use SQL

trigger features for doing the update. When there is a new event inserted to a trace, the trigger takes the last event in the trace, retrieves its corresponding resource and activity name, and then updates the intermediate structure tables.

Retrieving the intermediate structures. Once the intermediate structures are available, we can easily access them by executing SQL queries over DB-XES. In general, this operation is relatively fast since the size of intermediate structures is far less than the log size. In the worst case, the size of intermediate structures is as big as the log size, when each resource only executes one event. However, this would be a very atypical scenario where it would not be worthwhile to create a social network.

Mining the social network. The next step is to compute the handover of work values between pairs of resources. As explained before, the intermediate structure only captures the metric's numerator, hence we need to compute the denominator in order to get the handover values. Computing the denominator (i.e., counting the number of events and traces in the log) is a constant operation, hence it can be done during the analysis time. Moreover, we have to deal with causality relation since the intermediate structure only captures the directly follows relation. Given the directly follows graph and causal handover of work tuples (n_1, n_2, r_1, r_2, f) with n_1, n_2 are the activity names, r_1, r_2 are the resources, and f is the frequency number, we select the pair of resources (r_1, r_2) such that the causality relation $n_1 \rightarrow_L n_2$ is preserved in log L. Checking causality relation is a linear operation to the size of intermediate structures. Due to this simplicity, we can compute it on the fly during the analysis and still provide instant mining time.

For the visualization of the network, users can set a threshold value to show the most important relation between two resources. In the node representation, users can select to show the name of resource, the name of event, or combination of both of them.

6 Implementation

We implement this work as a ProM plug-in called *Database Social Network* and it is distributed within the *DatabaseSocialNetwork* package (`https://svn.win.tue.nl/repos/prom/Packages/DatabaseSocialNetwork/Trunk/`).

The *DatabaseSocialNetwork* plug-in requires a database configuration of DB-XES as the input, which includes username and password, database name, server, and a log identifier which the handover of work will be built upon. After the plug-in establishes a connection to DB-XES, it retrieves all necessary rows from the handover of work tables in DB-XES. Based on these row values, it constructs the network representations using Java objects. To visualize the network, it uses the GraphViz library [7] and the current implementation provides support for four types of handover of work: *Absolute Handover of Work*, *Boolean Handover of Work*, *Absolute Causal Handover of Work*, and *Boolean Causal Handover of Work*. To incorporate causality relations, the plug-in gives options to display the activity names executed by the resources. In addition, the plug-in provides a feature to filter infrequent handover values. If the threshold is set to 0.8, for instance, it keeps handover values which are greater or equal than 80% of the maximum value.

Figure 3 shows handover of work mining from a real dataset of a company which contains 154 resources and 846 handovers between resources. The process pertains to

the data entry operations of insurance claim forms. These forms are sent by insurance companies; they are sorted, classified, scanned, and some automated OCR (Optical Character Recognition) are done. They are then sent for manual data entry which could happen at different locations based on the type of form. Once the manual data entry is done, the entries are checked for quality and then archived.

Figure 3 shows handover of work before applying any filtering. This spaghetti-like network is not readable hence it is difficult to grasp any insights. Therefore, we set the threshold value to remove some infrequent edges and nodes as depicted in Figure 4a. The network in this figure is simpler and easier for getting the insights. For example, the network shows that there are handovers of work in both directions between Mr. Auto and Mr. CorrAck. Moreover, the self loop in Mr. Auto indicates that there are some activities which are executed consecutively by the same resource.

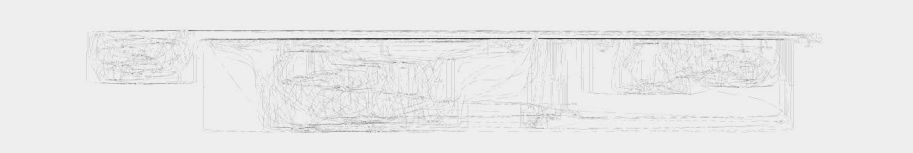
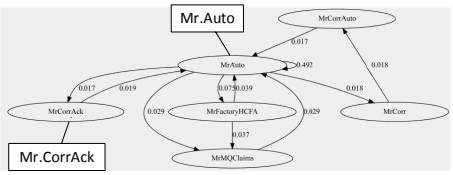
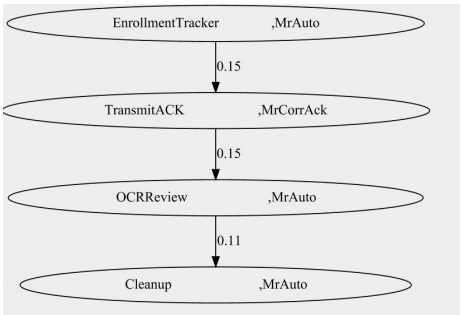


Fig. 3: Social network based on handover of work before applying any filtering



(a)



(b)

Fig. 4: (a) Social network based on handover of work after setting the threshold value, (b) Social network based on handover of work with activity name information

In order to see the activity name executed by these resources, the plug-in provides a feature to display nodes with resources and activity names information. In Figure 4b, we see that after Mr. Auto did ”Enrollment Tracker”, he handed over the work to Mr. CorrAck to do the ”Transmit ACK”. Then, Mr. CorrAck handed back the work to Mr. Auto for doing ”OCR Review”, and finally Mr. Auto himself who ”Clean up” the process.

7 Experimental Result

This section discusses the experimental result. We conducted two different experiments: (1) an experiment to show that the time spent in handover of work analysis based on DB-XES is less than the traditional approach, ignoring the in-database computations, and (2) an experiment to show the running time of each step in mining a handover of work from DB-XES and the feasibility of the approach for repetitive discovery.

In the first experiment, we used a real dataset from a company with some extensions in the number of traces, events and attributes. We extend the dataset by inserting copies of the original dataset with some modifications in the identifier, activity name, and timestamp. More precisely, we extend the dataset to be two, three, four, six, and eight times bigger than the original dataset.

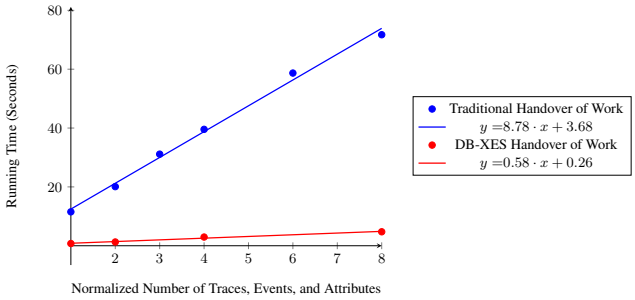


Fig. 5: Comparison between mining handover of work using traditional approach and DB-XES

The result from the first experiment is shown in Figure 5. Here we compare two different approaches for mining absolute handover of work. The first approach (shown as a red line in the figure) used DB-XES for storing the event data and computing the intermediate structure. The second approach (shown as a blue line in the figure) used the traditional way where the event data is stored in XES file and imported to ProM. In this figure, the x-axis represents the normalized number of traces, events, and attributes. For example, ”2” in x-axis means that the number of traces, events, and attributes is twice bigger than the original dataset. The y-axis represents the running time for mining the network in seconds scale.

As shown in Figure 5, both approaches present a linear trendline. However, the gradient of the traditional approach is higher than the DB-XES approach. This is due to the fact that the handover of work algorithm passes through the log and counts how many times a handover happens between two resources. Therefore, the increasing log size gives linear impact to the running time of the algorithm. In contrast, the increasing log size does not give significant influence to DB-XES since the number of resources does not necessarily increase linearly too. Note that during the mining time, the DB-XES approach only retrieves pre-computed values from the database and computes a handover network based on these. Most of the computation has been already done before the min-

ing time. This experiment proves that the total time needed by users to mine a handover of work from DB-XES is less than the current existing approach when intermediate structures are present in the database.

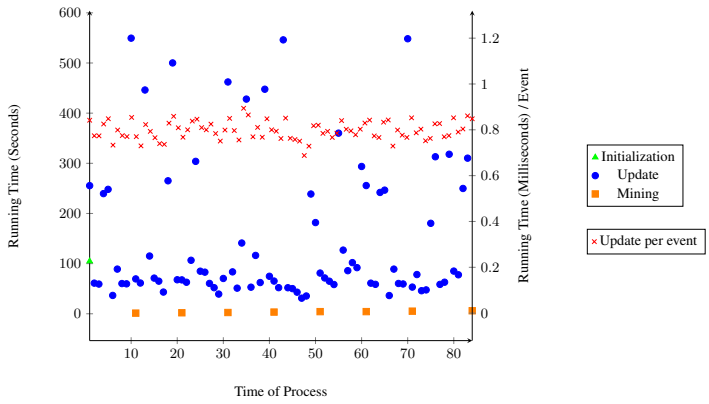


Fig. 6: Running time of each step in mining handover of work from DB-XES: initialization, update, and mining are plotted to the left y-axis; update per event is plotted to the right y-axis

In the second experiment (as depicted in Figure 6), we use a real dataset which includes 460 different resources, around one million traces, and fifteen million events spanning one month. The goal of this experiment is to analyze the end-to-end steps of mining handover of work in DB-XES and to show that the technique is well-suited for repetitive discovery.

At the first step we imported one hundred thousand traces into the database. Then we *initialized* the intermediate structures. In this step we created the intermediate structures from scratch and the running time is shown by the green triangle in the figure. Then we *updated* the database by inserting ten thousand traces. In each insertion, a trigger function is automatically called to update the intermediate structure. The running time to insert ten thousand traces and call the triggers is depicted by the blue dots in the figure. The total number of inserted events in each update is variable, hence we also count the average time to insert a single event. In other words, the average time is the total time to update ten thousand traces divided by the number of events. This average time is depicted by the red cross in the figure. After ten times doing update, i.e., after one hundred thousand new traces were added to the database, we *mined* the handover of work. The running time for mining the network is shown by the orange square. In each mining, the network size was increasing since we added traces every time we did the update (typical scenario in repetitive discovery).

Figure 6 shows four types of marks. The green triangle, blue circles, and orange squares are plotted according to the first y-axis on the left hand side, while the red crosses are plotted according to the secondary y-axis on the right hand side. The x-axis represents time of process where each unit of work is undertaken, the left y-axis rep-

resents the running time in seconds scale, while the right y-axis represents the running time for an event update in milliseconds scale.

From Figure 6 we see that the average time needed to insert an event and call the trigger to update the intermediate result (red crosses) is relatively fast, i.e., 0.8 milliseconds. The update time for a set of traces (blue dots) are variable since the number of imported events in each trace is diverse as well. The mining time (orange squares) are always close to 0, and this shows the increase of log size does not really influence the mining time as the number of resources does not increase dramatically. This also shows that the time in each mining does not really change and quite stable, hence it matches with the needs in repetitive discovery. Moreover, the mining time is always faster than the initialization and update time, which generally can be ignored since these steps can be done offline and run automatically. Our approach always provides immediate answers because there is no need to traverse the event log and reload the prior data.

8 Conclusion

This paper focuses on scalability in social network discovery. We use DB-XES [13] as the relational representation of event log to store event data and shift the work from analysis time to insertion time. On top of the core DB-XES schema, we add specific intermediate structures for social network mining. We define the intermediate structure for handover of work metrics by determining which part should be computed in the database and be kept up-to-date when inserting new events into the database, and which part should be computed on the fly during the analysis time.

The paper explains the approach step-by-step and shows that the time required for mining the result is less than the time required for initialization and update which can be done offline and automatic. Moreover, using experiments on real-life data, the paper shows that the time spent in handover of work analysis based on DB-XES is far less than the existing techniques (thereby ignoring the in-database computations done at insertion). The experiment also shows the compatibility of applying the technique for repetitive discovery where we repeatedly do mining to the data that grows over time.

This paper uses handover of work as the social network. However, the work trivially extends to other types of social networks as long as we can identify an intermediate structure used by the technique which can be updated when inserting new events into the database. For example, in working-together network, a new inserted event will trigger an update in the intermediate structure by looking into all preceding events in the trace. Not only for organizational mining, it is also feasible to apply the approach into others discovery techniques, such as control flow discovery. In fact, the approach has been successfully implemented to do process discovery using Inductive Miner algorithm [13]. Finally, as a concrete research product, this work has been implemented in ProM.

For future work, we plan to implement more social network metrics, such as working together and subcontracting. We also plan to encompass other process discovery paradigms, e.g., declarative process discovery. Besides, we aim to improve the updating process of the intermediate structures by looking at a batch of new inserted events. Hence we reduce the updating time of the intermediate structures which now is always

triggered every time a new event comes. Furthermore, we plan to implement also the removal of events such that the intermediate structures remain consistent under both insertion and deletion of events.

References

1. A. Appice, M. Di Pietro, C. Greco, and D. Malerba. *Discovering and Tracking Organizational Structures in Event Logs*, pages 46–60. Springer, 2016.
2. C.T. Butts. Social Network Analysis: A Methodological Introduction. *Asian Journal of Social Psychology*, page 13, 2008.
3. D. Calvanese, G. De Giacomo, D. Lembo, M. Lenzerini, A. Poggi, M. Rodríguez-Muro, and R. Rosati. Ontologies and Databases: The DL-Lite Approach. In *RW 2009*, volume 5689 of *Lecture Notes in Computer Science*, pages 255–356. Springer, 2009.
4. D. Calvanese, M. Montali, A. Syamsiyah, and W.M.P. van der Aalst. Ontology-Driven Extraction of Event Logs from Relational Databases. In *BPI 2015*, pages 140–153, 2015.
5. D.R. Ferreira and C. Alves. *Discovering User Communities in Large Event Logs*, pages 123–134. Springer Berlin Heidelberg, Berlin, Heidelberg, 2012.
6. B. Furht. *Handbook of Social Network Technologies and Applications*. Springer-Verlag New York, Inc., New York, NY, USA, 1st edition, 2010.
7. Emden R. Gansner. Using Graphviz as a Library, 2014.
8. C.W. Günther. XES Standard Definition. www.xes-standard.org, 2014.
9. A. Jalali. *Supporting Social Network Analysis Using Chord Diagram in Process Mining*, pages 16–32. Springer International Publishing, Cham, 2016.
10. A. Mislove, M. Marcon, K.P. Gummadi, P. Druschel, and B. Bhattacharjee. Measurement and Analysis of Online Social Networks. In *IMC 2007*, pages 29–42, New York, USA, 2007.
11. A. Poggi, D. Lembo, D. Calvanese, G. De Giacomo, M. Lenzerini, and R. Rosati. Linking Data to Ontologies. In *Journal on Data Semantics X*, pages 133–173. Springer-Verlag, Berlin, Heidelberg, 2008.
12. S. Schönig, A. Rogge-Solti, C. Cabanillas, S. Jablonski, and J. Mendling. *Efficient and Customisable Declarative Process Mining with SQL*, pages 290–305. Springer International Publishing, Cham, 2016.
13. A. Syamsiyah, B.F. van Dongen, and W.M.P. van der Aalst. DB-XES: Enabling Process Mining in the Large. In *SIMPDA 2016*, pages 63–77, 2016.
14. J. Tang, J. Sun, C. Wang, and Z. Yang. Social Influence Analysis in Large-Scale Networks. In *KDD 2009*, pages 807–816, New York, USA, 2009.
15. W.M.P. van der Aalst. *Process Mining: Data Science in Action*. Springer, 2016.
16. W.M.P. van der Aalst, H.A. Reijers, and M. Song. Discovering Social Networks from Event Logs. *CSCW*, 14(6):549–593, 2005.
17. W.M.P. van der Aalst and M. Song. *Mining Social Networks: Uncovering Interaction Patterns in Business Processes*, pages 244–260. Springer, 2004.
18. B.F. van Dongen and S. Shabani. Relational XES: Data Management for Process Mining. In *CAiSE 2015*, pages 169–176, 2015.
19. S.J. van Zelst, B.F. van Dongen, and W.M.P. van der Aalst. *Online Discovery of Cooperative Structures inBusiness Processes*, pages 210–228. Springer, 2016.
20. L. Yu, J. Zheng, W.C. Shen, B. Wu, B. Wang, L. Qian, and B.R. Zhang. BC-PDM: Data Mining, Social Network Analysis and Text Mining System Based On Cloud Computing. In *KDD 2012*, pages 1496–1499. ACM, 2012.