Correlation miner

International Journal of Cooperative Information Systems
Vol. 26, No. 2 (2017) 1742002 (32 pages)

DOI: 10.1142/S0218843017420023

Correlation Miner: Mining Business Process Models and Event Correlations Without Case Identifiers

Shaya Pourmirza*, Remco Dijkman† and Paul Grefen‡
School of Industrial Engineering, Eindhoven University of Technology
Den Dolech 2, P. O. Box 513, 5600MB
Eindhoven, The Netherlands
**s.pourmirza@tue.nl*
†*r.m.dijkman@tue.nl*
‡*p.w.p.j.grefen@tue.nl*

Received 13 May 2016
Revised 17 September 2016
Accepted 27 December 2016
Published 18 May 2017

Process discovery algorithms aim to capture process models from event logs. These algorithms have been designed for logs in which the events that belong to the same case are related to each other — and to that case — by means of a unique case identifier. However, in service-oriented systems, these case identifiers are rarely stored beyond request-response pairs, which makes it hard to relate events that belong to the same case. This is known as the *correlation challenge*. This paper addresses the correlation challenge by introducing a technique, called the *correlation miner*, that facilitates discovery of business process models when events are not associated with a case identifier. It extends previous work on the correlation miner, by not only enabling the discovery of the process model, but also detecting which events belong to the same case. Experiments performed on both synthetic and real-world event logs show the applicability of the correlation miner. The resulting technique enables us to observe a service-oriented system and determine — with high accuracy — which request-response pairs sent by different communicating parties are related to each other.

Keywords: Process mining; process discovery; event correlation; business process intelligence (BPI); business process management (BPM).

1. Introduction

Over the past decade, there has been an increasing interest in the area of process mining.[1–8] The goal of process mining is to extract information about processes from event logs, i.e. execution histories. One of the prominent branches of process mining is process discovery,[1] which concerns itself with generating a business process model from an event log.

Process discovery techniques assume that an event log contains at least, for each recorded event: (i) a reference to the executed activity, (ii) a reference to the

*Corresponding author.

Table 1. An example event log.

Case	Activity	Timestamp	Case	Activity	Timestamp	Case	Activity	Timestamp
1	*A*	00:20	4	*B*	05:04	7	*E*	09:17
1	*B*	02:04	4	*E*	07:26	8	*A*	06:20
1	*E*	02:32	5	*A*	03:40	8	*D*	08:36
2	*A*	02:15	5	*B*	05:59	8	*E*	10:03
2	*D*	03:14	5	*E*	07:49	9	*A*	06:41
2	*E*	05:06	6	*A*	04:18	9	*D*	08:56
3	*A*	02:27	6	*C*	07:08	9	*E*	10:20
3	*D*	04:17	6	*E*	09:05	10	*A*	07:13
3	*E*	06:51	7	*A*	05:54	10	*C*	09:10
4	*A*	03:06	7	*C*	07:30	10	*E*	10:26

case for which the activity was executed, and (iii) the timestamp at which the activity was completed.[2] Table 1 shows an example of the event log involving 30 events in 10 cases. The main idea behind discovery algorithms is to merge, cluster and aggregate the different cases in an event log and generate a suitable business process model based on that. For example, considering the event log in Table 1, it is possible to capture three different traces of execution: $\langle A, B, E\rangle$ for Cases 1, 4 and 5; $\langle A, D, E\rangle$ for Cases 2, 3, 8 and 9; and finally, $\langle A, C, E\rangle$ for Cases 6, 7 and 10. Figure 1 presents the corresponding business process model, as it would have been mined by the Disco process mining tool.[9]

While process mining techniques require an event to be associated with a case, making this association can be problematic. This is also referred to as the *correlation challenge*.[5] This challenge arises, for example, when information about the occurrence of activities is stored in separate service-oriented systems and there is no obvious identifier to correlate the occurrences. It is considered especially problematic in service mining,[10] because the correlations between messages that are being exchanged for the same case, may not be stored beyond request-response pairs. When associations between events that belong to the same case are not present, process discovery becomes impossible, as illustrated by Table 2, which shows the event log from Table 1 without case identifiers. In this event log, it is not possible to identify cases and, consequently, it is not possible to cluster and aggregate them into a business process model. Moreover, we cannot see which events belong to the

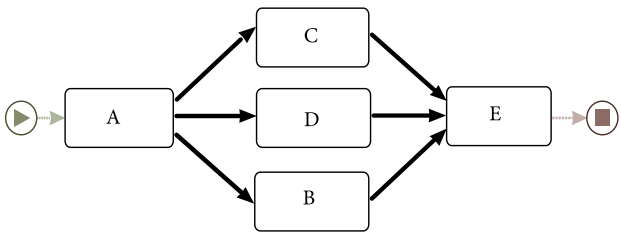


Fig. 1. Business process model generated from the example event log.

Table 2. An example event log without case identifiers.

Activity	Timestamp	Activity	Timestamp	Activity	Timestamp
A	00:20	*B*	05:04	*C*	07:30
B	02:04	*E*	05:06	*E*	07:49
A	02:15	*A*	05:54	*D*	08:36
A	02:27	*B*	05:59	*D*	08:56
E	02:32	*A*	06:20	*E*	09:05
A	03:06	*A*	06:41	*C*	09:10
D	03:14	*E*	06:51	*E*	09:17
A	03:40	*C*	07:08	*E*	10:03
D	04:17	*A*	07:13	*E*	10:20
A	04:18	*E*	07:26	*E*	10:26

same case. For example, both event A that happens at 02:15 and event A that happens at 02:27 could be related to D that happens at 03:14.

Therefore, the goal and contribution of this paper is to develop a process discovery technique, the correlation miner, that can:

Phase I: Discover business process models when events are neither correlated by case identifiers nor by additional data elements; and

Phase II: Discover the relations between the events themselves, which determine the events that belong to the same case.

The main reason for distinguishing between these two phases is that the inputs of Phase II, which is a process model, is actually the output of Phase I. Note that the output process model of Phase I cannot be constructed via traditional process mining techniques, because these techniques require event logs with case identifiers. In summary, in Phase I, a process model is captured from an event log without case identifiers, and in Phase II, an event log with case identifier is generated based on the process model captured in Phase I.

Note that this paper is an extension of our previous paper,[11] which presents only the first phase of the correlation miner. It extends this work by also enabling the discovery of relations between events in an event log. This extension is mainly presented in Secs. 5 and 6.

Although extensive research has been carried out on both process discovery and event correlation techniques, to the best of our knowledge, only one[12] such process discovery technique exists that will be further discussed in this paper. Additionally, event correlation techniques do exist, but they require additional data elements (such as customer names) upon which to perform the correlation.

The remainder of this paper is structured as follows: Section 2 presents some preliminaries regarding the event logs and business process models. Section 3 introduces the first phase of the correlation miner that is used to capture business process models from event logs without case identifiers and Sec. 4 discusses the evaluation of this phase. Section 5 introduces the second phase of the correlation miner that is employed to reconstruct cases from event logs without case identifiers, and subsequently, Sec. 6 presents the evaluation of this phase. Section 7 compares

our study with related literatures. Finally, Sec. 8 concludes the paper by giving a brief overview and its findings.

2. Preliminaries

In the literature, an event log has been defined as a multiset of cases.[5] However, in this paper, we cannot reuse this definition since we have no explicit case. Therefore, we define an event log as follows.

Let $\mathcal{A}$ be a set of activities and T be a set of timestamps. $L \subseteq \mathcal{A} \times T$ is an *event log*. We use E_a to denote a set of events referring to the activity $a \in \mathcal{A}$, such that $E_a = \{(a,t) \,|\, (a,t) \in L\}$. In addition, we represent the timestamp at which event $e \in E_a$ has happened by t_e. Note that, strictly speaking, multiple events for the same activity can happen at the same time. However, for simplicity, this is not covered by our definition. While the occurrence of identical events is sufficiently rare to not create problems with the algorithm, it does require some pre-processing of the log to make sure that identical events are distinguished from each other (e.g. by adding a suffix to the timestamp).

Various notations exist in which the business process model that is the result of applying a process discovery algorithm can be represented (e.g. Petri-Nets, BPMN and YAWL).[5] In this paper, we adopt a rather abstract definition of a business process model, which is in line with the notation that is used by the Disco tool for process discovery.[9] Consequently, we define that a business process model is a directed *weighted* graph $G = (\mathcal{V}, \mathcal{E}, \omega)$ such that:

- $\mathcal{V} = \{(a,n) \,|\, a \in \mathcal{A} \wedge n = |E_a|\}$, where $\mathcal{A}$ is the set of activities and n indicates for each node the number of occurrences of that activity in the event log;
- $\mathcal{E} \subseteq \mathcal{V} \times \mathcal{V}$ is the set of edges;
- $\omega : \mathcal{E} \to \mathbb{N}$ maps edges to natural numbers, where $\omega(a,b) = n$ if and only if there are n observations of activity a being directly followed by activity b in some case in the log. We require that $l \in \mathcal{E}$ if and only if $\omega(l) > 0$.

For a node $v \in \mathcal{V}$, the in-degree, denoted $deg^{in}(v)$, is $\sum_{(w,v)\in\mathcal{E}} \omega(w,v)$. The out-degree, denoted $deg^{out}(v)$, is $\sum_{(v,w)\in\mathcal{E}} \omega(v,w)$. When $deg^{in}(v) = 0$, we call the node a source node. When $deg^{out}(v) = 0$, we call the node a sink node.

It is important to note that, for any node $(a,n) \in \mathcal{V}$ that is not a sink node, it must hold that $deg^{out}(a) = n$, because the number of cases that pass through the node must also continue to another node. Similarly, for any node $(a,n) \in \mathcal{V}$ that is not a source node, it must hold that $deg^{in}(a) = n$. We call these constraints the *process graph rule*.

3. Phase I: Discovery of Process Models from Event Logs

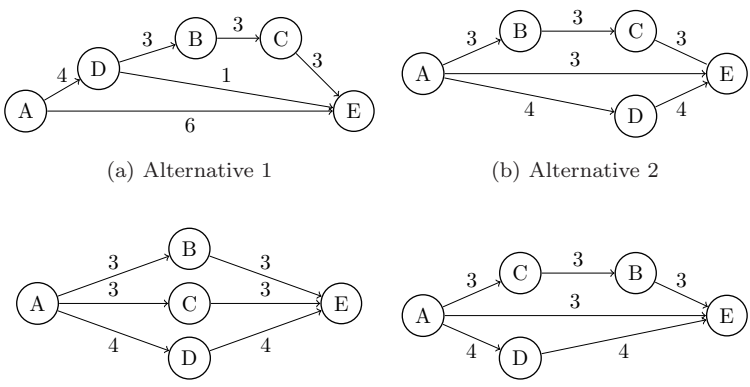
The core idea behind the first phase of the correlation miner is based on the process graph rule, which has been defined in the previous section, stating that the number

of cases that pass through an activity must be equal to the number of incoming and outgoing cases for that activity. Note that this rule primarily holds for acyclic business process models since otherwise one activity can appear multiple times for one case. This implies that in this paper, we limit ourselves to acyclic models, which is a limitation of the work that we aim to study further in future work.

According to this rule, we can count the number of cases that pass through an activity a by counting the number of occurrences $|E_a|$ of that activity in the log. Subsequently, we can draw the edges between the activities, in such a way that the process graph rule is met. This is a constraint programming problem that can be solved using integer linear programming. At this point, we assume that the source and sink nodes are set manually. However, it is also possible to use existing techniques such as the one suggested in Ref. 13 to determine source and sink nodes automatically. Consequently, an important area of future work would be to integrate one of these existing techniques with the correlation miner.

As an example, Fig. 2(a) shows a business process model that can be constructed from the log in Table 2. In this log A occurs 10 times, while D occurs 4 times. We manually select A as a source node. Consequently, it has an in-degree of 0. The number of occurrences of A corresponds to the out-degree of A, which is 4 (to D) plus 6 (to E) equals 10. Similarly, the number of occurrences of D corresponds to the in-degree and the out-degree of D. In this way, creating a business process model that meets the process graph rule for all activities, produces Fig. 2(a). However, Figs. 2(b)–2(d) are also models that meet this rule. While Fig. 2(c) is the model that matches the original log from Table 1 best, the process graph rule alone is not enough to determine this.

As the example shows, it is often possible to create more than one model that meets the criteria. Therefore, additional measurements from the event log are required in order to select the best model. To this end, the correlation miner creates two matrices, the *Precede/Succeed matrix* and the *Duration matrix*. Each



(a) Alternative 1 (b) Alternative 2

(c) Alternative 3 (d) Alternative 4

Fig. 2. Feasible process models from the example log.

score in the Precede/Succeed matrix, $P/S_{i,j}$, indicates the fraction of events referring to the activity i that have occurred before events referring to the activity j. If this score is high, it is more likely that there is an edge from i to j. Each score in the Duration matrix, $D_{i,j}$, indicates the average time or standard deviation (we will experiment with both alternatives) of the time difference between the events referring to the activity i and the events referring to the activity j. If this score is low, it is more likely that there is an edge from i to j. In the remainder of this section, we explain how these matrices are computed, how they are used to construct the linear programming problem and, ultimately how the business process model is constructed.

3.1. *Precede/Succeed matrix*

The first step in the first phase of the correlation miner is to calculate the Precede/Succeed matrix. This matrix is a square matrix of order n, with $n = |\mathcal{A}|$. Let i represent a row in the matrix as well as an activity from $\mathcal{A}$ and let j represent a column and an activity. A value $P/S_{i,j}$ in the matrix represents the fraction of events from i that occurred before events from j and is computed as follows. $\Omega_{i,j} = \{(e, f) \,|\, e \in E_i \wedge f \in E_j\}$ is the set of all pairs of occurrences of events referring to activity i and events referring to activity j. Furthermore, given events e and f, let $\mathcal{B}(e, f)$ be a function that is 1 if $t_e < t_f$ and 0 otherwise.

$$P/S_{i,j} = \frac{\sum_{(e,f)\in\Omega_{i,j}} \mathcal{B}(e, f)}{|\Omega_{i,j}|}.$$

Returning to our example (Table 2), $P/S_{C,E}$ can be computed as follows. $\Omega_{C,E}$ contains 30 elements since $|E_C| = 3$ and $|E_E| = 10$. $\sum_{(e,f)\in\Omega_{C,E}} \mathcal{B}(e, f) = 17$, because in 17 cases C occurred before E. Therefore, $P/S_{C,E} \approx 0.57$.

Similarly, we can calculate the value for each pair of activities in a given event log. Figure 3 illustrates the matrix that is calculated based on the example log.

For performance reasons, we can remove some pairs so as not to include them for consideration during the next two steps of the algorithm. This can be done via a threshold filter. The goal of the threshold filter is to remove pairs of activities that have a very low probability of forming an edge. For example, in Fig. 3, we have $P/S_{A,C} = 0.97$ and $P/S_{C,A} = 0.03$. These values indicate that it is likely that there is an edge from A to C, but no edge from C to A. The challenge is to select a

$$P/S = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} 0.00 & 0.47 & 0.97 & 0.73 & 0.88 \\ 0.53 & 0.00 & 1.00 & 0.67 & 0.90 \\ 0.03 & 0.00 & 0.00 & 0.33 & 0.57 \\ 0.26 & 0.33 & 0.67 & 0.00 & 0.70 \\ 0.12 & 0.10 & 0.43 & 0.30 & 0.00 \end{pmatrix} \end{array}$$

Fig. 3. P/S matrix for the example log.

threshold that removes as many pairs as possible in order to increase performance, but to prevent false positives, because pairs that are removed at this stage will never become edges in the business process model.

3.2. *Duration matrix*

The second step in the first phase of the correlation miner is to generate the Duration matrix. This matrix is a square matrix of order n, with $n = |\mathcal{A}|$. Let i represent a row in the matrix as well as an activity from $\mathcal{A}$ and let j represent a column and an activity. A value $D_{i,j}$ indicates the average time difference or the standard deviation of the time difference between the events referring to activity i and the events referring to activity j.

In order to calculate the value of $D_{i,j}$, we need to have a *mapping* between the elements of E_i and the elements of E_j. The idea behind this mapping is that it maps events that belong to the same case. We compute the mapping, by relating the events in such a way that the variance in the time difference is as low as possible, because we argue that the time difference between events from the same case have a more constant probability distribution than the time difference between events from different cases. This principle is not likely to yield a perfect mapping, in the sense that events are mapped if and only if they belong to the same case. However, a perfect mapping is not necessary. We just need a mapping that has a time distribution that is close enough to the perfect mapping.

Formally, let $\Omega_{i,j}$ be defined as in Sec. 3.1. Then we compute the mapping $M_{i,j} \subseteq \Omega i, j$ that satisfies the following constraints.

$$\forall\,(e,f) \in M_{i,j} : \exists\,(e,f') \in M_{i,j} \Rightarrow f = f'$$

$$\forall\,(e,f) \in M_{i,j} : \exists\,(e',f) \in M_{i,j} \Rightarrow e = e'$$

$$\forall\,(e,f) \in M_{i,j} : t_e < t_f.$$

These constraints state that each event can be mapped at most once and that each event from E_i must be mapped to an event from E_j that occurs at a later point in time. In addition, the mapping must maximize its size (i.e. as many events must be mapped as possible), while minimizing the standard deviation of the time difference between the mapped events. With this mapping, we can compute $D_{i,j}$ as either the mean $\overline{\Delta M_{i,j}}$ or the standard deviation $\sigma_{\Delta M i,j}$.

$|Mi,j|$ must be maximized

$\overline{\Delta M_{i,j}}$ must be minimized, where $\Delta M_{i,j} = \{t_f - t_e \,|\, (e,f) \in M_{i,j}\}$.

With this mapping, we can compute $D_{i,j}$ as either the mean $\overline{\Delta M_{i,j}}$ or the standard deviation $\sigma_{\Delta M i,j}$. Returning to our example (Table 2), $D_{D,C}$ can be computed as follows. Figure 4 shows the mapping from events E_D to events E_C that satisfies the constraints above. The average time difference between the mapped events is

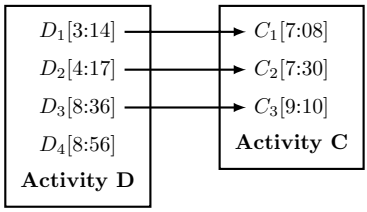


Fig. 4. Mapping of E_D to E_C.

$$D = \begin{array}{c} \\ A \\ B \\ C \\ D \\ E \end{array} \begin{array}{c} \begin{array}{ccccc} A & B & C & D & E \end{array} \\ \begin{pmatrix} 0 & 74 & 109 & 108 & 220 \\ 54 & 0 & 214 & 164 & 69 \\ 6 & 0 & 0 & 87 & 42 \\ 0 & 106 & 150 & 0 & 124 \\ 0 & 102 & 62 & 96 & 0 \end{pmatrix} \end{array}$$

Fig. 5. Duration matrix of example log.

approximately 153 sec. Similarly, we can calculate the value for each pair of activities in a given event log. Figure 5 illustrates the matrix that is calculated based on the example log. Note that the time difference in the matrix is slightly different from the time difference that we computed above. This is because we implemented an efficient (greedy) algorithm to compute the mapping, which does not always produce the mapping that satisfies all constraints. In particular, it does not always return the mapping with the lowest standard deviation, but rather the mapping with a local minimum. Although a lower score of $D_{i,j}$ indicates that it is more likely that there is an edge from i to j, a score of 0 means it is not possible to have such an edge, since we assume that events cannot happen at the same time in the event log.

It is worth mentioning that each activity in an event log may generate multiple events for one case in various transition statuses such as started, suspended, resumed and completed. In other words, one can argue that activities have a duration that is reflected in the event log. However, the duration matrix calculates the duration between the occurrence of the two activities and not the duration of each activity. Taking the transition status into consideration, the duration matrix can be extended in such a way that it calculates the time difference between the completion (or termination) of the preceding activity and the start of the succeeding activity.

3.3. *Business process model construction*

The final step in the first phase of the correlation miner is to generate the business process model. We first explain how all the possible business process models can be generated, taking the process graph rule into account. Then, we explain how the best business process model can be selected from all the possible business process models, using the Precede/Succeed matrix (Sec. 3.1) and the Duration matrix (Sec. 3.2).

In order to construct feasible models, we formulate our problem as an *Integer Linear Programming* (ILP) problem. Each possible edge (i, j) between activities becomes a variable x_{ij} of our ILP problem. The value that is assigned to each variable, x_{ij}, indicates the frequency with which an event for activity i is directly followed by an event from activity j in a case. Therefore, this value can be seen as an edge weight for the edge that connects the node that refers to activity i to the

node that refers to activity j. Let $\mathcal{X}$ denote the set that contains the introduced variables. Also, let $\mathcal{A}_s$ be the set of activities that we consider as start activities and $\mathcal{A}_e$ be the set of activities that we consider as end activities. We can reduce the problem space, by removing all the following variables:

- x_{ij} that represent unlikely edges because of $P/S_{i,j}$ or $D_{i,j}$ thresholds;
- $x_{is}, s \in \mathcal{A}_s$, which represent incoming edges to start activities; and
- $x_{ei}, e \in \mathcal{A}_e$, which represent outgoing edges from end activities.

Subsequently, for the resulting set of variables $\mathcal{X}$, we can formulate the constraints of our ILP problem as follows. The lower bound for each variable is 0, while the upper bound for each variable is $\min(|E_i|, |E_j|)$ because these are the minimum and the maximum number of cases that can flow on the edges. The sum of the number of cases on the incoming edges of an activity a must be equal to the number of times an event E_a occurs for that activity. Similarly, the sum of the number of cases on the outgoing edges of an activity a must be equal to the number of times an event E_a occurs for that activity. Consequently, the constraints become:

$$x_{ij} \geq 0 \qquad \text{for each } a \notin \mathcal{A}_s : \sum_{x_{ia} \in \mathcal{X}} x_{ia} = |E_a|$$

$$x_{ij} \leq \min(|E_i|, |E_j|) \quad \text{for each } a \notin A_e : \sum_{x_{ai} \in \mathcal{X}} x_{ai} = |E_a|.$$

Returning to the example of the log from Table 2, we can formulate the following constraints: $x_{AE} \geq 0$ and $x_{AE} \leq 10$. Similarly, $x_{AB} \geq 0$ and $x_{AB} \leq 3$. Also, $x_{AE} + x_{BE} + x_{CE} + x_{DE} = 10$ and $x_{AB} + x_{AC} + x_{AD} + x_{AE} = 10$. In addition, $x_{CB} + x_{CD} + x_{CE} = 3$, and $x_{AC} + x_{BC} + x_{DC} = 3$. Note that since x_{CA} represents an incoming edge to a start node A and x_{EC} represents an outgoing edge from an end node E, these two variables have been removed and, consequently, are ignored in these constraints.

By applying these constraints, we can construct all feasible business process models. Now in the second part of this section, our goal is to select the model, from all feasible ones, that best represents the event log behavior. For this purpose, we employ matrices that we computed in the first two steps in order to formulate an *objective function* to eventually select the *optimal* model out of all the feasible ones using ILP principles.

The matrices from steps 1 and 2 provide some evidence to suggest that if the activity pair (i, j) contains a high value for $P/S_{i,j}$ and a low value for $D_{i,j}$, these two activities may have a higher chance to form an edge. Based on this statement, our objective is to select the model that has a higher cumulative value of P/Ss and a lower cumulative value of Ds for the edges. To achieve this objective, we define a *coefficient* for each variable x_{ij}, in our ILP in order to consider these values. Since we are interested in a high value of $P/S_{i,j}$ and a low value of $D_{i,j}$, we define an *edge ratio* as $\frac{D_{i,j}}{P/S_{i,j}}$ for each variable x_{ij} that must be minimized. Note that we need to keep the sum of edge ratios for each edge independent of the assigned value of the

referring variables and count the edge ratio if there is *an* edge, irrespective of the weight of that edge. Thus, ideally, each x_{ij} is divided by its value and set to 0 if $x_{ij} = 0$ in order to eliminate their influence. However, this is not possible in an ILP. Therefore, in order to reduce this influence, we divide each variable by its upper bound. In conclusion, we formulate the objective function for our ILP as follows:

$$\mathcal{C}_{ij} = \frac{D_{i,j}}{P/S_{i,j}} \cdot \frac{1}{\min(|E_i|, |E_j|)}$$

$$\text{minimize} \sum_{x_{ij} \in \mathcal{X}} \mathcal{C}_{ij} \cdot x_{ij}.$$

Note that one can envision different objective functions, which still reflect the requirement of high $P/S_{i,j}$ and low $D_{i,j}$. Since it is highly recommended in the area of design science to explain alternatives, we introduce two more alternatives. In the first alternative, the Proceed/Succeed matrix has a higher influence in finding the optimal solution. We prioritize this matrix by squaring its values in the edge ratio. Analogously, in the second alternative, the Duration matrix has a higher influence by squaring its values. Having employed these three alternatives in our experiments, on average, the proposed objective function produced better results.

Moreover, the main reason that we have not defined our optimization problem as a multi-objective optimization is that the Proceed/Succeed matrix and the Duration matrix are not conflicting and, because of that, we only defined the edge ratio (as it covers both) to be minimized. Multi-objective optimization can be applied, most of the time, where optimal decisions need to be taken in the presence of trade-offs between two or more "possibly conflicting" objectives.

Returning to the example log (Table 2), by considering the constraints, multiple models can be constructed as shown on Fig. 2. Now, we can calculate the objective function for each of these business process models and select the one that has a lowest value as an output of our algorithm. The value for the business process model from Fig. 2(c) is the lowest. Therefore, we select the business process model in Fig. 2(c) as the final result for the first phase of the correlation miner.

The algorithm that we described above may return a model that contains cycles. However, since the correlation miner only deals with acyclic processes, these cycles should be removed. Therefore, the correlation miner employs Johnson's algorithm[14] to capture all the edges that are involved in cycles in the model. It then calculates the edge ratio for these edges and removes the edge with highest edge ratio. The algorithm then reruns the third step repetitively until a model without cycles is captured.

4. Evaluation of Phase I

This section presents an evaluation of the first phase of the correlation miner. It first presents the setup of the evaluation in Sec. 4.1. Then it presents two evaluations,

one using synthetic event logs (Sec. 4.2) and one using a modified version of a real-world event log (Sec. 4.3).

4.1. *Evaluation setup for Phase* I

In order to conduct our evaluation, we have implemented the first phase of the correlation miner as a Java application. Since the algorithm uses the Gurobi ILP Solver,[15] which is a commercial software, one needs to install this software on one's machine with a proper license. For the interested reader we provide on-line access to the algorithm,[a] which can only be used for academic purposes. Moreover, we are currently investigating the licensing issue in order to implement the algorithm as a ProM[16] plug-in.

Note that although in Sec. 3.1 we have suggested to set a threshold filter for P/S matrix in order to remove less probable values from the calculation (which indeed leads to higher performance), we do not apply this filter in our evaluation. The main reason that we neglected the threshold is we aimed at finding the most optimal solution in the current version of the work. However, in the implemented prototype, there is a scrollbar which allows users to set the threshold manually. Indeed, in our future work, we will extend the correlation miner in such a way that the threshold will be aligned automatically according to some criteria of the input event log.

In addition, as discussed in Sec. 3.2, the correlation miner can construct the values for the duration matrix according to the average time difference or the standard deviation of the time difference between events referring to each pair of activities. We evaluated the correlation miner based on both the techniques and realized that, in general, using the duration matrix with average values often works better than using the duration matrix with standard deviation values. Consequently, the presented results in this section are based on the average time difference values for the duration matrix. Same with the previous note, in the implemented prototype of the correlation miner, one can choose to use either of the two methods.

Figure 6 depicts the procedure for evaluating our algorithm. For the evaluation with synthetic logs, we first create synthetic business process models with particular properties, in order to investigate the effect that these properties have on our algorithm. This will be explained in detail in Sec. 4.2. Subsequently (step 1(a)), we generate synthetic logs for these models, using the BIMP simulator.[b] For the evaluation with real-world logs, we take a log from practice. Subsequently (step 1(b)), we generate a business process model for that log. This will be explained in detail in Sec. 4.3. Now that we have both a log and a model, we remove the case identifiers from the log (step 2) to generate a log that can be used for correlation mining and (step 3) we apply the first phase of the correlation miner. Finally (step 4), we assess the quality of the mined model.

[a] http://is.ieis.tue.nl/research/correlation.
[b] http://bimp.cs.ut.ee/.

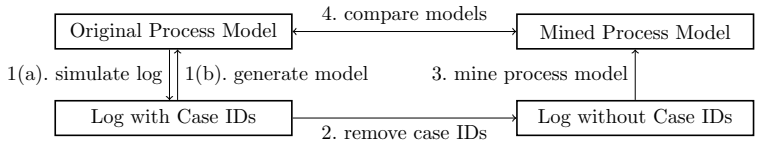


Fig. 6. Evaluation procedure of mined process models.

In order to measure the quality of the mined models, we use *precision* and *recall*. Precision measures the fraction of edges in the mined model that are correct, i.e. are also in the original business process model. Recall measures the fraction of correct edges that have been found. These measures are defined as follows. Let TP be the set of edges that exist in the mined model and also in the original; FN be the set of edges that do not exist in the mined model but do exist in the original model; and FP be the set of edges that exist in the mined model but do not exist in the original model. Then precision and recall are defined as:

$$\text{precision} = \frac{TP}{TP + FP} \quad \text{recall} = \frac{TP}{TP + FN}.$$

Returning to Fig. 2 and assuming that Fig. 2(b) is the mined model and Fig. 2(c) is the original model, we can calculate precision and recall as follows. TP for these two models is 4, including AB, AD, CE and DE; FN is 2, including AC and BE; and, finally, FP is again 2, including BC and AE. Consequently, precision and recall for Fig. 2(b) are approximately 0.67.

In related work, fitness and appropriateness have been introduced[4] as measures to evaluate the quality of a mined business process model. However, these measures evaluate the quality of a business process model as it is compared to a log. We evaluate the quality of the business process model as it is compared to another business process model; the business process model that should have been returned. By doing so, we can get more meaningful results. However, clearly this is only possible when such a business process model exists. In other situations fitness and appropriateness should be used.

4.2. *Synthetic event logs for Phase* I

We evaluated the first phase of the correlation miner using over 250 synthetic event logs with different properties. We varied the properties that we assumed would have an effect on the quality of the mined business process model. Specifically, we generated logs:

- using both structured and unstructured business process models (according to the criteria in Ref. 17) and using business process models with different numbers of branches because these factors influence model complexity and we assume that the more complex models are more difficult to mine;

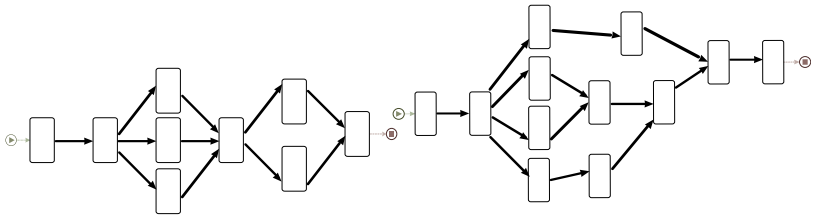
- that contained different numbers of cases because a higher number of cases provides more data to mine from and should, therefore, produce more accurate models;
- with different inter-arrival times and activity durations because time properties play an important role in our algorithm and should, therefore, have an impact on the quality of the result.

In total, we used six business process models to generate our logs. It is worth mentioning that in all generated logs, we assumed that (i) all the executions of the tasks were recorded successfully; and there were no noisy, missing or incomplete events or cases in the logs. For reasons of space, we do not present all the results, but rather the results for the model that produced the best results (Fig. 7(a)) and the model that produced the worst results (Fig. 7(b)). These models primarily vary with respect to whether they are structured or unstructured. The maximum number of branches does not vary much because our evaluation showed that the number of branches did not play a substantial role in the quality of the results.

Table 3 presents the results for mining logs that were generated from the model from Fig. 7(a). We generated different logs from this model in such a way that the duration of activities and the inter-arrival times were either:

- distributed uniformly with a median that was selected randomly from the interval $(1, 100]$;
- distributed normally with a mean that was selected randomly from the interval $(1, 100]$ and a standard deviation that was selected randomly from the interval of $\frac{1}{5}$ to $\frac{1}{7}$ of the selected mean;
- distributed exponentially with a mean that was selected randomly from the interval $(1, 100]$.

For each of these three distributions, we produced an event log with 200, 2,000, and 10,000 cases (which is the maximum number of cases that can be produced by the BIMP Simulator). In two experiments, the correlation miner was not able to generate a business process model because after removing some variables (e.g. by removing cycles or the P/S threshold filter), it was unable to solve the model with ILP, as the model became infeasible with regard to its constraints. The results in



(a) (b)

Fig. 7. Evaluation business process models.

Table 3. Results of using the first phase of the correlation miner for Fig. 7(a).

Distribution	Uniform			Exponential			Normal		
Number of Cases	200	2000	10,000	200	2000	10,000	200	2000	10,000
Model recall	1.00	1.00	1.00	n/a	1.00	1.00	n/a	1.00	1.00
Model precision	1.00	1.00	1.00	n/a	1.00	1.00	n/a	1.00	1.00

this table show that the correlation miner can discover a simple structured model perfectly, provided that there are sufficiently many cases in the log.

Table 4 presents an overview of the results for mining logs that were generated from the model from Fig. 7(b). Experiments on these logs have been conducted with activity duration distributions that were selected in the same manner as for the model from Fig. 7(a), but with inter-arrival times that were either selected in the same manner or from an interval with a longer duration ($(200, 300]$). We also experimented with the ranges $(1, 10]$ and $[10, 10]$. However, these did not lead to substantially different results. Therefore, due to space restrictions, we do not publish those results here. For each of these combinations of probability distributions, we produced an event log with 100, 1,000, and 10,000 cases.

Under most conditions, the results for this more complex unstructured model are worse than for the structured model. The main reason that the correlation miner produces better results for structured model is as follows:

- If a structured model contains an activity with n outgoing flows, there will be, for sure, another activity with n incoming flows; however,
- if an unstructured model contains an activity with n outgoing flows, the model may not contain any activity with n incoming flows and, consequently, solving the structured model would be easier for our ILP.

For example, the model shown in Fig. 7(a) contains the activity with three outgoing flows and the activity with three incoming flows while the model illustrated in Fig. 7(b) contains the activity with four outgoing flows but no activity with four incoming flows. Moreover, the perfect result shown in Table 3 can be explained by the fact that its underlying structured model only contains nine activities, where six of which have only one incoming flow and one outgoing flow.

The other interesting finding to emerge from Table 4 is that if the inter-arrival time between cases is higher than the service time of activities, this leads to substantially better results. Comparing the two sets of results in this table shows that in the experiments with higher inter-arrival time, the results for recall and precision noticeably increased on average by 27% and 42%, respectively. This result can be explained, because if the inter-arrival time is the same as the service time, cases are more 'intertwined' in the log, such that the correlation miner has more difficulties telling them apart based on their timing properties (which other mining algorithms can do based on case identifiers).

Table 4. Results of using the first phase of the correlation miner for Fig. 7(b).

	Distribution					
	Uniform			Exponential		
Number of cases	100	1,000	10,000	100	1,000	10,000
Inter-arrival time	[1–100]					
Model recall	0.57	0.79	0.64	0.57	0.57	0.71
Model precision	0.42	0.65	0.47	0.42	0.42	0.56
Inter-arrival time	[200–300]					
Model recall	0.75	0.93	1.00	0.93	0.71	1.00
Model precision	0.86	0.87	1.00	0.81	0.59	1.00

Also, as expected, and in line with the findings from Table 3, a higher number of cases in the log leads to better models in most cases. However, in the experiments in which activities were distributed exponentially, our algorithm did not find a significant difference between the event logs that originally contained 100 and 1,000 cases.

Hence, based on these results, we can conclude that the correlation miner is applicable to discover process models from event logs without case identifiers, especially for structured models and when the inter-arrival time between cases is higher than the service time of activities and the number of cases in the log is sufficiently high.

It is worth mentioning that this phase of the correlation miner solves a mixed-integer linear programming (MILP) problem and, therefore, the time complexity of this phase will be weakly polynomial.[18] The scatter plot in Fig. 8 shows the elapsed time for the first phase of the correlation miner to produce the final process models in our experiments. For this set of experiments, we used 57 event logs where the number of cases in each three logs increased by 500. Therefore, the three smallest logs contained 500 cases and the three largest log contained 9,500 cases. Each plot in Fig. 8 represents the average elapsed time for each three event logs with the same number of cases. The results shown in this graph seem to be consistent with the mentioned time-complexity. On average, the elapsed time for Phase I of the correlation miner using the event logs with 500 cases was 0.1 sec, using the event logs with 5,000 cases was 10.3 sec and using the event logs with 9,500 cases was 43.2 sec.

4.3. *Real-world event log for Phase* I

We also evaluated the algorithm based on a real-world event log. To this end, we used a modified version of the log from the BPI Challenge of 2012[19] taken from a Dutch Financial Institute. The underlying process represented in this real-world event log is an application process for a personal loan or overdraft. First, we mined a business process model from this event log using Disco.[9] We then removed all loops in order to make it acyclic because our algorithm only deals with acyclic models at this stage. For the same reason, we then removed the cases from the log in which loops appeared. The resulting version of this real-world event log contained

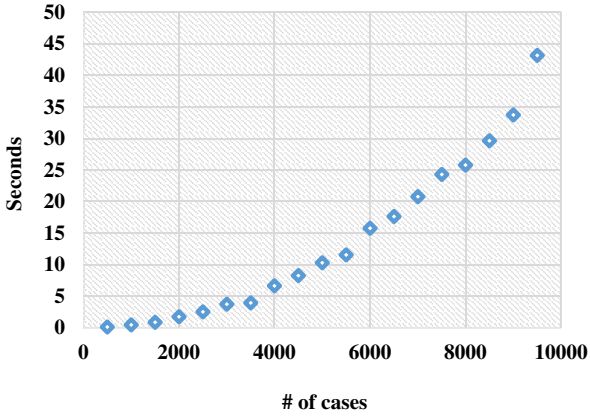


Fig. 8. The elapsed time for Phase I of the correlation miner.

13 activities, 70,425 events and 11,647 cases. Figure 9 depicts the most abstract version of this business process model which has been mined by the Disco process mining tool[9] using 0% of variation paths.

In accordance with the evaluation procedure described in Sec. 4.1, we then removed all case identifiers from the log, discovered process models from the log using the correlation miner and, subsequently, evaluated the quality of the discovered model. The results obtained from this experiment show a precision of 85% and a recall of 63%. The lower result for recall can be explained by the fact that our algorithm seeks to find edges with higher frequencies, while we used the Disco model that included all possible edges that could be mined from the log not just the ones with the higher frequency. Normally speaking, one would not be interested in the Disco model that contains all the edges because this model also includes edges that represent exceptional situations. Indeed, if we remove edges with lower frequency (i.e. <50) from the original model our recall increases to 68% as precision decreases to 79%.

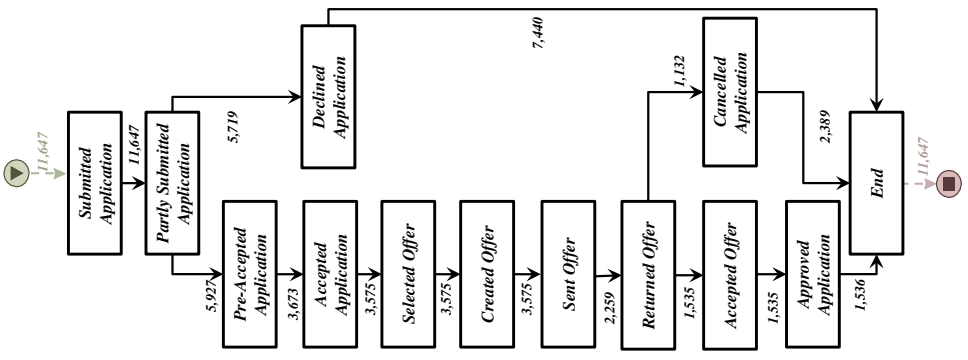


Fig. 9. The most abstract version of the used process model of the real-world log.

Based on these results, we claim that the correlation miner is also applicable for mining real-world event logs.

5. Phase II: Reconstruction of Cases from Event Logs

The first phase of the correlation miner (cf. Sec. 3) allows us to construct a process model from an event log without case identifiers. However, it does not allow us to reconstruct the cases themselves; it does not reveal which events belong to the same case. This section fills that gap by presenting the second phase of the correlation miner which aims to accurately determine which events belong to the same case.

The algorithm proposed in this section works by interpreting the log as a set of binary relations between events. A relation from event e to event f represents that event e is directly followed by event f in the same case. For example, considering the event log shown in Table 1, Case 1 contains:

- a binary relation between the event referring to activity A that happened at 00:20 and the event referring to activity B that happened at 02:04, depicted by $(A_1[00{:}20], B_1[02{:}04])$, and
- a binary relation between the event referring to activity B that happened at 02:04 and the event referring to activity E that happened at 02:32, depicted by $(B_1[02{:}04], E_1[02{:}32])$.

Therefore, Case 1 can be denoted as $\{(A_1[00{:}20], B_1[02{:}04]), (B_1[02{:}04], E_1[02{:}32])\}$. Alternatively, the relations can be represented in a graph. For example, the graph that represents Case 1 is illustrated in Fig. 10.

Against this background, the goal of the second phase of the correlation miner is to find the binary relations that determine the events that directly follow each other in a case. We achieve that goal by casting the problem into a Quadratic Programming (QP) problem, in which we define the constraints that must hold for the binary relations, and an optimization function that defines what the 'best' set of binary relation is.

In the remainder of this section, we describe how the mentioned constraints and the objective function are formulated, in Secs. 5.1 and 5.2, respectively.

5.1. *Constraints*

We present a number of constraints that must hold for relations between events. These constraints help us to reduce the space of possible relations that must be

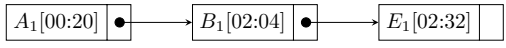


Fig. 10. Graph representation case 1 from the example event log.

explored. In particular, the constraints that must hold are that:

- two events can only be related if their activities are also related in the mined process model; and
- an event can be directly preceded or directly followed by, at most, one other event.

Consider a discovered process model that includes an edge from a node referring to activity i to a node referring to activity j. There is a possibility that $e_i \in E_i$ was directly followed by $e_j \in E_j$ and, therefore, the referring binary relation (e_i, e_j) must exist in one of the execution cases. In general, each binary relation, e.g. (e_x, e_y), introduces a binary variable, $r_{e_x e_y}$, in our QP problem, which indicates whether the referring binary relation exists in a case or not. Clearly, we can reduce the problem space by removing all variables $r_{e_x e_y}$ that represents a binary relation in which e_x has occurred after e_y. Let $G = (\mathcal{V}, \mathcal{E}, \omega)$ be a discovered process model and $\mathcal{R}$ denote the set that contains all the introduced variables. We formulate the first set of constraints of our QP problem as follows:

$$\text{for each edge } (x, y) \in \mathcal{E} : \sum_{r_{e_x e_y} \in \mathcal{R}} r_{e_x e_y} = \omega(x, y).$$

Returning to the example of the log from Table 2 and the discovered process model shown in Fig. 2(c), we can, for example, conclude that there must be three binary relations between the events referring to activity C and the events referring to activity E out of 17 possible relations as shown in Fig. 11. Therefore, we can formulate the following constraint for the edge between the node referring to activity C and the node referring to activity E.

$$r_{C_1 E_4} + \cdots + r_{C_1 E_{10}} + r_{C_2 E_5} + \cdots + r_{C_2 E_{10}} + r_{C_3 E_8} + \cdots + r_{C_3 E_{10}} = \omega(C, E) = 3.$$

Obviously, if the value of $r_{C_1 E_4}$ is 1 (i.e. C_1 and E_4 belong to the same case), no other binary relation can contain C_1 as a source event and E_4 as a target event. Therefore, to apply this restriction, we introduce two additional sets of binary variables for each event to specify whether the referring event has already been selected as (i) a source event, and as (ii) a target event. Let $\mathcal{S}_{e_a e_x}$ denote the set that contains all the binary variables referring the usage of e_a as a source event, and $\mathcal{T}_{e_x e_a}$ denote the set that includes all the binary variables referring the usage of e_a as a target event. In addition, let E_s be the set of events referring to the start activities and E_e be the set of activities referring to the end activities. We can reduce the problem space by removing the variables:

- $t_{e_x e_s} \in \mathcal{T}_{e_x e_s}$ such that $e_s \in E_s$, which represent the usage of events referring to start activities as target events; and
- $s_{e_e e_x} \in \mathcal{S}_{e_e e_x}$ such that $e_e \in E_e$, which represent the usage of events referring to end activities as source events.

$$r_{C_1E_4} + \ldots + r_{C_1E_{10}} + r_{C_2E_5} + \ldots + r_{C_2E_{10}} + r_{C_3E_8} + \ldots + r_{C_3E_{10}} = \omega(C, E) = 3$$

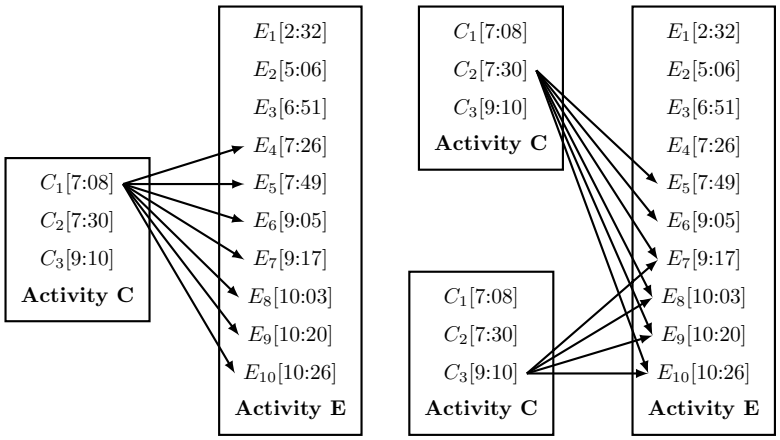


Fig. 11. All possible binary relations between events referring to activity C and events referring to activity E.

Subsequently, for the resulting sets of binary variables, we can formulate the constraints of our QP problem as follows:

$$\text{for each event } e_a \in E_a \wedge E_a \neq E_s : \sum_{t_{e_x e_a} \in \mathcal{T}_{e_x e_a}} t_{e_x e_a} = 1$$

$$\text{for each event } e_a \in E_a \wedge E_a \neq E_e : \sum_{s_{e_a e_x} \in \mathcal{S}_{e_a e_x}} s_{e_a e_x} = 1.$$

Returning to the example of the log from Table 2 and the discovered process model shown in Fig. 2(c), we can, for example, conclude that C_1 must be a target event for only one of the nine possible relations as shown on the left-hand side of Fig. 12, and similarly, it is a source event for only one of the seven possible relations as shown on the right-hand side of Fig. 12. Therefore, we can formulate the following constraints for this event:

- $t_{A_1C_1} + t_{A_2C_1} + t_{A_3C_1} + t_{A_4C_1} + t_{A_5C_1} + t_{A_6C_1} + t_{A_7C_1} + t_{A_8C_1} + t_{A_9C_1} = 1.$
- $s_{C_1E_4} + s_{C_1E_5} + s_{C_1E_6} + s_{C_1E_7} + s_{C_1E_8} + s_{C_1E_9} + s_{C_1E_{10}} = 1.$

Accordingly, the value of $r_{C_1E_4}$ indicates the values of $s_{C_1E_4}$ and $t_{C_1E_4}$. In other words, if C_1 and E_4 belong to the same case, there is only one binary relation in which C_1 acts as a source event (i.e. $s_{C_1E_4}$), and only one binary relation in which E_4 acts as a target event (i.e. $t_{C_1E_4}$). Conversely, if these two events belong to different cases, the values of $s_{C_1E_4}$ and $t_{C_1E_4}$ are 0. In general, we can state that the value for variable $r_{e_xe_y} \in \mathcal{R}$ will be 1, if and only if there exists:

- a variable $s_{e_xe_y} \in \mathcal{S}_{e_xe_y}$ such that $s_{e_xe_y} = 1$, and
- a variable $t_{e_xe_y} \in \mathcal{T}_{e_xe_y}$, such that $t_{e_xe_y} = 1$.

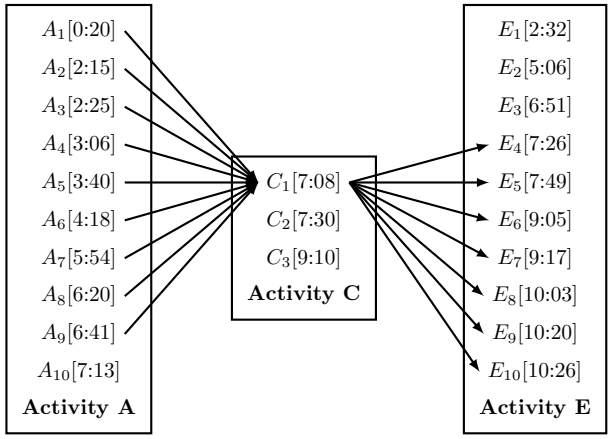


Fig. 12. All possible situation for the first event of C.

To ensure that the value of a binary relation is the same as the values of the referring source and target usage variable, we formulate the following constraint in our QP problem:

$$\text{for each two events } e_a \in E_a \quad \text{and} \quad e_b \in E_b : r_{e_a e_b} = s_{e_a e_b} = t_{e_a e_b}.$$

Finally, note that these constraints are formulated according to the process models that are discovered in the first phase of the correlation miner. Consequently, if a discovered process model does not precisely depict the behavior of an event log, these constraints may result in an infeasible QP problem. In order to mitigate this threat, we allow the constraints to be violated according to a set of predefined penalty costs. For instance, the constraints referring to the discovered process models (e.g. the first set of constraints) have the lowest penalty costs (i.e. they can be easily violated). This decision can be explained by the fact that — in practice — there is no ground truth process model for an event log since event logs only contain a fraction of possible cases (i.e. many other cases can possibly occur) and also event logs cannot show impossible cases (i.e. cases that cannot occur). In contrast with this penalty cost, the constraints referring to the usage of events have the highest penalty cost since, as noted before, events can be directly preceded or directly followed by, at most, one other event.

5.2. *Objective function*

By applying the constraints in Sec. 5.1, we can construct all feasible sets of 'directly follows' relations between events from the same case. However, we aim to select the best set. For this purpose, we further employ the time difference between the source and the target event of each binary relation in order to formulate an objective function to pick the optimal set of relations.

More specifically, this objective function attempts to form a subset of all possible binary relations in such a way that the variation in the time difference among the selected binary relations is as low as possible. This objective function is designed to comply with the assumption that time differences between binary relations that belong to the same cases are more likely to follow a certain probability distribution, than time differences between binary relations from different cases.

The most obvious coefficient that can be used to achieve the mentioned objective is the actual time difference between the source and the destination event for each binary relation. However, to precisely define a coefficient for the binary relations, we further adopt the idea of the correlation coefficient suggested by Bradley Efron,[20] which is based on the variance and standard error of an unknown distribution for an arbitrary sample. Based on this idea, we define error as the squared difference between the observed data and the expected value. Although, in general, the mean of a sample can be used as an (expected) parameter that minimizes the total error, in the context of this study, we attempt to minimize the total error by minimizing the squared difference between the source and target event for each binary relation. In other words, we expect that the variation between the source and the target event of binary relations is as low as possible. Consequently, we formulate the objective function for our QP problem as follows. Let O_{e_a} denote the timestamp at which event e_a has occurred.

$$\text{minimize} \sum_{r_{e_i e_j} \in \mathcal{R}} (O_{e_j} - O_{e_i})^2 \left(\frac{r_{e_i e_j} (O_{e_j} \cdot t_{e_i e_j} - O_{e_i} \cdot s_{e_i e_j})}{O_{e_j} - O_{e_i}} \right).$$

Same as Phase I, other objective functions can also be envisioned. In order to mitigate this issue, we introduce two more alternatives. In the first alternative, the objective function minimizes the time differences between binary relations and in the second alternative, it maximizes the time difference between the relations. Having employed these three objective functions in our experiments, on average, the one proposed objective function produced better results.

By way of illustration, let us consider the simple example given below. Table 5 shows the original event log, Table 6 exemplifies this event log without case identifiers and Fig. 2 presents the discovered process model using the first phase of the correlation miner.

Table 5. Simple event log with case identifiers.

Case	Activity	Timestamp
1	X	02:00
1	Y	04:00
1	Z	08:00
2	X	03:00
2	Y	05:00
2	Z	09:00

Table 6. Simple event log without case identifiers.

Activity	Timestamp
X	02:00
X	03:00
Y	04:00
Y	05:00
Z	08:00
Z	09:00

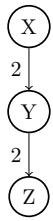


Fig. 13. Mined process model.

This section sets out to determine the cases in the original log (Table 5) by employing the event log without case identifiers (Table 6) and the discovered process model (Fig. 13). Looking closely at the events in the log without case identifiers, there are three possible sets of binary relations that are applicable for the discovered model. For brevity, we use the shortened form of the timestamps of the mentioned events within the square brackets. These three possible sets are:

- $\{(X_1[2], Y_1[4]), (Y_1[4], Z_1[8]), X_2[3], Y_2[5]), (Y_2[5], Z_2[9])\}$, which results in two execution cases: $\langle X_1, Y_1, Z_1\rangle$ and $\langle X_2, Y_2, Z_2\rangle$;
- $\{(X_1[2], Y_2[5]), (Y_2[5], Z_1[8]), X_2[3], Y_1[4]), (Y_1[4], Z_2[9])\}$, which results in two execution cases: $\langle X_1, Y_2, Z_1\rangle$ and $\langle X_2, Y_1, Z_2\rangle$; and
- $\{(X_1[2], Y_2[5]), (Y_2[5], Z_2[9]), X_2[3], Y_1[4]), (Y_1[4], Z_1[8])\}$, which results in two execution cases: $\langle X_1, Y_2, Z_2\rangle$ and $\langle X_2, Y_1, Z_1\rangle$.

Having identified these three sets of binary relations, we can calculate the objective function for each of them in order to select the one that has the lowest value as the best candidate. The value of the objective function for the first set is 40, for the second set is 44, and for the last set is 42. Therefore, the first set of execution cases has the lowest value and, thus, this set of binary relations will be selected to produce the event log with cases. As confirmed by the original log (Table 5), the selected set of binary relations best reveals the actual 'directly follows' relations among the events within the event log without case identifiers (Table 6). In the same way, we can compute the best set of binary relations among the events in Table 2 by employing the discovered process model shown in Fig. 2(c). This calculation results in exactly the same event log as shown in Fig. 1.

6. Evaluation of Phase II

This section presents the evaluation of the second phase of the correlation miner. Firstly, the evaluation setup is discussed in Sec. 6.1. Then two types of evaluations are presented, one using synthetic event logs (Sec. 6.2) and one using a real-world event log (Sec. 6.3).

6.1. *Evaluation setup for Phase* II

In order to conduct the evaluation of the second phase of the correlation miner, we have further extended the online version of the correlation miner by implementing the algorithm proposed in the previous section. The procedure for evaluating the new algorithm is shown in Fig. 14.

The first three steps in this procedure are similar to the first three steps for the evaluation procedure of the previous algorithm as shown in Fig. 6. However, this evaluation procedure goes on to reconstruct a log with case identifiers from the mined process model in step 4. Subsequently, we compare cases within the reconstructed event log with cases within the original event log in step 5.

In order to measure the quality of the reconstructed event log, we adapt the definition of precision and recall to be used for comparing the two event logs. In this context, precision measures the fraction of binary relations in the reconstructed log that is correct (i.e. events that also directly follow each other in a case in the original event log with case identifiers) and recall measures the fraction of correct binary relations that has been found. We further adapt the definitions of TP, FN and FP so that they are applicable for comparing the two event logs. For this purpose, we defined these elements as follows.

Let TP be the set of binary relations that exists in the reconstructed event log and also in the original event log; FN be the set of binary relations that does not exist in the reconstructed log but exists in the original log; and FP be the set of binary relations that exists in the reconstructed log but does not exist in the original log. Then, precision and recall can be calculated as explained in Sec. 4.

The main reason that we did not utilize fitness and appropriateness[4] as measures to evaluate the quality of a reconstructed event log is that these measures evaluate the quality of a mined business process model on the basis of an original event

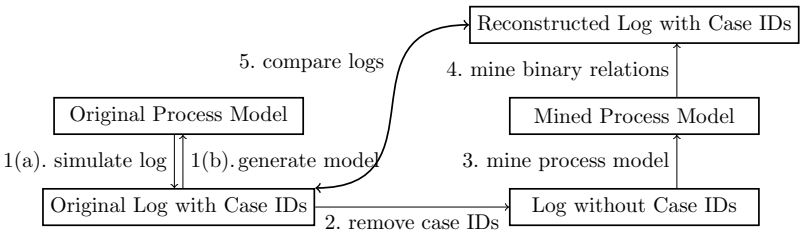


Fig. 14. Evaluation procedure of reconstructed event logs with cases.

log, while we evaluate the quality of a reconstructed event log on the basis of that original event log.

Returning to the example given in Table 6 and assuming that the original event log consists of $\{\langle X_1, Y_1, Z_1\rangle, \langle X_2, Y_2, Z_2\rangle\}$ and the reconstructed event log consists of $\{\langle X_1, Y_2, Z_2\rangle, \langle X_2, Y_1, Z_1\rangle\}$, we can calculate precision and recall as follows. TP for these two event logs is 2, including (Y_1, Z_1) and (Y_2, Z_2); FN is also 2, including (X_1, Y_1) and (X_2, Y_2); and, finally, FP is again 2, including (X_1, Y_2) and (X_2, Y_1). Consequently, both precision and recall for this example are 0.5.

6.2. *Synthetic event logs for Phase* II

We evaluated the proposed technique using the synthetic event logs that were introduced in Sec. 4.2. However, we decreased the number of cases in these event logs because, at this stage, the new algorithm has a high time and space complexity which makes it impractical for use with event logs with a large number of events. In future work, we will work on reducing the time complexity of the algorithm.

We present the results for the events logs that produced the best results and the event logs that produced the worst results.

Table 7 presents the results for mining logs that were generated from the model from Fig. 7(a). We generated different logs from this model by varying different properties, as explained in Sec. 4.2. The table shows both the precision and recall of the mined process model and the precision and recall of the (binary relations in the) reconstructed event log.

The table illustrates that the precision and recall for both the mined process model and the reconstructed event log are high. As shown in Sec. 4, since this model is structured (according to the criteria in Ref. 17), the correlation miner is generally able to generate a business process model with a very high value for recall and precision. Since the mined process model is utilized for mining the relations, this helps to correctly reconstruct the event log.

The 'n/a' values in the table are caused by the fact that the correlation miner was not able to complete the second phase for the exponentially distributed experiments with 25 cases. The reason for this was that the generated process model was less accurate, and this was due to the lack of an adequate number of cases in the log. However, in general, we conclude that the correlation miner can mine a simple structured model accurately, provided that there are sufficiently many cases in the log.

Table 7. Results of using the correlation miner for Fig. 7(a).

Distribution	Uniform			Normal			Exponential		
Number of cases	25	50	75	25	50	75	25	50	75
Process model recall	0.92	1.00	1.00	0.92	1.00	1.00	0.75	1.00	1.00
Log recall	0.89	1.00	1.00	0.83	1.00	1.00	n/a	1.00	1.00
Process model precision	0.85	1.00	1.00	0.85	1.00	1.00	0.75	1.00	1.00
Log precision	0.92	1.00	1.00	0.86	1.00	1.00	n/a	1.00	1.00

Table 8. Results of using the correlation miner for Fig. 7(b).

	Distribution								
	Uniform distribution								
Number of cases	25			50			75		
Random number	[1–10]	[1–100]	[10]	[1–10]	[1–100]	[10]	[1–10]	[1–100]	[10]
Model recall	0.79	0.86	0.86	0.86	0.86	0.86	0.93	0.93	0.93
Log recall	0.73	0.84	0.57	0.58	0.59	0.59	0.83	0.82	0.82
Model precision	0.69	0.80	0.75	0.80	0.75	0.75	0.87	0.87	0.87
Log precision	0.78	0.90	0.62	0.66	0.63	0.63	0.84	0.84	0.84
	Exponential distribution								
Model recall	0.93	0.86	1.00	0.93	0.93	1.00	0.93	0.93	0.86
Log recall	0.78	0.79	1.00	0.81	0.80	1.00	0.81	0.82	0.70
Model precision	0.87	0.75	1.00	0.87	0.87	1.00	0.87	0.87	0.86
Log precision	0.81	0.80	1.00	0.85	0.83	1.00	0.83	0.83	0.72

Table 8 presents an overview of the results for mining logs that were generated from the model shown in Fig. 7(b). The precision and recall for these models are the worst that we obtained. However, generally, they are still quite good at around 0.8 or even 0.9 for precision and recall. Again, there seems to be a strong relation between the quality of the mined process model and the quality of the reconstructed event log.

The results shown in this table have been produced by using the experiments in which the inter-arrival time between the cases have a longer duration than the service times of activities within an event log. We only present the results for this set of experiments because the correlation miner can only mine unstructured process models from event logs that contain a few cases if the inter-arrival time between these cases is sufficiently high (i.e. the cases are less intertwined in the event logs). As we discussed in Sec. 4, the results for more complex and unstructured process models are generally worse than for the structured ones and, consequently, we can infer that the quality of reconstructed event logs from unstructured models must be substantially lower than the quality of reconstructed event logs from structured ones. In Table 8, we can see that the results for event logs in which activities and the inter-arrival time between cases has exponential distributions are slightly better (9% on average) than event logs in which activities and inter-arrival time between cases have uniform distributions. Finally, the data shown in this table suggest that using different ranges in our experiments did not cause substantially different results since, on average, the difference between the ranges leads to less than 1% difference among the quality of the mined process models and approximately 4% difference among the quality of the reconstructed event logs. Table 9 summarizes the data presented in Table 8 by presenting the average value for each quality criterion.

Interestingly, what stands out in the presented results in Tables 7 and 8, as well as in all the other experiments that we have conducted, is the value of precision and recall for the reconstructed event log, when the value of precision and recall for

Table 9. Summary of data presented in Table 8.

Model recall	Recons. log recall	Model precision	Recons. log precision
0.89	0.76	0.82	0.78

a mined process model is 1. During our experiments, we never observed a situation where the value of the precision and recall for the mined process models is 1 while the value of these two quality criteria for their related reconstructed event logs is lower than 1.

Hence, based on these results, we can conclude that the correlation miner is applicable for mining logs without case identifiers, especially for structured models and when the inter-arrival time between cases is higher than the service time of activities and the number of cases in the log is high enough.

It is worth mentioning that this phase of the correlation miner solves a mixed-integer quadratic programming solver (MIQP) and, therefore, the time-complexity of this phase, in general, will be non-deterministic polynomial-time hard (NP-hard).[21] However, since the defined MIQP is convex, the solver that we used for the experiments (i.e. Gurobi solver[15]) tries to solve the problem in polynomial running time. Figure 15 shows the elapsed time for the second phase of the correlation miner to produce the final event log with case identifiers. For this experiment, we used 45 event logs where the number of cases in each three logs increased by 5. Therefore, the three smallest logs contained five cases and the three largest log contained 75 cases. Each plot in this chart represents the average elapsed time for the three event logs with the same number of cases. The results shown in this graph corroborate the theory that, in general, MIQP problems run in NP time; however, the employed solver attempts to (partially) solve the problems in weakly polynomial time. On average, the elapsed time for Phase II of the correlation miner using the three event logs with five cases was 0.02 sec, using the three event logs with 25 cases

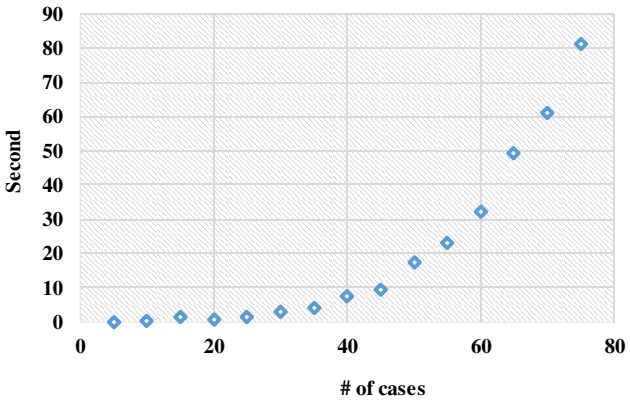


Fig. 15. The elapsed time of Phase II of the correlation miner.

was 1.38 sec, using the three event logs with 50 cases was 17.52 sec and using the three event logs with 75 cases was 81.11 sec.

6.3. *Real-world event log for Phase* II

We also evaluated the log reconstruction algorithm based on the real-world event log introduced in Sec. 4.3. However, to make this log usable for the algorithm, we reduced the size of the log by randomly selecting 90 cases. To factor out the effects of the random selection, we ran the experiment four times on four random selections. The results obtained from these experiments are illustrated in Table 10.

It is apparent from this table that recall and precision for the model reconstruction phase of the correlation miner are better than for its process discovery phase. In other words, although the quality of the mined process models is low — which is not surprising as we only selected 90 cases, which is low for reliably mining process models —, the second phase of the correlation miner was able to find a reasonable number of relations in the event logs.

As we discussed in Sec. 4.3, the value for the recall of the real-world event log is generally lower due to the exceptional cases that have been recorded in the event logs. In order to minimize the effect of these exceptions, we shrank the original real-world event by removing the edges that have a low frequency (i.e. <50) and, subsequently, we generated four-event logs with randomly selected cases. These event logs resulted in an increase both reconstructed event logs' recall and precision, on average, to 71% and 74%, respectively. The detailed results of these experiments are illustrated in Table 11. Consequently, based on these results, we claim that the event log reconstruction phase of the correlation miner is also applicable for mining real-world event logs.

Table 10. Results of using the correlation miner for a real-world event log.

Run	Model recall	Log recall	Model precision	Log precision
1	0.62	0.74	0.72	0.78
2	0.57	0.66	0.60	0.69
3	0.59	0.63	0.56	0.70
4	0.55	0.59	0.63	0.65
Average	0.58	0.65	0.63	0.70

Table 11. Results of using the correlation miner for a real-world event log after removing less frequent edges.

Run	Model recall	Log recall	Model precision	Log precision
1	0.60	0.74	0.71	0.81
2	0.70	0.74	0.78	0.76
3	0.63	0.69	0.67	0.72
4	0.68	0.68	0.72	0.71
Average	0.65	0.71	0.72	0.74

7. Related Work

To the best of our knowledge, the closest work to the correlation miner has been published in Ref. 12 wherein the authors tackled the same problem, but based their analysis on event logs neither with case identifiers nor with timestamps. Since the correlation miner also uses timestamps, it is expected to produce more reliable results. Specifically, for the real-world event log (Sec. 4.3) the algorithm presented in Ref. 12 yields a model with precision 61% and recall 41%, which are 24% and 22% lower than the correlation miner's results respectively. However, Ref. 12 can process cyclic orchestration models.

Correlating events in an event log is a continuing challenge in the field of process mining, and more specifically, as suggested in Ref. 22, this challenge has a high importance in service mining. Therefore, we discuss related work in three categories: (i) process discovery, (ii) service mining, and (iii) event correlation.

A large and growing body of literature has been published in the field of process discovery. Van Dongen *et al.* have reviewed discovery algorithms that generate process models in the Petri Net notation.[2] Also, other surveys such as the one conducted by Tiwari *et al.*,[7] have shown that a significant number of studies employ different approaches to discover process models from event logs, including, but not limited to, data mining techniques,[23,24] genetic algorithms,[25,26] fuzzy algorithms,[3] markovian approaches[27] and cluster analysis.[28] A recent study by Verbeek and van der Aalst,[8] in which the authors have introduced a generic divide-and-conquer framework to enable process discovery and conformance checking for big event logs. Their approach is similar to our correlation miner in the sense that both have been implemented using Integer Linear Programming. However, the precise optimization problem that they solve is different.

The topic of service mining has been investigated in Ref. 6, where the authors have illustrated the potential of applying process mining in the context of web services by employing the ProM framework as a process mining tool and IBM's WebSphere as a reference system. Furthermore, Dustdar *et al.*[29,30] have introduced Web Services Interaction Mining, which concerns itself with performing process mining techniques in order to analyze service interactions. In Ref. 31 the authors have presented a web service mining framework aiming at the discovery of unexpected and interesting service compositions that automatically emerge in a bottom-up fashion. Moreover, Gaaloul, Baïna and Godart[32] has proposed a mining algorithm which discovers composite web service patterns from event logs by iteratively composing locally captured patterns. Finally, a study by van der Aalst[22] has reported challenges in service mining, which include the correlation challenge that we have tackled in this paper.

To the best of our knowledge, apart from Ref. 12, no process or service mining algorithms currently exist that can mine a log that contains no case identifiers and no additional data elements to do the correlation on.

Several techniques have been developed to facilitate event correlation in the context of service-oriented systems. Reference 33 identified a set of 18 correlation

patterns that have been grounded in a formal model. The concept of correlation set, a query to retrieve identifiers from messages that are unique for particular cases, has been included in many correlation techniques such as Ref. 34, in which an algorithm has been proposed to assigns a certain identifier to each case in a multi-party supply chains. De Pauw *et al.* carried out a study to discover conversations in web services by using semantic correlation analysis[35] in which different services pass dedicated identifiers inside their messages. Moreover, in Ref. 36, the authors developed an interactive semi-automated tool for event correlation from web service interaction logs.

In contrasts, the correlation miner enables fully automated event correlation and is only based on occurrences and timestamps of events rather than other data elements that are associated with events. In particular, our technique can be used to mine service collaboration scenarios, in which no main controller exists to manage interactions among the services, and therefore, no data element such as payload of services can be recorded (e.g. Ref. 37) whereas most of the related studies in which correlation will be performed on some data elements have been recorded by a main controller.

8. Conclusion

In this paper, we have presented a process discovery technique, the correlation miner, which enables mining of both business process models and relations from event logs that do not contain case identifiers. The correlation miner consists of two phases.

The core idea for the first phase of the correlation miner is to construct a business process model in such a way that the number of cases that flow into, and out of, an activity are equal to the number of events that happen for that activity. However, it is often possible to generate more than one process model that meets this rule. Therefore, we defined additional criteria that need to be fulfilled in order to select the best model. Since logs without case identifiers provide two elements of information: (i) how many times an event has occurred for a particular activity, and (ii) at what time an event has occurred, we designed our algorithm based on these two characteristics.

Accordingly, the process discovery part of the correlation miner has three steps. In the first two steps, it creates two matrices, the Precede/Succeed matrix and the Duration matrix. Given two activities, the corresponding element in the Precede/Succeed matrix indicates the fraction of events referring to the first activity that have occurred before the events referring to the second activity. If this value is high, it is more likely that there is an edge from the first to the second activity. Similarly, given two activities, the corresponding element in the Duration matrix indicates the average time difference between the events referring to the first activity and the events referring to the second activity. If this value is low, it is more likely that there is an edge from the first to the second activity. Finally, in the third

step, the correlation miner constructs all the possible business process models that meet the rule mentioned above and then it selects the best one based on the values from the Precede/Succeed matrix and the Duration matrix.

The second phase of the correlation miner aims at reconstructing cases from the event log. The algorithm proposed for this phase works by interpreting an event log as a set of binary relations between events, where a relation from one event to another means that this event is directly followed by another event in the same case. To determine these relations, the correlation miner tries to cast the problem into a Quadratic Programming (QP) problem, in which a set of constraints must hold for the binary relations and, subsequently, an optimization function selects the 'best' set of binary relations. This objective function is designed to comply with the assumption that the time differences between events that belong to the same case are more likely to follow a certain probability distribution, than the time differences between binary relations from different cases.

To evaluate the applicability of the correlation miner, we performed experiments with both synthetic event logs and a real-world event log. The results from the evaluation show that the correlation miner produces good results under most conditions. In particular, it produced a business process model with 85% precision and 63% recall for the real-world log with 11,647 cases. A set of truncated versions of this real-world event log was used to further evaluate the second phase of the correlation miner which resulted in a reconstructed event log with 74% precision and 71% recall when outliers were removed from the log. Furthermore, the evaluations with the synthetic logs show that results are better for structured models than for unstructured models. Also, results are better — especially for the first phase — when there are more cases in the log to mine from and when the inter-arrival time of the cases is higher than the duration of the activities.

The current correlation miner has been developed based on three assumptions. First of all, we assume that the underlying business process models that are mined by the correlation miner are acyclic. There is, therefore, a definite need for further research to enable mining of cyclic processes as well. Secondly, we assume that the source and the sink activities of the underlying business processes can be manually identified. In order to eliminate this assumption, future work is required to integrate one of existing techniques for determination of such activities in event logs with the correlation miner. Finally, for the synthesis event logs, we assume that all the executions of the tasks were recorded successfully; and there were no noisy, missing or incomplete events or cases in the logs. We have already mitigated this assumption by evaluating the correlation miner using a real-world event log. However, it is recommended that further experiments be undertaken using more noisy and incomplete event logs. Furthermore, additional work is needed to ensure that our algorithm can produce an accurate average time difference between activities for the duration matrix, given that we do not know which events belong to the same case and which should, therefore, be used to compute the time difference. As a closing note, there is room for further improvement in determining more efficient

mechanisms to explore the constraints in the second phase of the correlation miner which may improve the time performance of this algorithm when event logs with very large numbers of cases are mined.

Acknowledgment

The research leading to these results has received funding from the European Union's Seventh Framework Programme Under Grant Agreement 2012-318275 (GET Service).

References

1. W. van der Aalst, A. Adriansyah, A. K. A. de Medeiros, F. Arcieri, T. Baier, T. Blickle, J. C. Bose, P. van den Brand, R. Brandtjen, J. Buijs *et al.*, Process mining manifesto, in *Business Process Management Workshops* (Springer, 2011), pp. 169–194.
2. B. F. Van Dongen, A. K. de Medeiros and L. Wen, Process mining: Overview and outlook of petri net discovery algorithms, in *Transaction on Petri Nets and Other Models of Concurrency II* (Springer, 2009), pp. 225–242.
3. C. Günther and W. van der Aalst, Fuzzy mining–adaptive process simplification based on multi-perspective metrics, in *Business Process Management* (Springer, 2007), pp. 328–343.
4. A. Rozinat and W. van der Aalst, Conformance checking of processes based on monitoring real behavior, *Inf. Syst.* **33**(1) (2008) 64–95.
5. W. van der Aalst, *Process Mining*: *Data Science in Action* (Springer, 2016).
6. W. van der Aalst and E. Verbeek, Process mining in web services: The websphere case, *IEEE Data Eng. Bull.* **31**(3) (2008) 45–48.
7. A. Tiwari, C. J. Turner and B. Majeed, A review of business process mining: State-of-the-art and future trends, *Bus. Process Manag. J.* **14**(1) (2008) 5–22.
8. E. Verbeek and W. van der Aalst, Decomposed process mining: The ILP case, in *International Conference on Business Process Management* (Springer, 2014), pp. 264–276.
9. C. Günther and A. Rozinat, Disco: Discover your processes, in *Proc. Business Process Management 2012 Demo Track*, Vol. 940, *CEUR Workshop Proceedings* (2012), pp. 40–44.
10. W. van der Aalst, Service mining: Using process mining to discover, check, and improve service behavior, *IEEE Trans. Serv. Comput.* **6**(4) (2013) 525–535.
11. S. Pourmirza, R. Dijkman and P. Grefen, Correlation mining: Mining process orchestrations without case identifiers, in *Service-Oriented Computing* (Springer, 2015), pp. 237–252.
12. D. R. Ferreira and D. Gillblad, Discovering process models from unlabelled event logs, in *Business Process Management* (Springer, 2009), pp. 143–158.
13. A. A. de Medeiros, B. F. van Dongen, W. M. Van der Aalst and A. Weijters, Process mining: Extending the α-algorithm to mine short loops. Technical report, BETA working paper series, WP 113, Eindhoven University of Technology, Eindhoven (2004).
14. D. B. Johnson, Finding all the elementary circuits of a directed graph, *SIAM J. Comput.* **4**(1) (1975) 77–84.
15. Gurobi Inc. Gurobi optimizer reference manual. 2012. http://www.gurobi.com.
16. B. van Dongen, A. K. de Medeiros, E. Verbeek, T. Weijters and W. van der Aalst, The prom framework: A new era in process mining tool support, in *Applications and Theory of Petri Nets 2005* (Springer, 2005), pp. 444–454.

17. J. Mendling, H. A. Reijers and J. Cardoso, What makes process models understandable? in *Business Process Management* (Springer, 2007), pp. 48–63.
18. A. Schrijver, Combinatorial optimization: polyhedra and effciency, *Discrete Applied Mathematics* **146** (2005) 120–122.
19. B. van Dongen, Bpi challenge 2012. dataset (2012).
20. B. Efron, Nonparametric estimates of standard error: The jackknife, the bootstrap and other methods, *Biometrika* **68**(3) (1981) 589–599.
21. C. Bliek1ú, P. Bonami and A. Lodi, Solving mixed-integer quadratic programming problems with ibm-cplex: A progress report, in *Proceedings of the Twenty-Sixth RAMP Symposium* (2014), pp. 16–17.
22. W. van der Aalst, Challenges in service mining: Record, check, discover, in *Web Engineering* (Springer, 2013), pp. 1–4.
23. M. Hammori, J. Herbst and N. Kleiner, *Interactive Workflow Mining* (Springer, 2004).
24. S.-Y. Hwang and W.-S. Yang, On the discovery of process models from their instances, *Decis. Support Syst.* **34**(1) (2002) 41–57.
25. W. van der Aalst, A. K. de Medeiros and T. Weijters, Genetic process mining, in *Applications and Theory of Petri Nets* (Springer, 2005), pp. 48–69.
26. A. K. de Medeiros, T. Weijters and W. van der Aalst, Genetic process mining: An experimental evaluation, *Data Min. Knowl. Discov.* **14**(2) (2007) 245–304.
27. H. Mannila and D. Rusakov, Decomposition of event sequences into independent components, in *Proceedings of the 2001 SIAM International Conference on Data Mining* (SIAM, 2001), pp. 1–17.
28. G. Greco, A. Guzzo, L. Pontieri and D. Sacca, Mining expressive process models by clustering workflow traces, in *Advances in Knowledge Discovery and Data Mining* (Springer, 2004), pp. 52–62.
29. S. Dustdar and R. Gombotz, Discovering web service workflows using web services interaction mining, *Int. J. Bus. Process. Inter. Manage.* **1**(4) (2006) 256–266.
30. S. Dustdar, R. Gombotz and K. Baïna, Web services interaction mining. Technical report, Technical Report TUV-1841-2004-16, Information Systems Institute, Vienna University of Technology, Wien, Austria, 2004.
31. G. Zheng and A. Bouguettaya, Service mining on the web, *IEEE Trans. Serv. Comput.* **2**(1) (2009) 65–78.
32. W. Gaaloul, K. Baïna and C. Godart, Log-based mining techniques applied to web service composition reengineering, *Serv. Oriented Comput. Appl.* **2**(2–3) (2008) 93–110.
33. A. Barros, G. Decker, M. Dumas and F. Weber, Correlation patterns in SoA, in *Fundamental Approaches to Software Engineering* (Springer, 2007), pp. 245–259.
34. K. Gerke, J. Mendling and K. Tarmyshov, Case construction for mining supply chain processes, in *Business Information Systems* (Springer, 2009), pp. 181–192.
35. W. De Pauw, R. Hoch and Y. Huang, Discovering conversations in web services using semantic correlation analysis, in *ICWS* (IEEE, 2007), pp. 639–646.
36. H. R. Motahari-Nezhad, R. Saint-Paul, F. Casati and B. Benatallah, Event correlation for process discovery from web service interaction logs, *VLDB J.* **20**(3) (2011) 417–444.
37. S. Pourmirza, R. Dijkman and P. Grefen, Switching parties in a collaboration at runtime, in *IEEE 18th Int. Enterprise Distributed Object Comput. Conf.* (*EDOC*) (2014), pp. 136–141.