A Probabilistic Approach to Event-Case Correlation for Process Mining

Dina Bayomie[1 [0000−0002−2549−6407]], Claudio Di Ciccio[1 [0000−0001−5570−0475]], Marcello La Rosa[2 [0000−0001−9568−4035]], and Jan Mendling[1 [0000−0002−7260−524X]]

[1] Vienna University of Economics and Business, Austria
{dina.sayed.bayomie.sobh,claudio.di.ciccio,jan.mendling}@wu.ac.at
[2] University of Melbourne, Australia
marcello.larosa@unimelb.edu.au

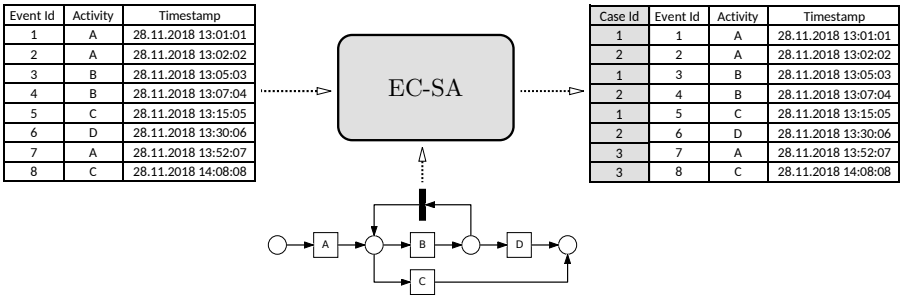
Abstract. Process mining aims to understand the actual behavior and performance of business processes from event logs recorded by IT systems. A key requirement is that every event in the log must be associated with a unique case identifier (e.g., the order ID in an order-to-cash process). In reality, however, this case ID may not always be present, especially when logs are acquired from different systems or when such systems have not been explicitly designed to offer process-tracking capabilities. Existing techniques for correlating events have worked with assumptions to make the problem tractable: some assume the generative processes to be acyclic while others require heuristic information or user input. In this paper, we lift these assumptions by presenting a novel technique called *EC-SA* based on probabilistic optimization. Given as input a sequence of timestamped events (the log without case IDs) and a process model describing the underlying business process, our approach returns an event log in which every event is mapped to a case identifier. The approach minimises the misalignment between the generated log and the input process model, and the variance between activity durations across cases. The experiments conducted on a variety of real-life datasets show the advantages of our approach over the state of the art.

Keywords: Event correlation · Simulated annealing · Process mining

1 Introduction

Recent years have seen a drastically increasing availability of process execution data from various data sources [16]. Process mining offers different analysis techniques that can help to extract business insights from these data, known as event logs. To this end, each event in an log must have at least three attributes [17]: *(i)* the *event class* referring to a specific activity in the process (e.g., “Order checked” or ”Claim assessed”), *(ii)* the *end timestamp* capturing the completion time for that activity, and *(iii)* the *case identifier* (e.g., the order number in an order-to-cash process, or the claim ID in a claims handling process). Data lake infrastructures, though, often put more emphasis on storing and synchronising data than on structuring them in a way that process mining can be readily applied [6,11].

Prior research has described the problem of missing case identifiers as a correlation problem, because the connections between different events has to be reestablished based on heuristics, domain knowledge or payload data. In essence, the correlation problem is concerned with identifying which events belong together to the same case when a unique



(a) Uncorrelated log (b) Process model (c) Event log

Fig. 1: Overview of the EC-SA approach.

case identifier is missing. Existing correlation techniques face the challenge of operating in a large search space. For this reason, previous proposals have introduced assumptions to make the problem tractable. Some techniques assume the generative processes to be acyclic [14,9] while others require heuristic information about the execution behavior of activities in addition to the process model [7]. Beyond that, performance has been an issue.

In this paper, we address the correlation problem as a multi-level optimization problem. We propose a novel technique called *EC-SA* (Events Correlation by Simulated Annealing), which is based on simulated annealing. As illustrated in Fig. 1, the technique takes as input a set of uncorrelated events and a (normative or descriptive) process model that captures knowledge of the underlying business process, and produces an event log as output. The technique revolves around two nested objectives. First, it seeks to minimize the misalignment between the generated log and an input process model; second, it seeks to minimize the activity execution time variance across cases. This latter objective builds on the assumption that same activities tend to have similar duration across cases.

The remainder of this paper is organized as follows. Section 2 discusses the related work. Section 3 presents the different phases of our novel EC-SA event correlation technique. Section 4 then discusses the experimental evaluation on real-life logs before Section 5 concludes the paper.

2 Related Work

Several techniques have been defined to address the event correlation problem. The following ones correlate the events from the control flow perspective. The greedy approach in [9] estimates a Markov model for an uncorrelated event log. It does not support cyclic behavior. It is sensitive to concurrency and the number of overlapping cases at a given point in time. In [18], the authors provide an approach that uses sequence partition to search the solution space for the minimal set of patterns that can represent the uncorrelated event log. The approach does not support cyclic. As an output, it produces the behavioral patterns of the log. The Correlation Miner (CMiner) approach [14] works in two phases. The first phase is discovering an acyclic process model from the uncorrelated log using linear programming. In the second phase, the discovered model is used to correlate the events by solving quadratic programming constraints. It does not support cyclic models.

The performance of the approach is highly sensitive to the amount of uncorrelated events mainly because of the quadratic-constraints based phase. The Deducing Case Ids (DCI) approach in [7] searches the solution space for the possible correlation between the events, and prunes the search space based on the given input in terms of the process model and heuristic data on activity execution behavior. DCI supports the cyclic processes.It is sensitive to the quality of the input data and not computationally efficient.

Our approach has common factors with CMiner and DCI. The three techniques consider the activity duration to find a solution, and they use the control flow to identify the correlation between the events.

Two techniques tackle the correlation problem by considering the data perspective have been devised to date. In [13] the authors address the correlation problem in the web service environment. Their semi-automated approach correlates events of the service logs as process views based on a correlation condition using the event data from different data layers. In [15], the authors address the problem of having the event data stored in databases. They mine the association and correlation rules over the different attributes in the database. Then, they measure the support of these rules over the data and use the most supported rules for correlating the events. They use the MapReduce technique to improve the performance of applying the correlation constraints.

In summary, these recent techniques make assumptions about process behavior, available information and size of search space. Our *EC-SA* technique lifts those assumptions.

3 Event Correlation Technique

We treat the problem of automatically correlating events as a multi-level optimization problem. Specifically, our technique, EC-SA, is based on population-based simulated annealing [2]. The technique revolves around two nested objectives: *(i)* minimizing the misalignment between the generated event log and the input process model, and *(ii)* minimizing the activity execution time variance across cases. In order to describe our technique precisely, we first introduce a number of preliminary concepts.

3.1 Preliminaries

We introduce the basic notions of event, uncorrelated event log, case and projection of a case over an event attribute. Thereupon, we present the definitions of event log and trace.

An *event* e is an atomic unit of execution. Events bear *attributes*. In particular, we assume the following attributes to be mandatory: activity name Act (string) and timestamp Ts (date-time). The value of attribute X on event e shall be denoted as $e.X$, e.g., $e.\mathrm{Ts}$ refers to the timestamp of e. We assume a total order $\preccurlyeq$ to be defined over the universe of events. Therefore, we assign to every event a unique integer index (or *event id* for short), induced by $\preccurlyeq$ on the events. We shall denote the index i of an event e as a subscript, e_i. We assume the assignment of Ts to be coherent with $\preccurlyeq$, i.e., if $e \preccurlyeq e'$ then $e.\mathrm{Ts} \leqslant e'.\mathrm{Ts}$. In Fig. 1(a), e.g., event e_3 is such that $e_3.\mathrm{Act} = \mathsf{B}$ and $e_3.\mathrm{Ts} = 28.11.2018\ 13{:}05{:}03$.

An *uncorrelated event log* (or *uncorrelated log* for short) UL is a finite set of events with total order $\preccurlyeq$. Figure 1(a) depicts an uncorrelated event log UL.

A *case* $\sigma = \langle e_{\sigma_1},\ldots,e_{\sigma_n}\rangle$ is a finite sequence of length n of events e_{σ_i} with $1 \leqslant i \leqslant n$ induced by $\preccurlyeq$, i.e., such that $e_{\sigma_i} \preccurlyeq e_{\sigma_k}$ for every $i \leqslant k \leqslant n$. We assume every case to be

assigned a unique case identifier (case id for short), namely an integer in a convex subset. The value of attribute X over case σ shall be denoted as $\sigma.X$. In Fig. 1(c), $\sigma_2 = \langle e_2, e_4, e_6 \rangle$. We write $L(e) = \sigma$ to indicate that the case of event e in log L is σ. In the example of Fig. 1(c), $L(e_2) = L(e_4) = L(e_6) = \sigma_2$.

An *event log* $L = \{\sigma_1, \ldots, \sigma_n\}$ is a finite non-empty set of non-overlapping cases, i.e., if $e \in \sigma_i$, then $e \notin \sigma_j$ for all $i, j \in [1 \ldots n]$, $i \neq j$. We denote its cardinality as $|L| = N$ with $N \geqslant 1$. Figure 1(c) shows an event log consisting of cases σ_1, σ_2, and σ_3. Notice that the union of all events of every case of a log L, together with the total order defined on them, is an uncorrelated log. For the sake of conciseness, we denote this totally ordered events set as $UL(L)$. A *trace* t is a projection of a case σ over the activity names: $t = \mathrm{Act}(\sigma)$. In Fig. 1(c), $\mathrm{Act}(\sigma_2) = \langle \mathsf{A}, \mathsf{B}, \mathsf{D} \rangle$.

3.2 The problem

Given an uncorrelated log UL as input (like the one in Fig. 1(a)), the output of EC-SA is an event log L that partitions UL into a set of cases, i.e., such that for every event $e \in UL$ there exists one and only one case $\sigma \in L$ s.t. $e \in \sigma$. Figure 1(c) shows such a log. To derive L from UL, EC-SA considers as an additional input a process model (e.g., the Workflow net depicted in Fig. 1(b)), which drives the mapping of events to cases. We assume the process model to have exactly one start activity (initially, only one activity is enabled) and to expose terminating conditions (at some stage of the run, no activity is enabled any longer).

3.3 Multi-level optimization

The event correlation problem can be solved by optimization metaheuristic techniques. EC-SA uses a multi-level simulated annealing as optimization technique. *Simulated Annealing* (SA) is a metaheuristic that searches for the nearest approximate global solution in the search space of the optimization problem, by simulating the cooling process of metals through the annealing process. Using SA to solve the event correlation problem helps in finding a global optimal correlated log in reasonable time.

SA explores the search space through the following steps. It starts by creating the initial *population*, as we are using the population-based SA [2]. A population (pop) is a non-empty set of individuals ($|\mathrm{pop}| \geqslant 1$). The population is formed by generating random *individual*s. Then SA initializes the current step $S_{\mathrm{curr}} = 1$ and the current temperature with an initial temperature, $\tau_{\mathrm{curr}} = \tau_{\mathrm{init}}$. The annealing process begins with the generation of a neighbor solution x' for the current individual x. Next, SA computes the *energy cost function* between x and x', namely $\delta \mathrm{f_c}(x, x')$. Both $\delta \mathrm{f_c}(x, x')$ and τ_{curr} are used as input to compute the *acceptance probability* of the new neighbor solution, which we denote as $\mathrm{prob}(x')$. $\mathrm{prob}(x')$ determines if the new neighbor, x', can be used as the next individual. Notice that $\mathrm{prob}(x')$ may select x' even though it performs worse than x in order to increase the chances to skip the local optimum and let the algorithm further explore the search space. At each iteration, SA compares the global optimal solution x_{G} at step $S_{\mathrm{curr}} - 1$, i.e., the best solution over the iterations $[0, S_{\mathrm{curr}}[$, with the local optimal solution x_{L} in pop at S_{curr} based on $\delta \mathrm{f_c}(x_{\mathrm{G}}, x_{\mathrm{G}})$. Thus, SA can return the best solution over all the iterations. Finally, SA uses a cooling schedule that defines the rate at which the temperature (τ_{curr}) cools down, and increments S_{curr} by 1. SA repeats the annealing and cooling process till S_{curr} reaches the maximum number of iterations (S_{max}).

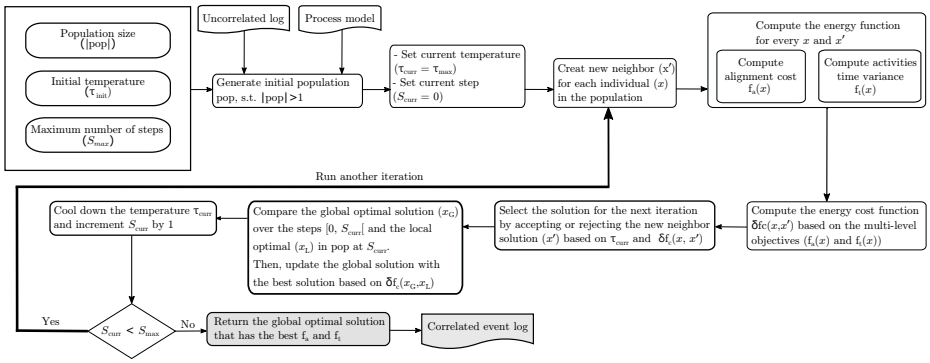


Fig. 2: The EC-SA technique

SA has a set of parameters that influence the annealing process: *(i)* the initial temperature (τ_{init}), *(ii)* the maximum number of steps ($S_{\max}$), and *(iii)* the population size ($|\text{pop}|$). In addition to these parameters, SA requires the following main functions to be defined: *(i)* the creation of a new neighbor (x'), *(ii)* the energy cost function ($\delta \text{f}_\text{c}(x,x')$), *(iii)* the acceptance probability ($\text{prob}(x')$), and *(iv)* the cooling schedule. We implement those functions to resolve the event correlation problem. In the rest of this section, we discuss the SA steps using the defined functions.

The cooling schedule simulates the cooling-down technique of the annealing process by controlling the computation of the current temperature τ_{curr}. We use the logarithmic function schedule, as follows: $\tau_{\text{curr}} = \frac{\tau_{\text{init}}}{\ln(1+S_{\text{curr}})}$.

Figure 2 shows the steps of EC-SA. Aside of the aforementioned SA-specific parameters, it requires as input *(i)* an uncorrelated event log (e.g., the one depicted in Fig. 1(a)) and *(ii)* a process model (e.g., the one of Fig. 1(b)). The approach generates a (correlated) event log as its output (depicted, e.g., in Fig. 1(c)). We discuss the steps in details through the following subsections.

Creating the initial population. As shown in Fig. 2, the first step is the generation of the initial population, pop, of size $|\text{pop}| \geqslant 1$. An individual represents a possible event log, such as the one depicted in Fig. 3(a). We use a dedicated data structure for such individuals that we name log array.

A *log array* LA is an associative array mapping every event to a case. The size of the log array is $|LA| = |UL|$. We write $LA(e) = \sigma$ to indicate that event e is mapped to case σ by log array LA. In Figs. 3(a) and 4, the log array elements are labeled with the corresponding event (e.g., B_4 is event e_4, where $e_4.\text{Act} = \text{B}$). The content of each cell is the assigned case id (e.g., B_4 is assigned with case id 2, i.e., $LA(e_4) = \sigma_2$). LA is created by replaying the uncorrelated events on the process model. The replaying step is repeated based on the population size. In our example, we assume $|\text{pop}| = 1$ for readability purposes.

To generate the log array, we replay all events on the process model. Every run from the initial activity to the termination conditions will correspond to a case. We name the cases corresponding to non-terminated runs as *open cases*. We figure three scenarios when replaying an event e over the input process model:

1. e corresponds to the execution of the start activity of the process model (we name it *start event*); then, a new run starts and a new case is open, accordingly;
2. e corresponds to the execution of an enabled (non-starting) activity on one (or more) runs (*enabled event*); then, e is assigned to the case of the run that enables its activity, or, if more runs enable it, it is assigned randomly to one of those.
3. e does not correspond to any enabled activity (*non-enabled event*); then, e is assigned randomly to one of the open cases, although its activity was not enabled. This way, we guarantee to correlate all the events even when the log has deviated from the model.

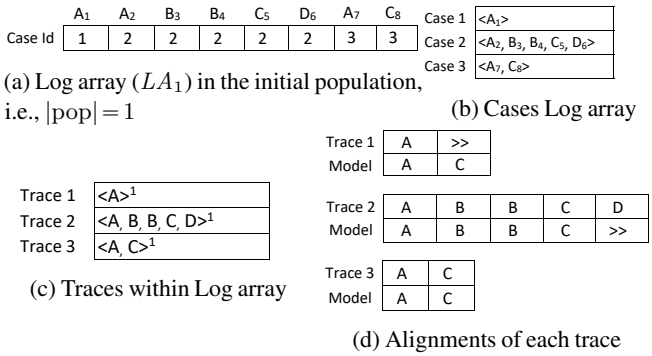
The replaying of the uncorrelated events in Fig. 1(a) on the process model in Fig. 1(b) generates the LA_1 individual in Fig. 3(a). In the example, upon e_1 we generate σ_1, and upon e_2, σ_2 starts. Afterwards, there are two open cases within the log before e_3: both σ_1 and σ_2 expect the execution of activity B or activity C. Therefore, e_3 can belong to each of those cases. In Fig. 3(a), e_3 is assigned to σ_2, as well as e_4 and e_5. To guarantee the assignment of all uncorrelated events to some case, we randomly assign the non-enabled events to one of the open cases. For instance, e_6 is a non-enabled event, as both σ_1 and σ_2 are not expecting activity D. Thus, e_6 is randomly assigned to one of the open cases (σ_2, in the example).

Creating a neighbor solution. The fundamental step of exploring the search space in the simulated annealing technique is creating a new neighbor based on the current solution. We explore the search space by selecting a changing point in the current individual (x) and replay the events on the model from that point on in order to find a different solution (x'). The selection of the changing point is based on the current step (S_{curr}) to determine from which part of the LA the change may occur. For instance, when $S_{\text{curr}} = 1$ the changing point is randomly selected from the beginning of the log, i.e., the first few events. Instead, when $S_{\text{curr}} = S_{\max} - 1$ the changing point is randomly selected from the end of the log, i.e., the last few events. The continuous increment of S_{curr} leads to reducing the number of events to be replayed at each iteration; this is in line with the cooling down mechanism of the simulated annealing approach.

Energy cost function. The *energy function* is a fundamental part of simulated annealing. As shown in Fig. 2, this function is divided into two components to support a bilevel objective and solve the correlation problem. The first-level objective ($\mathrm{f_a}(x)$) aims at minimizing the misalignment between the output log and the given process model. The second-level objective ($\mathrm{f_t}(x)$) aims at minimizing the activity execution time variance within the log.

To measure the model-log misalignment we use the well-established *alignment cost* function proposed in [1]. Figure 3 shows an example of the alignment computation. The first step is to extract the cases from the log array as shown in Fig. 3(b). Then, we project the traces from the cases as shown in Fig. 3(c). For each trace within the log, we compute the raw alignment cost ($\delta_A(t_i)$) of the trace w.r.t. the process model. For example, Fig. 3(d) shows that the model cannot execute activity D in t_2, so it is considered as a move in the log. On the other hand, activity C in t_1 is considered as a move in the model as it does not occur in the trace although the model would require it. The third trace has no deviations. The raw cost of the log is the summation of the traces' alignment cost. For instance, the raw cost of the log array in Fig. 3(a) is $\mathrm{f_a}(LA_1) = \delta_A(t_1) + \delta_A(t_1) + \delta_A(t_3) = 1 + 1 + 0 = 2$.

The second objective of EC-SA is to minimize the activity execution time variance. We assume that the same activities tend to have similar execution duration across cases. We thus calculate the time variance using the Mean Square Error (MSE) as in Eq. (2). MSE measures the deviation between the expected activities durations and the correlated



(a) Log array (LA_1) in the initial population, i.e., $|\text{pop}| = 1$

(b) Cases Log array

(c) Traces within Log array

(d) Alignments of each trace

Fig. 3: Alignment computation of LA_1

events durations. Given an activity a, we compute the average of the activity durations as the expected value $\text{Time}_{\text{avg}}(a)$. Given a case $\sigma = \langle e_{\sigma_1},...,e_{\sigma_n}\rangle$, we compute the Elapsed Time (ET), i.e., the event duration, of an event $e_{\sigma_i} \in \sigma$ (with $1 \leqslant i \leqslant n$) as follows:

$$\text{ET}(\sigma, e_{\sigma_i}) = \begin{cases} e_{\sigma_i}.\text{Ts} - e_{\sigma_{i-1}}.\text{Ts} & \text{if } 1 < i \leqslant n \\ 0 & \text{otherwise} \end{cases} \tag{1}$$

Recalling that with $LA(e)$ we indicate the case to which e is assigned by log array LA, the (time-)MSE is computed as follows.

$$\text{MSE}(LA) = \frac{1}{|LA|} \sum_{i=0}^{|LA|} (\text{Time}_{\text{avg}}(e_i.\text{Act}) - \text{ET}(LA(e_i), e_i))^2 \tag{2}$$

The MSE is used as the second-level objective function, f_t. For example, the average execution times (in minutes) of each activity in the log array of Fig. 3(a) are $\text{Time}_{\text{avg}}(\text{B}) = 2.52$, $\text{Time}_{\text{avg}}(\text{C}) = 12.02$, and $\text{Time}_{\text{avg}}(\text{D}) = 15.02$. Considering the expected time values for each of the events, we have that $\text{f}_\text{t}(LA_1) = \text{MSE}(LA_1) = 4.1$ mins.

Based on the energy function, i.e., on objective functions f_a (alignment cost) and f_t (time-MSE), the *energy cost function*, $\delta\text{f}_\text{c}(x, x')$, is computed as follows.

$$\delta\text{f}_\text{c}(x, x') = \begin{cases} \text{f}_\text{a}(x') - \text{f}_\text{a}(x) & \text{if } \text{f}_\text{a}(x') > \text{f}_\text{a}(x) \\ \text{f}_\text{t}(x') - \text{f}_\text{t}(x) & \text{otherwise.} \end{cases} \tag{3}$$

$$\text{prob}(x') = \exp^{\frac{-\delta\text{f}_\text{c}(x,x')}{\tau_\text{curr}}} \tag{4}$$

The acceptance probability, $\text{prob}(x')$, is computed using $\delta\text{f}_\text{c}(x, x')$ and the current temperature (τ_curr) as shown in Eq. (4). EC-SA compares the value of $\text{prob}(x')$ with a random value in the interval $[0,1]$ to accept (if higher) or reject (if lower) the new neighbor. In this way, we simulate the annealing process, enforced by the fact that the decrease of the τ_curr temperature also diminishes the randomness of the choice. Furthermore, notice that the memory-less stochastic perturbation makes it possible to skip the local optimal.

Algorithm 1: Selection of the solution for the next iteration

input : Current LogArray x; new neighbor LogArray x'
output : Selected LogArray

```
if f_a(x') ⩽ f_a(x) then return x' ;
else if f_a(x') = f_a(x) then
    if f_t(x') ⩽ f_t(x) or prob(x') ⩾ random(0,1) then return x' ;
else if f_a(x') > f_a(x) and prob(x') ⩾ random(0,1) then return x' ;
return x
```

Selection of the next individual. Algorithm 1 shows the full selection procedure of the individual for the next iteration. Its decision between x and x' is based on the objective functions, $\mathrm{f_a}$ (first-level) and $\mathrm{f_t}$ (second-level), and $\mathrm{prob}(x')$ (perturbation). If the new neighbor (x') has a lower alignment cost, then it is selected. If the new neighbor (x') and current individual (x) have the same alignment cost, then we check the activity time variance. We alter the final decision with a random selection weighed by $\mathrm{prob}(x')$ which, in turn, is calculated on the basis of the current temperature, τ_{curr}, and $\delta \mathrm{f_c}(x,x')$. As Eq. (3) shows, also $\delta \mathrm{f_c}(x,x')$ considers $\mathrm{f_t}(x)$ and $\mathrm{f_t}(x')$ in this case. On the contrary, if the new neighbor has a higher alignment cost than the current individual, we calculate $\delta \mathrm{f_c}(x,x')$ based on $\mathrm{f_a}(x)$ and $\mathrm{f_a}(x')$. The acceptance probability, again, may alter the decision. This process is repeated for each individual within the population.

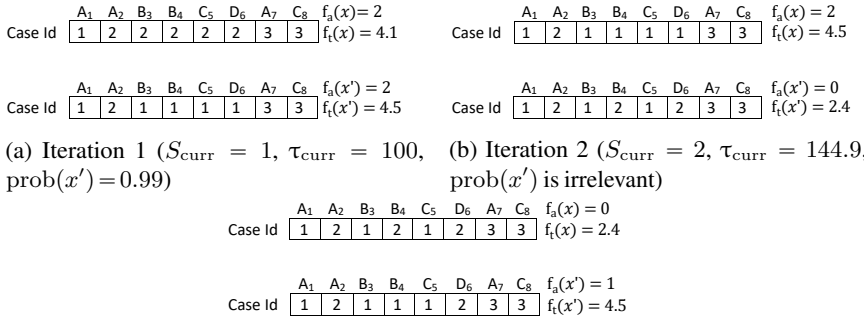
Running Example. Figure 4 shows the intermediate results within the EC-SA iterations. We assume that $S_{\max} = 4$ and $\tau_{\mathrm{init}} = 100$. Figure 4(a) shows the new individual created from the initial individual $x = LA_1$. Since $\mathrm{f_a}(x) = \mathrm{f_a}(x')$ and $\mathrm{f_t}(x) < \mathrm{f_t}(x')$, then $\delta \mathrm{f_c}(x,x')$ is computed considering $\mathrm{f_t}$. $\delta \mathrm{f_c}(x,x')$ is equal to 0.4. Thus, $\mathrm{prob}(x')$ is calculated on the basis of $\tau_{\mathrm{curr}} = 100$ and $\delta \mathrm{f_c}(x,x')$. Upon the comparison of $\mathrm{prob}(x')$ against a random value in $[0,1]$, we assume that x' is selected and replaces x in the population. Figure 4(b) shows the second iteration, where $\mathrm{f_a}(x') = 0 \leqslant \mathrm{f_a}(x) = 2$. Therefore, x' is directly selected without computing the acceptance probability as it performs better than the current individual (x). Figure 4(c) shows the last iteration. The new neighbor achieves a higher $\mathrm{f_a}(x') = 1$ than the current individual ($\mathrm{f_a}(x) = 0$). Thus, $\delta \mathrm{f_c}(x,x')$is computed on the basis of $\mathrm{f_a}$. Based on $\mathrm{prob}(x')$ at $\tau_{\mathrm{curr}} = 91$ and a selection against the random value, we assume that x' is rejected and x is kept in the population.

Finally, EC-SA returns the solution that has the best $\mathrm{f_a}$ and $\mathrm{f_t}$ over all the iterations, i.e., the global optimal solution x_{G} till $S_{\max}$, as shown in Fig. 1(c). Following the EC-SA steps in Fig. 2, the algorithm proceeds until $S_{\mathrm{curr}} = S_{\max}$. The replaying of the events from different changing points in the log over the iterations grows the search space to explore. Accepting a worse solution than the current solution in some iterations helps to skip the optimal local solution and reach the optimal global solution.

4 Evaluation

We implemented EC-SA in a freely available prototype tool.[3] Using this tool we conducted two experiments to evaluate the accuracy and time performance of our approach, and compared the results with the state-of-the-art approach DCI [7].

[3] Available at https://github.com/DinaBayomie/EC-SA/releases/tag/v1.0



(a) Iteration 1 ($S_{\text{curr}} = 1$, $\tau_{\text{curr}} = 100$, $\text{prob}(x') = 0.99$)

(b) Iteration 2 ($S_{\text{curr}} = 2$, $\tau_{\text{curr}} = 144.9$, $\text{prob}(x')$ is irrelevant)

(c) Iteration 3 ($S_{\text{curr}} = 3$, $\tau_{\text{curr}} = 91$, $\text{prob}(x') = 0.98$)

Fig. 4: EC-SA iterations, with $S_{\max} = 3$ and $\tau_{\text{init}} = 100$

4.1 Design

Given an event log with correlated events (the "original log"), we removed the case identifiers and created an uncorrelated set of events. Using the latter log as input, we conducted two experiments (see Fig. 5). First, we measured the accuracy of the log generated by our approach against the original log, by taking as input process knowledge the set of distinct traces extracted from the original log itself. The purpose of this first experiment was to measure the loss of accuracy in the log produced by EC-SA, when using as input the equivalent of a perfectly fitting and precise process model (as represented by the set of traces of the original log). In the second experiment, instead of the distinct traces of the original log, we used as process knowledge the process model mined from the original log using two state-of-the-art automated discovery methods: Split Miner [4] and Inductive Miner [10]. These two methods strike different tradeoffs in terms of fitness, which captures the degree to which the discovered process model is able to recognize the traces in the event log, and precision, which captures the extent to which the behavior allowed by the process model is observed in the event log. The purpose of this second experiment was to measure how well our approach is able to correlate events, in spite of an input model that is not perfectly fitting nor precise. This second scenario is closer to reality, where a process model may be available within the organisation, though this model is not a faithful representation of the behavior captured by the set of uncorrelated events we want to correlate. Finally, we compared the results of the second experiment with the DCI approach as a baseline.

To measure the accuracy of the event log generated by EC-SA w.r.t. the original log, we used two measures: L2L_{sim} and $\text{L2L}_{\text{SMAPE}}$. L2L_{sim} is a log-to-log similarity measure, defined as the average string-edit distance between each trace of the generated log and its closest trace in the original log, weighted by the relative frequency of each trace in the two logs (cf. Def. 1). In essence, this measure is the transposition of the alignment-based fitness measure between a model and a log [1] to the case of two logs. $\text{L2L}_{\text{SMAPE}}$ is the symmetric mean absolute error of the events elapsed time between the two logs (cf. Def. 2). We used this measure to assess the time deviation between the events in the generated log and those in the original log. Finally, we measured the time taken by our approach and by DCI to complete the correlation task, using a timeout of 24 hours.

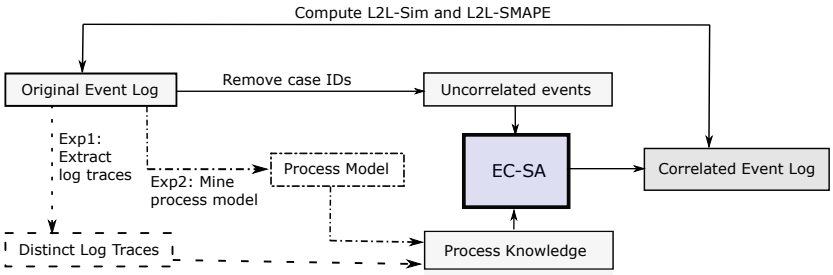


Fig. 5: Evaluation steps

Table 1: Example illustrating $\mathrm{L2L_{sim}}$ computation steps

(a) L_1 traces

$t_{1,1}=\mathrm{Act}(\sigma_{1,1})=\langle \mathrm{A,B,C}\rangle$
$t_{1,2}=\mathrm{Act}(\sigma_{1,2})=\langle \mathrm{A,B,D}\rangle$
$t_{1,3}=\mathrm{Act}(\sigma_{1,3})=\langle \mathrm{A,C}\rangle$

(b) L_2 traces

$t_{2,1}=\mathrm{Act}(\sigma_{2,1})=\langle \mathrm{A,B,B,D}\rangle$
$t_{2,2}=\mathrm{Act}(\sigma_{2,2})=\langle \mathrm{A,C}\rangle$
$t_{2,3}=\mathrm{Act}(\sigma_{2,3})=\langle \mathrm{A,C}\rangle$

(c) Iteration 1

$\Delta^{\mathrm{ins}}_{\mathrm{del}}(t_{1,3},t_{2,2})=0$	$\Delta^{\mathrm{ins}}_{\mathrm{del}}(t_{1,3},t_{2,3})=0$	$\Delta^{\mathrm{ins}}_{\mathrm{del}}(t_{1,1},t_{2,3})=1$
$\Delta^{\mathrm{ins}}_{\mathrm{del}}(t_{1,2},t_{2,1})=1$	$\Delta^{\mathrm{ins}}_{\mathrm{del}}(t_{1,1},t_{2,2})=1$	$\Delta^{\mathrm{ins}}_{\mathrm{del}}(t_{1,2},t_{2,2})=3$
$\Delta^{\mathrm{ins}}_{\mathrm{del}}(t_{1,2},t_{2,3})=3$	$\Delta^{\mathrm{ins}}_{\mathrm{del}}(t_{1,1},t_{2,1})=3$	$\Delta^{\mathrm{ins}}_{\mathrm{del}}(t_{1,3},t_{2,1})=4$

Definition 1 ($\mathrm{L2L_{sim}}$). *Let $\Delta^{\mathrm{ins}}_{\mathrm{del}}$ be the string-edit distance allowing only for insertions and deletions [12]. Let L_1 and L_2 be two event logs of same cardinality N, $L_1=\{\sigma_{1,1},\sigma_{1,2},\ldots,\sigma_{1,N}\}$ and $L_2=\{\sigma_{2,1},\sigma_{2,2},\ldots,\sigma_{2,N}\}$. We define the pair of trace-closest cases of L_1 and L_2, $(\sigma_{1,\star},\sigma_{2,\star})$, as follows:*

$$(\sigma_{1,\star},\sigma_{2,\star})=\underset{\substack{\sigma_{1,i}\in L_1\\ \sigma_{2,j}\in L_2}}{\mathrm{argmin}}\left\{\Delta^{\mathrm{ins}}_{\mathrm{del}}(\mathrm{Act}(\sigma_{1,i}),\mathrm{Act}(\sigma_{2,j}))\right\} \textit{ with } 1\leqslant i,j\leqslant N \tag{5}$$

The log-to-log similarity distance $\mathrm{L2L_{sim}}$ is thus inductively defined as follows:

$$N\times\mathrm{L2L_{sim}}(L_1,L_2)=\begin{cases}\Delta^{\mathrm{ins}}_{\mathrm{del}}(\mathrm{Act}(\sigma_{1,1}),\mathrm{Act}(\sigma_{2,1})) & \textit{if } |L_1|=|L_2|=1\\ \mathrm{L2L_{sim}}(L_1\backslash\{\sigma_{1,\star}\},L_2\backslash\{\sigma_{2,\star}\}) & \textit{otherwise}\end{cases} \tag{6}$$

Operationally, $\mathrm{L2L_{sim}}$ is computed as follows: we first sort L_1 and L_2 by their trace-closest pairs of cases, then we sum up the respective $\Delta^{\mathrm{ins}}_{\mathrm{del}}$ distances till saturation, and finally we derive $\mathrm{L2L_{sim}}$ by averaging the sum over the number of cases in the logs. Table 1 shows two example logs, L_1 in Table 1(a) and L_2 in Table 1(b), and the computation of $\mathrm{L2L_{sim}}(L_1,L_2)$. We compute $\Delta^{\mathrm{ins}}_{\mathrm{del}}$ for each pair of traces, and sort them as in Table 1(c). For instance, $\Delta^{\mathrm{ins}}_{\mathrm{del}}(t_{1,3},t_{2,2})$ is 0 as there is no deviation between the two traces. By selecting this pair, $\Delta^{\mathrm{ins}}_{\mathrm{del}}(t_{1,3},\sigma_{2,\star})$) and $\Delta^{\mathrm{ins}}_{\mathrm{del}}(t_{1,\star},\sigma_{2,2})$ are removed (the marked cells in Table 1(c)). Thus, $\mathrm{L2L_{sim}}(L_1,L_2)=\frac{2}{3}\approx 0.67$.

Definition 2 ($\mathrm{L2L_{SMAPE}}$). $\mathrm{L2L_{SMAPE}}$ *is the Symmetric Mean Absolute Error of the events Elapsed Time (ET) between two event logs, L_1 and L_2, such that their events are equivalent, i.e., $UL(L_1)=UL(L_2)$. Let $\mathrm{ET}(L(e),e)$ be the Elapsed Time of event e in log L as per Eq. (1). We define $\mathrm{L2L_{SMAPE}}$ as follows:*

$$\mathrm{L2L_{SMAPE}}(L_1,L_2)=\frac{1}{\sum_{i=0}^{|L_1|}|\sigma_i|}\times\sum_{e\in UL(L_1)}\frac{|\mathrm{ET}(L_1(e),e)-\mathrm{ET}(L_2(e),e)|}{|\mathrm{ET}(L_1(e),e)|+|\mathrm{ET}(L_2(e),e)|} \tag{7}$$

In the following, we will use the original log as L_1 and the generated log as L_2.

Table 2: Descriptive statistics of public logs

Evt. log	Traces		Events		Tr. length		
	Total	Dst. %	Total	Dst. %	m.	avg	M.
BPIC12	13087	33.4	262200	36	3	20	175
$BPIC13_{cp}$	1487	12.3	6660	7	1	4	35
$BPIC13_{inc}$	7554	20.0	65533	13	1	9	123
$BPIC14_{f}$	41353	36.1	369485	9	3	9	167
$BPIC15_{1f}$	902	32.7	21656	70	5	24	50
$BPIC15_{2f}$	681	61.7	24678	82	4	36	63
$BPIC15_{3f}$	1369	60.3	43786	62	4	32	54
$BPIC15_{4f}$	860	52.4	29403	65	5	34	54
$BPIC15_{5f}$	975	45.7	30030	74	4	31	61
$BPIC17_{f}$	21861	40.1	714198	41	11	33	113
RTFMP	150370	0.2	561470	11	2	4	20
SEPSIS	1050	80.6	15214	16	3	14	185

Table 3: Results of Exp. 1

Evt. log	$L2L_{sim}$	$L2L_{SMAPE}$
BPIC12	0.87	0.13
$BPIC13_{cp}$	0.85	0.20
$BPIC13_{inc}$	0.91	0.21
$BPIC14_{f}$	0.86	0.28
$BPIC15_{f1}$	0.89	0.22
$BPIC15_{f2}$	0.82	0.09
$BPIC15_{f3}$	0.91	0.02
$BPIC15_{f4}$	0.83	0.21
$BPIC15_{f5}$	0.94	0.02
BPIC17	0.96	0.31
RTFMP	0.96	0.32
SEPSIS	0.91	0.11

4.2 Datasets

We used a dataset of model-log pairs from a recent benchmark of automated discovery methods [3]. This collection contains twelve public real-life event logs extracted from the 4TU Centre for Research Data.[4] These logs record executions of business processes from a variety of domains, such as healthcare, finance, government and IT service management. They are the *BPI Challenge* (BPIC) logs from 2012 to 2017, the *Road Traffic Fines Management Process* (RTFMP) log, and the *SEPSIS Cases* log.[5]

Table 2 reports the logs characteristics. These logs are widely heterogeneous ranging from simple to very complex, with a log size ranging from 681 traces (for the BPIC152f log) to 150,370 traces (for the RTFMP log). A similar variety can be observed in the percentage of distinct traces, ranging from 0.2% to 80.6% of the total number of traces, and the number of event classes (i.e., the activities executed within the process), ranging from 7 to 82. Finally, the length of a trace also varies from very short, with traces containing only one event, to very long, with traces containing 185 events.

For each log, we used the process model obtained by Split Miner (SM) and Inductive Miner (IM), both with noise filtering enabled and default parameters, as per the benchmark in [3]. These two methods strike different tradeoffs between fitness and precision: IM tends to create models with higher fitness but low precision; SM tends to create smaller models with an overall higher F-Score (the harmonic mean of fitness and precision), though with lower fitness. This led to a total of 24 log-model pairs for our evaluation.

4.3 Results

Tables 3 and 4 show the results of the two experiments, respectively. From Exp. 1 we can see that using a perfectly accurate input (the distinct traces of the original log), the average loss of accuracy is only 10% (average $L2L_{sim}$ = 0.901, min = 0.82, max = 0.96). This is consistent with the events elapsed time, which is relatively low across all twelve logs (average $L2L_{SMAPE}$ = 17.6%, mix = 2%, max = 28%). Overall, these results indicate the

[4] https://data.4tu.nl/repository/collection:event_logs_real

[5] Seven of these logs, namely the BPIC14 log, the five BPIC15 logs and the BPIC17 log, were filtered in [3] using the technique in [8] to remove infrequent behavior. We kept this filtering to be able to use the models associated with these logs in the benchmark dataset.

Table 4: Results of Exp. 2 with Split Miner (SM) and Inductive Miner (IM)

	SM-mined model		SM-based output log		IM-mined model		IM-based output log	
Source log	Fitness	Precision	$L2L_{sim}$	$L2L_{SMAPE}$	Fitness	Precision	$L2L_{sim}$	$L2L_{SMAPE}$
BPIC12	0.97	0.72	0.83	0.44	0.98	0.50	0.82	0.42
BPIC13cp	0.90	0.93	0.78	0.41	0.82	1.00	0.78	0.40
BPIC13inc	0.98	0.92	0.92	0.61	0.92	0.54	0.89	0.57
BPIC14f	0.77	0.84	0.77	0.40	0.89	0.64	0.77	0.42
BPIC15f1	0.90	0.88	0.81	0.30	0.97	0.57	0.79	0.35
BPIC15f2	0.77	0.90	0.77	0.12	0.93	0.56	0.74	0.21
BPIC15f3	0.94	0.78	0.85	0.18	0.95	0.55	0.82	0.15
BPIC15f4	0.73	0.91	0.77	0.33	0.96	0.58	0.74	0.30
BPIC15f5	0.79	0.94	0.90	0.14	0.94	0.18	0.60	0.32
BPIC17	0.96	0.81	0.94	0.45	0.98	0.70	0.92	0.47
RTFMP	1.00	0.97	0.96	0.47	0.99	0.70	0.93	0.52
SEPSIS	0.76	0.77	0.89	0.27	0.99	0.45	0.72	0.40

robustness of the specific optimization technique chosen (multi-level simulated annealing), which proves to be appropriate for the problem at hand. These results were achieved by setting the initial temperature to 1,000 and the maximum number of steps to 100.

As expected, when comparing the results of the two experiments (cf. Tables 3 and 4), the logs generated in Exp. 1 using as input the distinct log traces of the original log have higher $L2L_{sim}$ and lower $L2L_{SMAPE}$ values than the logs generated in Exp. 2 using a process model as input. However, the loss in $L2L_{sim}$ between the two experiments is only 4.42% on average, barring a modest increase in $L2L_{SMAPE}$ (16.67% on average). These differences are attributable to the fact that the models used as input are not perfectly accurate. Specifically, fitness and precision affect the $L2L_{sim}$ measure negatively. Given that in general the precision of IM is much lower than that of SM, while its fitness is slightly higher, we obtain better results both in terms of $L2L_{sim}$ and $L2L_{SMAPE}$ when using as input the models discovered by SM. For example, the precision of the SM model for BPIC15f5 is 0.94 as opposed to 0.18 in the case of IM, while the two fitness measures are much closer to each other (0.79 for SM, 0.94 for IM). A very low precision as in the case of IM for this log, provides a large number of possibilities to replay the process and thus to correlate events in ways that are different than those in the original log. In the specific case of the BPIC15f5 log, this leads to a difference of 30% in $L2L_{sim}$ between SM and IM.

Leaving aside the specific differences between SM and IM, the average $L2L_{sim}$ across all 24 model-log pairs is still relatively high (0.82), which means that in most cases we can correlate events correctly. The average $L2L_{SMAPE}$ is also relatively low (0.36), meaning that the event times in the generated log deviate by 36% on average from those of the original log. In other words, we correctly assign events to their specific cases on 64% of the cases on average.

Table 5 compares the results of the second experiment with DCI using the models generated by SM. This table also reports the time performance of the two approaches. Looking at the accuracy, we can see that EC-SA outperforms DCI in all except three logs where $L2L_{sim}$ is higher for DCI, and one log where $L2L_{SMAPE}$ is lower for DCI. In those logs where EC-SA outperforms DCI, the differences in $L2L_{sim}$ range from small to substantial increases. For example, in the case of the BPIC15f5 log, our approach's

Table 5: Results of Exp. 2 with EC-SA and DCI (using the SM-mined model)

	EC-SA output (SM)			DCI output (SM)		
Source log	$\mathrm{L2L_{sim}}$	$\mathrm{L2L_{SMAPE}}$	Exec. [h]	$\mathrm{L2L_{sim}}$	$\mathrm{L2L_{SMAPE}}$	Exec. [h]
BPIC12	0.83	**0.44**	5	**0.89**	0.47	18.5
BPIC13cp	0.78	0.41	1.1	**0.81**	**0.30**	5
BPIC13inc	**0.92**	**0.61**	2.2	0.77	0.50	19
BPIC14f	**0.77**	**0.40**	2.7			>24
BPIC15f1	**0.81**	**0.30**	5.2	**0.81**	0.43	21.9
BPIC15f2	**0.77**	**0.12**	4	0.71	0.54	22.3
BPIC15f3	0.85	**0.18**	6	**0.89**	0.20	23.7
BPIC15f4	**0.77**	**0.33**	6	0.71	0.53	22.7
BPIC15f5	**0.90**	**0.14**	5	0.74	0.50	23.5
BPIC17	**0.94**	**0.45**	8			>24
RTFMP	**0.96**	**0.47**	7			>24
SEPSIS	**0.89**	**0.27**	1.6	0.84	0.25	17

$\mathrm{L2L_{sim}}$ is 22% higher than that of DCI. In this case, the discovered model has a fitness of 0.79. As DCI strictly depends on the process model behavior, it cannot assign the deviating events to any case, thus around 22% of the events will not be correlated and thus excluded from the generated log. EC-SA handles this problem by randomly assigning these events to one of the open cases, i.e. cases started before the event occurrence. On the contrary, when the model has very high fitness, DCI is able to correlate all the events to their possible cases because it builds a complete representation of the solution space, which will thus include the optimal solution. On the other hand, EC-SA's degree of randomness may lead to escaping the global optimum if the number of steps set is not sufficiently high, for the model-log at hand. As a result, DCI can have a slightly higher $\mathrm{L2L_{sim}}$ than EC-SA, as in the case of BPIC12 where the discovered model has fitness of 0.97, and DCI's $\mathrm{L2L_{sim}}$ is 6% higher than that of EC-SA. In reality, though, we cannot assume the input model to be highly fitting. Rather, we expect this model not to be very accurate both in terms of fitness and precision, given that it would be a model created manually by process analysts through interviews and workshops (so it may be biased towards the perspective of particular process participants), and may in addition be out-of-date.

Looking at the time performance, we can observe that DCI suffers from significant performance issues as it takes close to 20 hours for the majority of logs, timing out at 24 hours for three logs. Specifically, DCI takes $4\times$ the average execution time of EC-SA. In effect, DCI requires as input extra information such as minimum, average and maximum execution time for each activity. For our evaluation, we calculated this heuristic data based on the three quartiles of the activities execution time in the original log. The quality of the DCI output is affected by the quality of its inputs. The heuristic data affects the $\mathrm{L2L_{SMAPE}}$ and $\mathrm{L2L_{sim}}$ because it is used to prune the various correlation possibilities assessed by the approach.

5 Conclusion

We presented a novel approach called EC-SA to address the problem of correlating events that belong to the same case. Our approach uses multi-level objective simulated annealing

for mapping each event to a case. For optimization, we use trace alignment cost and activity execution time variance. Our evaluation in terms of log-to-log similarity, symmetric mean absolute error of event elapsed times and overall time performance on a range of real-life model-log pairs shows that our approach outperforms the state of the art. A possible avenue for future work is to include the payload of uncorrelated events to improve correlation accuracy, e.g. data inputs/outputs of process activities. Another avenue for future work is to explore different forms of process knowledge as input to EC-SA, e.g., declarative rules [5].

Acknowledgements. This research is partly funded by the Australian Research Council (DP180102839) and by the EU H2020 programme under agreement 645751 (RISE_BPM).

References

1. Adriansyah, A., van Dongen, B., van der Aalst, W.: Conformance checking using cost-based fitness analysis. In: Proc. of EDOC. IEEE (2011)
2. Askarzadeh, A., dos Santos Coelho, L., Klein, C., Cocco Mariani, V.: A population-based simulated annealing algorithm for global optimization. In: Prof. of SMC. IEEE (2016)
3. Augusto, A., Conforti, R., Dumas, M., La Rosa, M., Maggi, F., Marrella, A., Mecella, M., Soo, A.: Automated discovery of process models from event logs: Review and benchmark. IEEE TKDE **31**(4) (2019)
4. Augusto, A., Conforti, R., Dumas, M., La Rosa, M., Polyvyanyy, A.: Split Miner: Automated discovery of accurate and simple business process models from event logs. Knowledge and Information Systems (2018)
5. Baier, T., Di Ciccio, C., Mendling, J., Weske, M.: Matching events and activities by integrating behavioral aspects and label analysis. SoSyM **17**(2), 573–598 (2018)
6. Bala, S., Mendling, J., Schimak, M., Queteschiner, P.: Case and activity identification for mining process models from middleware. In: Proc. of PoEM. Springer (2018)
7. Bayomie, D., Awad, A., Ezat, E.: Correlating unlabeled events from cyclic business processes execution. In: Proc. of CAiSE. Springer (2016)
8. Conforti, R., La Rosa, M., ter Hofstede, A.: Filtering out infrequent behavior from business process event logs. IEEE TKDE **29**(2) (2017)
9. Ferreira, D., Gillblad, D.: Discovering process models from unlabelled event logs. In: Proc. of BPM. Springer (2009)
10. Leemans, S., Fahland, D., van der Aalst, W.: Discovering block-structured process models from event logs containing infrequent behaviour. In: Proc. of BPM Workshops. Springer (2014)
11. Meroni, G., Di Ciccio, C., Mendling, J.: An artifact-driven approach to monitor business processes through real-world objects. In: Proc. of ICSOC. Springer (2017)
12. Navarro, G.: A guided tour to approximate string matching. ACM Comput. Surv. **33**(1) (2001)
13. Nezhad, H., Saint-Paul, R., Casati, F., Benatallah, B.: Event correlation for process discovery from web service interaction logs. VLDB J. **20**(3) (2011)
14. Pourmirza, S., Dijkman, R., Grefen, P.: Correlation Miner: Mining business process models and event correlations without case identifiers. IJCIS **26**(02) (2017)
15. Reguieg, H., Toumani, F., Nezhad, H., Benatallah, B.: Using MapReduce to scale events correlation discovery for business processes mining. In: Proc. of BPM. Springer (2012)
16. Soffer, P., Hinze, A., Koschmider, A., Ziekow, H., et al.: From event streams to process models and back: Challenges and opportunities. Inf. Syst. **81**, 181–200 (Mar 2019)
17. van der Aalst, W.: Process Mining – Data science in action. 2nd Edition, Springer (2016)
18. Walicki, M., Ferreira, D.: Sequence partitioning for process mining with unlabeled event logs. DKE **70**(10) (2011)