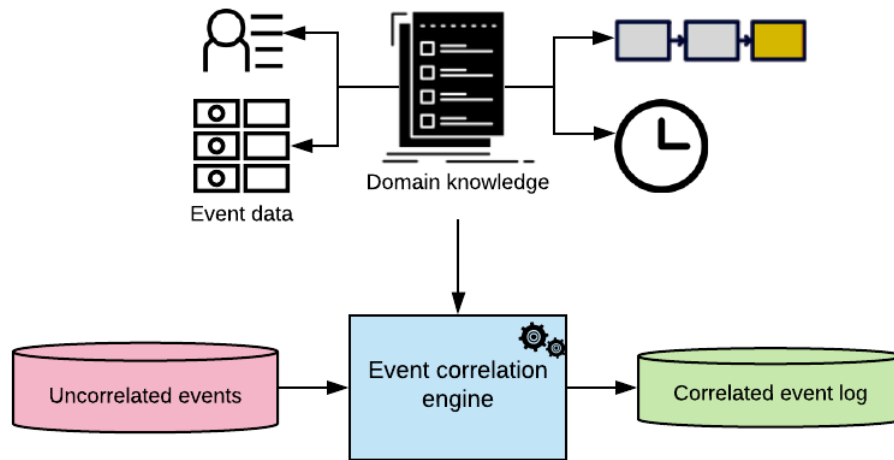


Graphical Abstract

Event-Case Correlation for Process Mining using Probabilistic Optimization

Dina Bayomie, Claudio Di Ciccio, Jan Mendling



Highlights

Event-Case Correlation for Process Mining using Probabilistic Optimization

Dina Bayomie, Claudio Di Ciccio, Jan Mendling

- We propose a novel event correlation engine: EC-SA-Data
- EC-SA-Data correlates events based on information pertaining to the data perspective in addition to the control-flow perspective
- We propose eight similarity measures to evaluate the accuracy of our technique

Event-Case Correlation for Process Mining using Probabilistic Optimization

Dina Bayomie^{a,b,*}, Claudio Di Ciccio^c and Jan Mendling^d

^aVienna University of Economics and Business, Welthandelsplatz 1, Vienna, 1020, Austria

^bCairo University, Gamaa Street 1, Giza, 12613, Egypt

^cSapienza University of Rome, Viale Regina Elena 295, Rome, 00161, Italy

^dHumboldt-Universität zu Berlin, Unter den Linden 6, Berlin, 10099, Germany

ARTICLE INFO

Keywords:

Process Mining
Event correlation
Simulated annealing
Constraints
Association rules

Abstract

Process mining supports the analysis of the actual behavior and performance of business processes using event logs. An essential requirement is that every event in the log must be associated with a unique case identifier (e.g., the order ID of an order-to-cash process). In reality, however, this case identifier may not always be present, especially when logs are acquired from different systems or extracted from non-process-aware information systems. In such settings, the event log needs to be pre-processed by grouping events into cases – an operation known as event correlation. 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. Moreover, they abstract the log to activities and timestamps, and miss the opportunity to use data attributes. In this paper, we lift these assumptions and propose a new technique called EC-SA-Data based on probabilistic optimization. The technique takes as inputs a sequence of timestamped events (the log without case IDs), a process model describing the underlying business process, and constraints over the event attributes. Our approach returns an event log in which every event is associated with a case identifier. The technique allows users to incorporate rules on process knowledge and data constraints flexibly. The approach minimizes the misalignment between the generated log and the input process model, maximizes the support of the given data constraints over the correlated log, and the variance between activity durations across cases. Our experiments with various real-life datasets show the advantages of our approach over the state of the art.

1. Introduction


Recent years have seen a drastically increasing availability of process execution data from various data sources [1, 2, 3]. Process mining offers different analysis techniques that can extract business insights from these data, known as event logs. Each event in a log must have at least three attributes [4, 5]: (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 occurrence of the event, 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). Recent process mining algorithms such as α S [6], Inductive Miner [7], Evolutionary Tree Miner [8], Fodina [9], Structured Heuristic Miner [10], Split Miner [11], and the Hybrid ILP Miner [12] need all of these three attributes together for discovering a process model.

Various data infrastructures such as data lakes often give more attention to data storage than to structuring them such that process mining can be readily applied [13, 14]. Prior research has described the problem of missing case identifiers as a *correlation problem*, because the connections between different events have to be reestablished based on heuristics,

domain knowledge or payload data. In essence, the correlation problem is concerned with identifying which events belong to the same case when a unique case identifier is missing. This identification can be done by an event correlation engine (see Figure 1). An event correlation engine constructs a correlated log with case identifiers by using domain knowledge, e.g. about the process control flow, organizational resources, maximal task durations, or other data knowledge over the event attributes. 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. The main assumptions they have in common is abstracting a log on activities and timestamps with ignoring the other attributes. Moreover, some techniques assume the generative processes to be acyclic [15, 16] while others require heuristic information about the execution behavior of activities in addition to the process model [17]. Beyond that, these approaches suffer from poor efficiency and miss the opportunity to make use of data attributes.

In previous work [18], we introduced a probabilistic optimization technique called EC-SA (Events Correlation by Simulated Annealing), which is based on a simulated annealing heuristic approach. EC-SA addresses the correlation problem as a multi-level optimization problem. In this paper, we extend EC-SA to consider a broader spectrum domain knowledge for the correlation process, integrating ideas of mixing process specification paradigms [19]. We call the extension *EC-SA-Data*. Using the domain knowledge

*Corresponding author

 dina.sayed.bayomie.sobh@wu.ac.at (D. Bayomie);

claudio.diccio@uniroma1.it (C. Di Ciccio); jan.mendling@hu-berlin.de (J. Mendling)

ORCID(s): 0000-0002-2549-6407 (D. Bayomie); 0000-0001-5570-0475 (C. Di Ciccio); 0000-0002-7260-524X (J. Mendling)

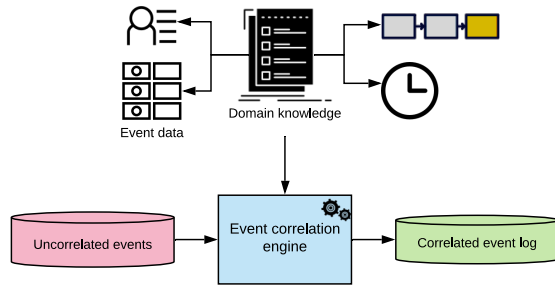


Figure 1: Overview of the event correlation engine

about the event data attributes improves the event correlation process as it decreases the random assigning of events to their corresponding cases. The technique revolves around three nested objectives. First, it seeks to minimize the misalignment between the generated log and an input process model. Second, it aims to maximize the support of the given data constraints over the correlated log. Third, it targets to minimize the activity execution time variance across cases. The latter objective builds on the assumption that the same activities tend to have similar duration across cases. Our extensive evaluation demonstrates the benefits of our novel technique.

The remainder of this paper is organized as follows. Section 2 discusses prior research on the event correlation problem. Section 3 defines preliminaries that are essential for our technique. Section 4 presents the different phases of our novel EC-SA-Data event correlation technique. Section 6 then discusses the experimental evaluation on real-life logs before Section 7 concludes the paper.

2. Related Work

The correlation problem focuses on identifying two or more variables that are related, even though this relationship is not explicitly defined. It received adequate attention in the data science research field for static data [20, 21, 22]. The existing techniques have various objectives to improve the analysis, monitoring, and mining of the data. Database correlation techniques work on improving query performance of knowledge discovery. Jermaine [23] discusses the discovery of correlations between database attributes. He develops measures of independence between the attributes to discover the actual correlation, and analyses the computational complexity of correlation techniques. Brown and Haas [24] present a data-driven technique that discovers the hidden relation between database attributes and integrates them into an optimizer to improve query performance. Though valuable for data, these works cannot be directly applied for process mining event correlation. They lack a notion of temporal order.

In this paper, we focus on a specific instance of the *event correlation* problem: how to correlate those events that have been generated from the execution of the same process instance? An event correlation technique takes as input at least an uncorrelated log [25]. Depending on whether or not

a technique relies on additional input, we classify existing techniques into four categories: 1) requiring no additional input; 2) requiring a process model; 3) relying on correlation conditions; and 4) relying on event similarity functions.

The first category of techniques uses as input only event names and ignores the other event data attributes. Ferreira and Gillbald [16] provide an Expectation-Maximization (E-Max) approach that builds a Markov model from the uncorrelated log and discovers the possible process behavior in the uncorrelated log. This technique is adversely sensitive to the density of the working cases within the system, i.e., the number of overlapping cases at a given point in time. Also, it does not support concurrency and cyclic process behaviors. Walicki and Ferreira [26] provide a sequence partitioning approach that searches for the minimal set of patterns that can represent the process behavior in an uncorrelated log. The technique does not support concurrency and cyclic process behaviors.

The second category includes correlation techniques that use timestamps of events and the process model. The Correlation Miner (CMiner) by Pourmiza et al. [27] builds on two matrices, one capturing proceed/succeed relations and another one capturing the time difference between pairs of events. An optimal correlation is calculated using integer linear programming. An extension of the CMiner [15] uses quadratic programming to find the optimal correlation by minimizing the duration between the events. CMiner does not support cyclic processes and relies on quadratic constraints solving, which limits its scalability. Experiments reported in [15] show that the approach can only handle logs with a few dozen cases. The Deducing Case Ids (DCI) approach [28] requires a process model and heuristic information about the activity execution durations. The approach utilizes a breadth-first approach to build a case decision tree to explore the solution space. In [17], DCI is extended with a pre-processing step to detect the cyclic behavior and build a relationship matrix for the correlation decision. DCI supports cyclic processes. It is sensitive to the quality of the input data, such that if the model and the log have low fitness, then the generated correlated logs will contain noise and missing events. Also, it is computationally inefficient due to the breadth-first search approach. In a previous paper [18], we propose the Event Correlation by Simulated Annealing (EC-SA) approach, which uses the event names and timestamp in addition to the process model. EC-SA addresses the correlation problem as a multi-level optimization problem, as it searches for the nearest optimal correlated log considering the fitness with an input process model and the activities' timed behavior within the log. The accuracy of the given model affects the quality of the correlated log, and the performance is affected by the number of uncorrelated events.

The third category includes correlation techniques that use event data attributes and apply correlation conditions to correlate the events. Motahari-Nezhad et al. [29] propose a semi-automated correlation approach to correlate the web service messages based on the correlation conditions. The approach derives correlation conditions using the event data

from different data layers. Also, it computes the interestingness of the attributes of the events to prune the conditions search space. Thus, it generates several log partitions and possible process views. The approach requires user-defined domain parameters and intermediate domain expert feedback to guide the correlation process. Engel et al. [30] propose the EDImine framework, which allows for the usage of process mining over electronic data interchange (EDI) messages. EDIminer resorts to message flow mining and physical activity mining methods to generate the events from EDI messages. Then, EDIminer employs user-defined correlation rules to correlate the events to their cases. This framework is limited to EDI messages and depends on the user-defined parameters for the event generation and the correlation process. Reguieg et al. [31] propose a MapReduce-based approach to derive the correlation conditions from the service interaction logs. It consists of two stages. The first stage defines the simple correlation conditions. The second stage derives more complex correlation conditions and correlates the events to their cases. In [32], the authors extend their previous work to improve scalability and efficiency. They introduce two strategies to perform the log partition and explore the complex correlation condition space. The main challenges of the approach are the log partitioning and the vast amount of network traffic communication. Cheng et al. [33] propose the Rule Filtering and Graph Partitioning (RF-Grap) approach, which follows the filtering and verification principle to improve the efficiency of event correlation using distributed platforms. RF-Grap prunes a large number of uninteresting correlation rules in the filtering step. Accordingly, not all the derived rules are investigated, but only the interesting rules that fulfil the criteria. In the verification steps, they use graph partitioning to decompose the correlation possibilities over the clusters. De Murillas et al. [34] propose an approach to extract the event log from a database based on the redo logs that contain the events of data manipulation. They use a data model to define the relation between the events. In [35], the authors provide a way to automatically generate different event logs from a database by defining the case notion based on the data relations in the data model. A case notion defines which events should be considered for the correlation based on the selected data objects that represent the investigated cases. They measure the interestingness of the generated logs and recommend the highest one to the user.

The fourth category includes the correlation techniques that rely on event similarity or the case identifier in the log. Djedović et al. [36] propose an algorithm to compute the similarity between pairs of events to correlate events with higher similarity. Event similarity function defined over the equality between the attributes over the events. The optimal correlation represents the highest similarity score between the case's events. The approach expects the existence of main attributes that do not change over the case. Abbad Andaloussi et al. [37] address the correlation problem under the assumption that event data already contains the case identifier. For each event log attribute A, the technique discovers a process model assuming that A is the case identifier. The resulting

models are compared based on four quality measures. The attribute that yields the highest-quality model is taken as the case identifier. Bala et al. [13] follow a similar direction based on the idea that identifiers are repetitive in the log. Burattin and Vigo [38] propose a framework that generalized from a real business case. They search the activities attributes over the log to define the possible process instance attributes. Then they used the equality relation between these attributes to correlate the events to their cases. This method relies on a-prior knowledge about the application domain and user-defined heuristic parameters, such as the number of events within a case and characteristics of the candidate attributes.

In summary, the approaches in the first category assume that the process is acyclic. Those in the second category expect that a full process model is available. The third category require correlation rules to be provided or discovered heuristically. Approaches in the fourth category assume that there is a case identifier attribute or a similarity function that allows to group events. The technique presented in this paper extends EC-SA approach. It integrates the strengths of the first and second category. Unlike the first category, it can handle cyclic behavior. It also allows users to provide additional domain knowledge that will support the correlation process. In this way, it relaxes the dependence on the control-flow knowledge.

3. Preliminaries

In this section, we discuss the fundamental notions that our approach builds upon. Section 3.1 describes the data structures we handle. Section 3.2 outlines our process modeling and execution language and notations. Section 3.3 illustrates the fundamental mechanisms underpinning simulated annealing, a core technique for our solution.

3.1. Process Event Data Structures

Starting with the basic notion of event (i.e., the atomic unit of execution), we introduce the uncorrelated event log, case and projection of a case over an event attribute. Thereupon, we present the definitions of event log and trace.

Definition 1 (Event). Let $E \ni e$ be a finite non-empty set of symbols. We refer to e as *event* and to E as the *universe of events*.

Definition 2 (Attribute). Given a non-empty set of domain values $\text{Dom} \in \mathfrak{D}$, an *attribute* $\text{Attr} \in \mathfrak{A}$ is a partial function $\text{Attr} : E \rightarrow \text{Dom}$ mapping events to domain values. We indicate the value mapped by Attr to an event e by using a dot notation, i.e., $e.\text{Attr}$.

In the following, we assume without loss of generality that attributes Act and Ts are *total* functions defined over events. The range of the former is a finite subset of strings we interpret as *activity* names and the range of the latter is a finite subset of integers to be interpreted as *timestamps*. For example, e_1 in Fig. 2 is mapped to four different values, one per attribute: $e_1.\text{Act} = A$ represents the executed activity, $e_1.\text{Ts} = "07/06/2020 09:00"$ represents the completion

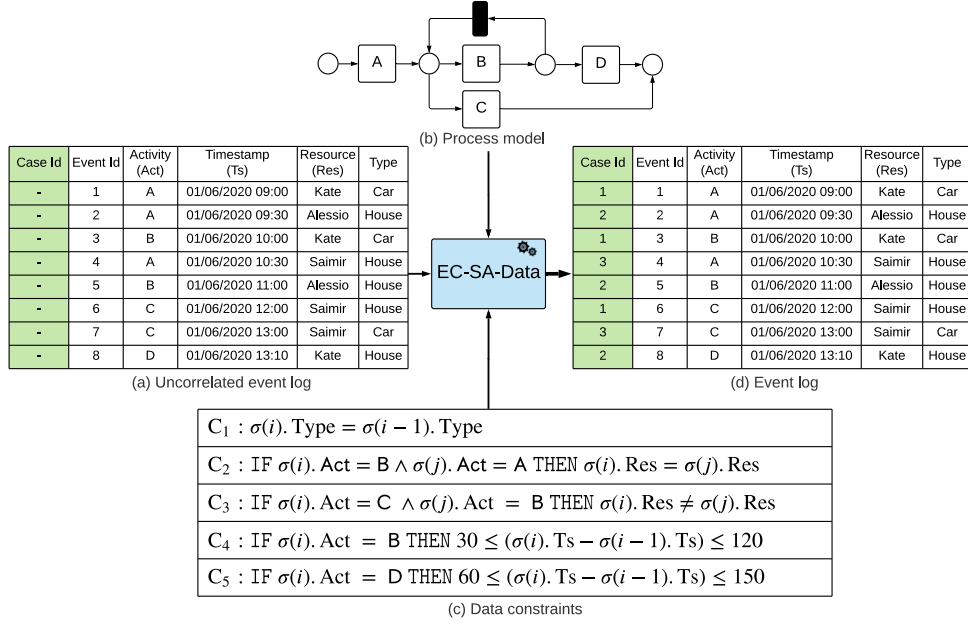


Figure 2: Running example of a sample loan application check process

timestamp, $e_1.\text{Res} = \text{Kate}$ represents the operating resource, and $e_1.\text{Type} = \text{Home}$ represents additional data knowledge about the event context.

Definition 3 (Uncorrelated log). Let $E \ni e$ be the universe of events, $\mathcal{D} \ni \text{Dom}$ the universe of domain values and $\mathcal{A} \ni \text{Attr}$ a set of attributes defined over E as per Def.s 1 and 2. Given a total order defined over E , $\leq \subseteq E \times E$ (henceforth referred to as *event ordering*), the *uncorrelated log* is a tuple $\text{UL} = (E, \mathcal{D}, \mathcal{A}, \leq)$.

We assume the mapping of Ts to be coherent with \leq , i.e., if $e \leq e'$ then $e.\text{Ts} \leq e'.\text{Ts}$. Considering the total ordering as a mapping from a convex subset of integers, we can assign to every event a unique integer index (or *event id* for short), induced by \leq on the events. We shall denote the index $i \in [1, |E|]$ of an event e as a subscript, e_i . For example, Fig. 2(a) depicts an uncorrelated log and e_3 is its third event.

Definition 4 (Event log). Let $\text{UL} = (E, \mathcal{D}, \mathcal{A}, \leq)$ be an uncorrelated log as per Def. 3 and $I \ni \iota$ be a universe of *case identifiers* (or *case id's* for short). An event log is a triple $L = (\text{UL}, I, \ell)$ where $\ell : E \rightarrow I$ is a total surjective function.

Notice that ℓ induces a partition of E into $|I|$ subsets. Figure 2(d) illustrates an event log. Notice that $I = \{1, 2, 3\}$ and ℓ maps events e_1, e_3 and e_6 to 1, e_2, e_5 and e_8 to 2, and e_4 and e_7 to 3. We name the sequences of events that stem from mapping ℓ and preserve \leq as *cases*.

Definition 5 (Case). Given a case id $\iota \in I$ and an event log L as per Def. 4, a *case* σ defined by ι over L is a finite sequence $\langle e_{\sigma,1}, \dots, e_{\sigma,n} \rangle \in E^*$ of length $|\sigma| = n \in \mathbb{N}$ of events $e_{\sigma,i}$ with $1 \leq i \leq n$ such that (i) $\ell(e_{\sigma,i}) = \iota$ and (ii)

the sequence is induced by \leq , i.e., $e_{\sigma,j} \leq e_{\sigma,i}$ for every $j \leq i \leq n$.

For example, the event log depicted in Fig. 2(d) is comprised of 3 cases. Case σ_1 defined by case id 1 is $\langle e_1, e_3, e_6 \rangle$. Notice that it preserves the order of the events within the case. We name the projection of a case over the activities of its events as *trace*.

Definition 6 (Trace). Given a case $\sigma = \langle e_{\sigma,1}, \dots, e_{\sigma,n} \rangle \in E^*$ and the total attribute function $\text{Act} : E \rightarrow \text{Dom}_{\text{Act}}$, a trace $t \in \text{Dom}_{\text{Act}}^*$ is the sequence induced by case σ through the mapping of Act, i.e., $t = \langle e_{\sigma,1}.\text{Act}, \dots, e_{\sigma,n}.\text{Act} \rangle$.

In our example, the trace corresponding to σ_1 is $\langle A, B, C \rangle$.

Short-hand notations

For the sake of readability, we shall use the following short-hand notations:

$L(\iota)$ indicates the case defined by ι over L ; in the example of Fig. 2(d), e.g., $L(2) = \langle e_2, e_5, e_8 \rangle$;

$S(L)$ denotes the set of all cases defined over event log L , i.e., $S(L) = \{L(\iota) | \iota \in I\}$; it follows that $|I| = |S(L)|$; in our example, $S(L) = \{\sigma_1, \sigma_2, \sigma_3\}$ where $\sigma_1 = \langle e_1, e_3, e_6 \rangle$, $\sigma_2 = \langle e_2, e_5, e_8 \rangle$, $\sigma_3 = \langle e_4, e_7 \rangle$;

$\sigma(i)$ refers to the i -th event within a case σ (e.g., the first event in σ_1 is denoted as $\sigma_1(1)$, whereby $\sigma_1(1) = e_1$ in our example);

$\sigma[i, j]$ indicates the segment of case σ from i to j , having $1 \leq i \leq j \leq |\sigma|$ (e.g., $\sigma_1[2, 3] = \langle e_3, e_6 \rangle$);

$\text{Act}(\sigma)$ denotes with a slight abuse of notation the trace induced by case σ through the mapping of Act (e.g., $\text{Act}(\sigma_1) = \langle A, B, C \rangle$);

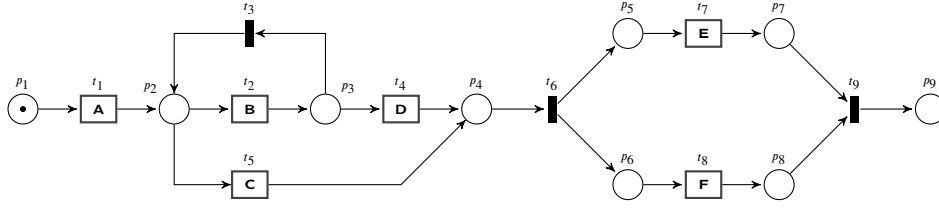


Figure 3: A workflow net

$e \in \sigma$ indicates that there exists an index i , $1 \leq i \leq n$ such that $\sigma(i) = e$ (e.g., $e_5 \in \sigma_2$);

$\langle e, e', e'', \dots, e^{(m)} \rangle \subseteq \sigma$ indicates that there exists an index j with $1 \leq j \leq n - m$ such that $\sigma(j) = e, \sigma(j+1) = e', \sigma(j+2) = e'', \dots, \sigma(j+m) = e^{(m)}$, where n is the length of σ and $m \leq n$ (e.g., $\langle e_2, e_5 \rangle \subseteq \sigma_2$);

$\sigma = \sigma'$ holds if and only if $\sigma \subseteq \sigma'$ and $\sigma' \subseteq \sigma$.

3.2. Process Modeling and Execution

In this section we outline the main notions we shall adopt in the remainder of this paper about process modeling and execution.

Alongside uncorrelated event logs and data constraints, our approach takes as input a behavioral model for processes. We require that the process model has a starting activity and an accepting state. In the context of this paper, we resort to workflow nets [39] such as the one in Fig. 3. Workflow nets are bipartite graphs consisting of (i) a finite non-empty set of nodes partitioned into places (graphically depicted as circles – e.g., $\{p_1, \dots, p_9\}$ in Fig. 3) and transitions (rectangles – e.g., $\{t_1, \dots, t_9\}$), and (ii) a flow relation, namely arcs connecting places to transitions and transitions to places (e.g., $p_1 \rightarrow t_1$, $t_1 \rightarrow p_2$ and $t_5 \rightarrow p_2$); the set of places at the tail of flow arcs toward a transition are the *preset* of that transition (e.g., the presets of t_2 and t_9 are $\{p_2\}$ and $\{p_7, p_8\}$, respectively); places at the head of flow arcs are the *postset* of the transition (e.g., the postsets of t_2 and t_6 are $\{p_3\}$ and $\{p_5, p_6\}$, respectively).

Transitions can be annotated with an activity name (e.g., t_1 is annotated with A); otherwise, they are *silent* (in which case, they are graphically depicted as a solid black rectangle as with t_5, t_6 and t_9 in the figure). The set of places contain exactly one *input* and one *output* place, namely a source and a sink node for the graph, respectively – e.g., p_1 and p_9 . Every node in the workflow net is reachable through a directed walk from the input place.

The execution semantics are specified by the production and consumption of *tokens* (represented as black dots). A transition can be executed (i.e., it is *enabled*) only if a token is in every place of its preset. The execution implies that a token is consumed from every place in its preset and a token is produced in every place in its postset. In the beginning, a token resides in the input place (p_1 in the figure) by default – therefore, it is sometimes omitted from the graphical representation. Transition t_1 , in the example, is the only one enabled at the beginning. Its execution consumes the token from p_1 and produces a token in p_2 . Then, both t_2 and

t_3 are enabled. Notice that their execution is thus *mutually exclusive* and that t_3 enacts a *cyclic* behavior. Executing t_2 and then t_4 enables t_6 which, in turn, can consume a token from p_4 and produce two tokens (one in p_5 and the other in p_6), thereby enabling both t_7 and t_8 . Notice that t_9 is enabled only if a token is assigned to p_7 and another one to p_8 . Therefore, t_7 and t_8 have to be executed, regardless of the order. The execution of t_9 consumes a token from p_7 and a token from p_8 to ultimately produce one in p_9 , the output place. The sequence of executions of enabled transitions from the initial state (with a token in the input place) to the final one (with a token in the output place) is a *run* of the Workflow net. The physical time it takes for a run to complete, from start to end, is called *cycle time*.

As previously discussed, a *trace* is a sequence of activities. The transcription of activities decorating the sequentially executed transitions determines a trace too. Notice that the execution of silent transitions such as t_5 in the example does not occur in a trace. Traces that correspond to runs of the Workflow net in Fig. 3 are, e.g., $\langle A, C, E, F \rangle$, $\langle A, B, B, D, E, F \rangle$ and $\langle A, B, D, F, E \rangle$.

Traces can be replayed to check if they correspond to a run of a Workflow net. If an activity in the sequence cannot be bound to the execution of an enabled transition, or requires one or more (non-silent) transitions to be previously executed though they are not recorded in the trace, we say an *asynchronous move* occurs. The computation of those *alignments* [40] is at the basis of a well-known technique for conformance checking [41]. For example, $\langle A, B, B, B, D, F, E \rangle$ conforms with the process model in Fig. 3, thus the alignment consists of sole synchronous moves. Instead, $\langle A, B, C, E, F \rangle$ does not conform with it as B is a move in the log that cannot correspond to a move in the model. Similarly, $\langle A, E, F \rangle$ requires the execution of transition C before the occurrence of E although it is not recorded in the trace. Intuitively, the fewer asynchronous moves occur, the higher the fitness of model and log is. The computation of alignments and quality measures for process mining are beyond the scope of this paper. The interested reader can find a detailed examination of these topics in [41].

3.3. Simulated Annealing

We address the correlation problem as an optimization problem and resort to simulated annealing to solve it. *Simulated Annealing* (SA) is a metaheuristic algorithm that explores the optimization problem's search space to find the nearest approximate global solution by simulating the cooling process of metals through the annealing process [42, 43]. SA

applies the stochastic perturbation theory [44] to search for an approximate global solution: by randomly changing the next individuals, it aims to skip the iterations' local optimal solution [45]. Reportedly, SA has been widely adopted in areas that are conceptually close to ours, i.e., with resource-allocation [46], project schedule [47], and matching optimization problems [48, 49].

In particular, we focus on the *population-based* SA [50], as it allows for the use of multiple individuals in the same iteration of the annealing process. SA explores the search space through the following steps. First, it starts by randomly generating an initial non-empty *population* (pop), namely a non-empty set of *individuals* ($x \in \text{pop}$, with $|\text{pop}| \geq 1$). Then, the algorithm initializes the current step S_{curr} as $S_{\text{curr}} = 1$ and the current *temperature* with a given initial value, $\tau_{\text{curr}} = \tau_{\text{init}}$. The annealing starts with a high temperature τ_{init} , then cools it down over the iterations. The initial temperature τ_{init} thus represents the highest temperature reached during the execution of the algorithm. The algorithm generates a neighbor solution x' for the current individual x by randomly changing some part of the current individual x . Notice that SA disregards the historical individuals and only focuses on x and x' . Therefore, it is considered as a memory-less algorithm. Next, SA computes the *energy cost function* based on x and x' , namely $\delta f_c(x, x')$.

The *acceptance probability* of the new neighbor solution $\text{prob}(x')$ is computed using $\delta f_c(x, x')$ and the current temperature τ_{curr} . In particular, $\text{prob}(x')$ determines whether the new neighbor, x' , can be used as the next individual. Notice that $\text{prob}(x')$ may select x' even though it performs worse than x to increase the chances of skipping the local optimum and let the algorithm explore the search space further. This phenomenon is more likely to occur with high temperatures. At each iteration, SA compares the (current) global optimal solution x_G (i.e., the best solution over the iterations $[0, S_{\text{curr}}]$), with the local optimal solution x_L in pop at S_{curr} based on $\delta f_c(x_G, x_L)$. Therefore, SA can return the best solution over all the iterations. Finally, SA uses a *cooling schedule* [51] that defines the rate at which the temperature (τ_{curr}) cools down and increments S_{curr} by 1. The cooling schedule permits SA to explore the search space more freely at the beginning of the annealing process, while it restricts the range of visited solutions over the iterations. The reason is, the lower the temperature gets, the less likely it becomes to prefer an individual x' of lower quality over x for the next iteration. Different cooling schedules [51], including the exponential schedule, linear schedule, and logarithmic schedule. SA repeats the annealing and cooling process until S_{curr} reaches the given maximum number of iterations (S_{max}). To sum up, 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 cooling schedule, (ii) the creation of a new neighbor, x' , (iii) the energy cost, $\delta f_c(x, x')$, and (iv) the acceptance probability, $\text{prob}(x')$.

4. The EC-SA-Data Solution

Equipped with the definitions and main notions defined in the above section, we define the input (I1, I2, I3), preconditions (P1, P2), output (O1) and effects (E1) of EC-SA-Data.

- I1.** An uncorrelated log UL, as per Def. 3 (depicted, e.g., in Fig. 2(a)).
- I2.** A process model (e.g., the Workflow net depicted in Fig. 2(b)).
- P1.** The process model is required to have exactly one start activity (e.g., A in Fig. 2(b)), which is enabled only at the beginning of the run, and thus cannot be part of any cycle.
- P2.** The process model is required to have a final state (such a state, e.g., is reached after C or D are executed with the model in Fig. 2(b)).
- I3.** A set of domain knowledge rules, i.e., data constraints on process data $\mathfrak{C} = \{C_1, \dots, C_m\}$. In the following, we will interchangeably use the terms “rule” and “data constraint”. Constraints are propositions [52] exerted on resources, time or additional event attributes. Syntax and semantics of the constraint expression language will be described in Section 4.3. Figure 2(c), e.g., illustrates four such constraints.
- O1.** EC-SA-Data generates an event log L as per Def. 4.
- E1.** The event log partitions UL into a set of cases, i.e., for every event $e \in \text{UL}$ there exists one and only one case $\sigma \in L$ s.t. $e \in \sigma$ (see, e.g., Fig. 2(d)).

The key idea of the EC-SA-Data technique is to treat the event correlation problem as a multi-level optimization problem. EC-SA-Data has three nested objectives: (i) minimizing the misalignment between the generated event log and the input process model, (ii) minimizing the constraint violations over the cases, and (iii) minimizing the activity execution time variance across the cases.

EC-SA-Data employs simulated annealing (SA) as the optimization technique [50], to find an *approximate* global optimal correlated log in a reasonable time. As discussed in Section 3.3, SA is a metaheuristic algorithm that requires the definition of the following four functions:

- (i) The cooling schedule τ_{curr} ;
- (ii) The creation of a new neighbor (x');
- (iii) The energy cost ($\delta f_c(x, x')$);
- (iv) The acceptance probability ($\text{prob}(x')$).

Figure 4 shows the steps of EC-SA-Data. We explain them in detail in the following subsections, together with the definition of the functions required by the metaheuristic algorithm of SA (i.e., cooling schedule, creation of a new neighbor, energy cost, and acceptance probability – see Section 3.3).

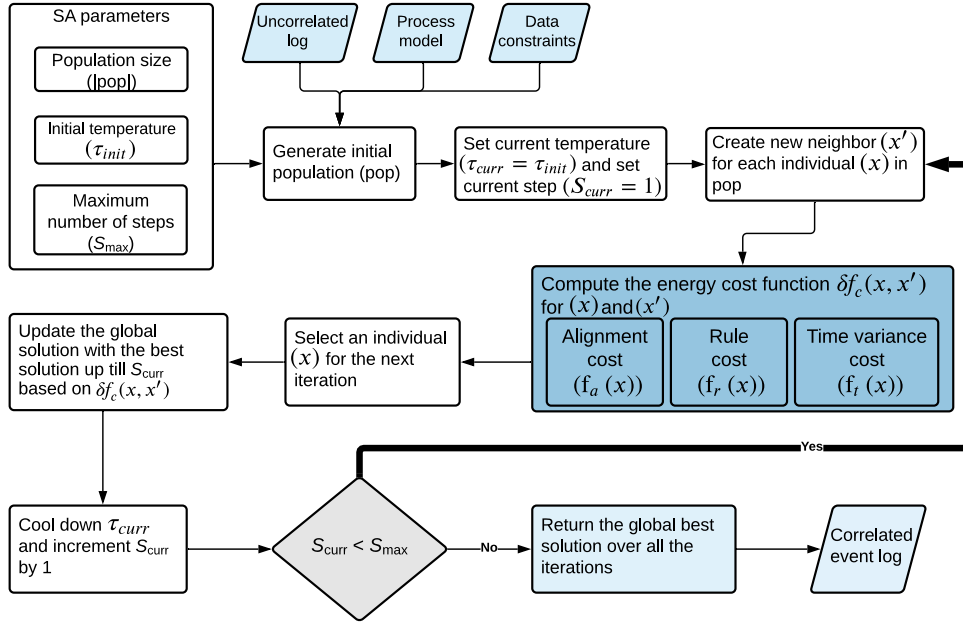
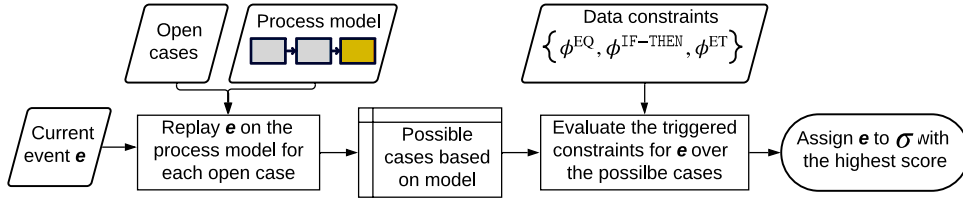
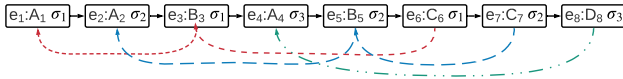


Figure 4: The EC-SA-Data technique overview


 Figure 5: Event correlation decision process for event e_i

 Figure 6: Log graph LG_1 of individual x

4.1. Initial population

As shown in Fig. 4, the first step is the generation of the initial population, pop , of size $|\text{pop}| \geq 1$. An individual $x \in \text{pop}$ is a candidate event log as defined as per Def. 4. For the sake of readability, we graphically depict the individuals as graphs such as that of Fig. 6. In the following, we shall name these graphs as *log graphs*, or *LG's* for short.

As shown in Fig. 6, every node in the log graph represents an event within the uncorrelated log, its assigned case and the related activity. For example, the third node represents e_3 , assigned with case σ_1 and annotated with B_3 as its activity e_3 . Act is B. We keep the event index as a subscript to distinguish the position in which activities recur in the log (e.g., B_3 and B_5 denote the occurrence of activity B with events e_3 and e_5). The log graph depicts two different flow connections. The first flow is represented via forward solid arcs connecting every event to its direct successor according to the temporal order in the uncorrelated log. The second flow

connection (depicted with backward dashed arcs) connects every event to its direct predecessor in the same case. For readability, we used different colors to differentiate between the cases. For example, blue backward dashed arcs connect e_7 to e_5 and e_5 to e_2 in the figure as these events belong to case $\sigma_2 = \langle e_2, e_5, e_7 \rangle$.

The data structure underlying the log graph is generated by replaying the uncorrelated event log on the process model and verifying the data constraints over the possible case assignments. This step is repeated based on the population size. In our example, we assume $|\text{pop}| = 1$ for readability purposes.

Figure 5 shows the steps taken to correlate an event e . The first step filters out the possible candidate cases for e based on the process model replay. Then, we rank the candidate cases based on the number of data constraints thereby satisfied by e_i in those cases. These steps are repeated for every event in UL.

4.2. Process model based correlation

Every run of the process model from the initial activity to the termination conditions corresponds to a case. We name the cases corresponding to non-terminated runs as *open cases*.

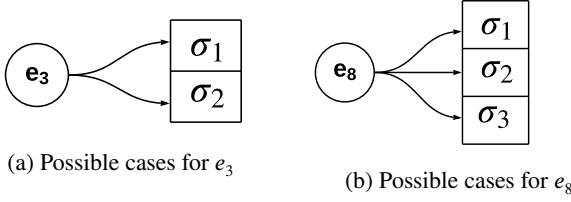


Figure 7: Process model based assignment of cases for events e_3 and e_8

We figure the following three scenarios when replaying an event e over the input process model.

1. Event 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. Event e corresponds to the execution of an enabled (non-starting) activity for one or more cases (*enabled event*). If only one run enables e , it is assigned to the case of that run. Otherwise, e is assigned to the case satisfying the highest number of data constraints.
3. Event e does not correspond to any enabled activity (*non-enabled event*). Then, e is assigned to the case satisfying the highest number of data constraints. This way, we guarantee that all events are correlated, even if the log deviates from the model.

For example, considering UL in Fig. 2(a) and the process model in Fig. 2(b), e_1 is a start event as it executes the start activity (A). Thus, it opens a new case, σ_1 . The same goes for e_2 and e_4 , which start σ_2 and σ_3 , respectively. 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 is an enabled event and both cases are considered as possible candidate assignments for the data-constraint based correlation step, as shown in Fig. 7(a). On the other hand, based on the assigned cases in Fig. 6, none of the cases σ_1 , σ_2 and σ_3 expect the execution of activity D. Thus, e_8 is a non-enabled event and all the three cases are considered for data constraints based correlation step in order to reduce the randomization of the correlation decision, as shown in Fig. 7(b). Following Fig. 5, we identify the possible case-assignments of an event based on the model that will be used in the next step.

4.3. Data constraint based correlation

Knowledge about data attributes provides an additional source of information for correlation. Intuitively, we use data constraints to filter and rank the cases to which an event can be assigned. In the context of this paper, we focus on the following types of data constraints. To define them, we take inspiration from the work of Nehzad et al. [29, 15] and extend it with IF-THEN rules. Though not intended to constitute an exhaustive set, we focus on these templates as we experimentally found them to be a good trade-off between expressiveness and tractability. Further studies on the suitability and effectiveness of the data constraints in

use are beyond the scope of this work and subject to future research.

Definition 7 (Equality constraint). Let $\sigma(i)$ and $\sigma(i-1)$ be two consecutive events in a case σ as per Def. 5. A data-attribute equality constraint (henceforth *equality constraint* for short) ϕ^{EQ} is a predicate over variables $\sigma(i)$ and $\sigma(i-1)$ formulated as follows:

$$\phi^{\text{EQ}} := \sigma(i).\text{Attr} = \sigma(i-1).\text{Attr}$$

For example, C_1 in Fig. 2 (i.e., $\sigma(i).\text{Type} = \sigma(i-1).\text{Type}$) enforces the equality between the Type attribute values. As depicted in Fig. 7(a), e_3 reports the execution of activity B. One of the possible cases, based on the process model, is σ_1 . The constraint is evaluated over $\sigma(i) = e_3$ and $\sigma(i-1) = e_1$.

Definition 8 (IF-THEN constraint). Let $\sigma(i)$ be an event in a case σ as per Def. 5 and j an index $1 \leq j < i$; let E be a universe of events and Attr an attribute $\text{Attr} : E \rightarrow \text{Dom}$ as per Def. 2. An IF-THEN constraint $\phi^{\text{IF-THEN}}$ is a predicate over pairs of events consisting of two propositional clauses: the *antecedent* (the IF clause) and *consequent* (the THEN clause). We define the syntax of well-formed IF-THEN constraints as follows.

$$\begin{aligned} \phi^{\text{IF-THEN}} &:= \text{IF } \varphi_{\text{IF}}^i \text{ THEN } \varphi_{\text{THEN}}^{i,j} \wedge j = i-1 \\ &\quad | \text{IF } \varphi_{\text{IF}}^{i,j} \text{ THEN } \varphi_{\text{THEN}}^{i,j} \wedge j < i \\ \varphi_{\text{IF}}^i &:= \sigma(i).\text{Attr} \leq a \mid \varphi_{\text{IF}}^i \wedge \sigma(i).\text{Attr} \leq a \\ \varphi_{\text{IF}}^{i,j} &:= \varphi_{\text{IF}}^i \wedge \sigma(j).\text{Attr} \leq a \\ &\quad | \varphi_{\text{IF}}^{i,j} \wedge \sigma(j).\text{Attr} \leq a \\ \varphi_{\text{THEN}}^{i,j} &:= \hat{\varphi}_{\text{THEN}}^{i,j} \mid \varphi_{\text{THEN}}^{i,j} \vee \hat{\varphi}_{\text{THEN}}^{i,j} \\ \hat{\varphi}_{\text{THEN}}^{i,j} &:= \sigma(j).\text{Attr} \leq a \mid \sigma(i).\text{Attr} \leq \sigma(j).\text{Attr}' \\ \leq &:= < \mid > \mid \geq \mid \leq \mid = \mid \neq \\ \vee &:= \vee \mid \wedge \end{aligned}$$

In the following, we may collectively refer to φ_{IF}^i and $\varphi_{\text{IF}}^{i,j}$ as φ_{IF} , and to $\varphi_{\text{THEN}}^{i,j}$ and $\hat{\varphi}_{\text{THEN}}^{i,j}$ as φ_{THEN} for the sake of readability.

$\phi^{\text{IF-THEN}}$ is an association rule in which φ_{IF} and φ_{THEN} act as selection and verification criteria, respectively. In other words, φ_{IF} seeks a pair of events, i.e., an event $\sigma(i)$ and a preceding event $\sigma(j)$, that satisfy the IF clause. EC-SA-Data gives priority to the closest j to i in the selection process. Notice that if φ_{IF} is in the φ_{IF}^i form, we impose $j = i-1$ by default (see the grammar rule above). Subsequently, φ_{THEN} is evaluated on the selected events to check whether $\sigma(i)$ and $\sigma(j)$ satisfy $\phi^{\text{IF-THEN}}$. For instance, the evaluation of C_2 in Fig. 2 (i.e., IF $\sigma(i).\text{Act} = \text{B}$ and $\sigma(j).\text{Act} = \text{A}$ THEN $\sigma(i).\text{Res} = \sigma(j).\text{Res}$) selects two events based on their activities ($\sigma(i).\text{Act} = \text{B}$ and $\sigma(j).\text{Act} = \text{A}$) and then checks whether the resources are equal for those events ($\sigma(i).\text{Res} = \sigma(j).\text{Res}$). In the example above, let us take e_3 . We notice that $e_3.\text{Act}$ is B and one of the assignable cases (based on the process model) is σ_1 as illustrated in Fig. 7(a). Considering

the log graph illustrated in Fig. 6, we pick $\sigma(2) = e_3$ and select $\sigma(1) = e_1$ as $e_1.\text{Act} = A$. Then, the THEN clause is evaluated over the two events: $(e_3.\text{Res} = e_1.\text{Res}) \equiv \text{True}$. We conclude that C_2 is satisfied by e_3 and e_1 in case σ_1 .

Definition 9 (Event-time constraint). Let $\sigma(i)$ and $\sigma(i-1)$ be two consecutive events in a case σ as per Def. 5. An event-time constraint ϕ^{ET} is a predicate over $\sigma(i)$ and $\sigma(i-1)$ consisting of two propositional clauses: an antecedent (IF clause) that selects $\sigma(i)$ and a consequent (THEN clause) that bounds the event execution duration $(\sigma(i).\text{Ts} - \sigma(i-1).\text{Ts})$ to a given minimum duration (dur) and a given maximum duration (dur') as follows.

$$\begin{aligned}\phi^{\text{ET}} &:= \text{IF } \phi_{\text{IF}}^i \text{ THEN } \phi_{\text{THEN}}^{\text{ET}} \\ \phi_{\text{THEN}}^{\text{ET}} &:= \text{dur} \leq (\sigma(i).\text{Ts} - \sigma(i-1).\text{Ts}) \leq \text{dur}'\end{aligned}$$

ϕ^{ET} is based upon the grammar of IF-THEN constraints to express rules on the activity execution duration. In particular, the IF clause selects $\sigma(i)$ based on propositional formulas, while $\sigma(j)$ is the directly preceding event $\sigma(i-1)$, taken in order to compute the event duration $(\sigma(i).\text{Ts} - \sigma(i-1).\text{Ts})$.

For example, C_4 in Fig. 2 (i.e., $\text{IF } \sigma(i).\text{Act} = B \text{ THEN } 30 \leq (\sigma(i).\text{Ts} - \sigma(i-1).\text{Ts}) \leq 120$) requires that the duration of activity B is between 30 min and 120 min. e_3 reports the execution of activity B. One of the assignable cases as per the process model is σ_1 , as illustrated in Fig. 7(a). Considering the log graph illustrated in Fig. 6, we select $\sigma(i) = e_3$ and compute the duration based on $\sigma(i-1) = e_1$, which amounts to 60 min. Thus, the THEN clause holds true as $30 \leq 60 \leq 120$.

We collectively refer to equality, IF-THEN and event-time constraints as data constraints.

Definition 10 (Data constraint). Let L be an event log defined over events in $E \ni e$ and let $\sigma \in S(L)$ be a case as per Def.s 4 and 5. A data constraint C is a predicate ϕ over events that belongs to either of the following three types: data-attribute equality constraint (as per Def. 7, hitherto indicated with the expression “ C is ϕ^{EQ} ”), IF-THEN constraint (as per Def. 8, “ C is $\phi^{\text{IF-THEN}}$ ”), or event-time constraint (Def. 9, “ C is ϕ^{ET} ”). The set of all possible constraints over E is the universe of data constraints $\mathfrak{C} \ni C$.

EC-SA-Data employs data constraints to rank possible case assignments based on the number of satisfied ones. To pursue our objective, we first need to restrict the evaluation of constraints to a current event under analysis. Therefore, we introduce the notion of i -preassignment: rather than seeking pairs of events at position i and $j < i$ in a case σ that satisfy a constraint, we fix i to an event $e \in \sigma$ so as to be able to check whether any j exists such that the constraint is satisfied by $\sigma(i)$ and $\sigma(j)$ in σ .

Definition 11 (i -Preassignment). Let L be an event log over the universe of events $E \ni e$ and $S(L) \ni \sigma$ be its case set as per Def.s 1, 4 and 5. Let $C \in \mathfrak{C}$ be a data constraint expressed by formula ϕ as per Def. 10. Denoting with $\tilde{i}(\sigma, e)$ the index of e in case σ , the i -preassignment of ϕ with e is the assignment of $\sigma(i)$ with e in its formula, i.e., $\phi[i/\tilde{i}(\sigma, e)]$.

The i -preassignment predetermines the i -th event to be considered for the evaluation of the constraint. In the case of IF-THEN and event-time rules, this entails that only j , with $1 \leq j < i \leq |\sigma|$, is sought for in order to have C satisfied by $\sigma(i)$ and $\sigma(j)$. For example, consider case σ_1 as illustrated in Fig. 2(d) and constraint $C_2 = \text{IF } \sigma(i).\text{Act} = B \wedge \sigma(j).\text{Act} = A \text{ THEN } \sigma(i).\text{Res} = \sigma(j).\text{Res}$ (Fig. 2(c)). With a slight abuse of notation, we extend the notion of i -preassignment to clauses within the formulation of a constraint (e.g., $\phi_{\text{IF}}[i/\tilde{i}(\sigma, e)]$). For example, although the IF clause of C_2 would be satisfied in σ_1 by setting $i = 2$ and $j = 1$ (i.e., considering e_3 and e_1), but it would not be satisfied if i -preassigned with e_6 as $e_6.\text{Act} = C$.

Equipped with this notion, we define how to check if an i -preassigned constraint is satisfied in a case. Notice we consider IF-THEN and event-time constraints as satisfied if and only if both the IF and THEN clauses are satisfied, unlike the common interpretation of an “if-then” implication may suggest. This design choice is motivated by the goal to avoid that the *ex falso quod libet* statement applies (i.e., that in case the antecedent evaluates to false, the constraint holds true regardless of the consequent), as it reportedly leads to an overestimation of the support of a given rule, as explained in detail in the context of declarative process mining [53, 54, 55]. Considering the example above, C_2 i -preassigned with e_6 is not satisfied because its IF clause is not satisfied either.

The score function, formalized in the following, counts the number of satisfied constraints that are i -preassigned with an event.

Definition 12 (Score function). Let E be the universe of events as per Def. 1, $S(L)$ represent the cases of log L as per Def. 5 and \mathfrak{C} be the universe of constraints as per Def. 10. Considering the i -preassignment with $e \phi[i/\tilde{i}(\sigma, e)]$ as per Def. 11, let $\text{eSat} : \mathfrak{C} \times E \times S(L) \rightarrow \{0, 1\}$ be a function that indicates whether an i -preassigned constraint is satisfied given $e \in E$ in case $\sigma \in S(L)$ as follows:

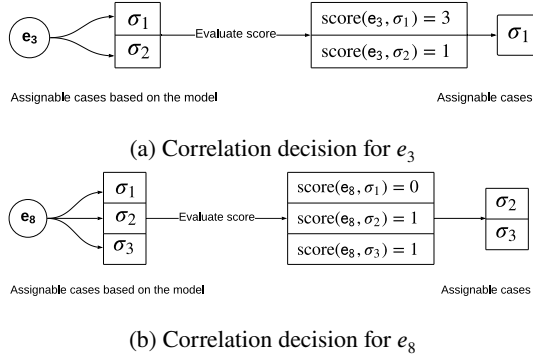
$$\text{eSat}(C, e, \sigma) = \begin{cases} 1 & \text{if } C \text{ is } \phi^{\text{EQ}} \text{ and } \phi^{\text{EQ}}[i/\tilde{i}(\sigma, e)] \equiv \text{true} \\ 1 & \text{if } C \text{ is } \phi^{\text{IF-THEN}} \text{ and } \phi_{\text{IF}}[i/\tilde{i}(\sigma, e)] \equiv \phi_{\text{THEN}}[i/\tilde{i}(\sigma, e)] \equiv \text{true} \\ 1 & \text{if } C \text{ is } \phi^{\text{ET}} \text{ and } \phi_{\text{IF}}[i/\tilde{i}(\sigma, e)] \equiv \phi_{\text{THEN}}[i/\tilde{i}(\sigma, e)] \equiv \text{true} \\ 0 & \text{otherwise} \end{cases} \quad (1)$$

Let $\mathcal{C} \subseteq \mathfrak{C}$ be a set of constraints. The score function $\text{score} : 2^{\mathfrak{C}} \times E \times S(L) \rightarrow \mathbb{N}$ counts the satisfied constraints within a set $\mathcal{C} \in \mathfrak{C}$ given event $e \in E$ in case $\sigma \in S(L)$ as follows:

$$\text{score}(\mathcal{C}, e, \sigma) = \sum_{C \in \mathcal{C}} \text{eSat}(C, e, \sigma) \quad (2)$$

We use the data constraints to guide the selection of the most proper case for an event e . We identify the best candidate as the first one ranked by the score. If multiple cases share the highest score, we randomly select one of them. Randomness is legitimate in this context as it helps to escape the local optimal solution over subsequent SA-iterations.

For example, considering the process model in Fig. 2(b) and the data constraints in Fig. 2(c), EC-SA-Data generates the individual represented as log graph LG_1 in Fig. 11(a)


 Figure 8: Event correlation decision for events e_3 and e_8

from the uncorrelated log in Fig. 2(a). Scanning the events, we initially correlate e_1 with σ_1 , and upon e_2 , σ_2 starts. At this stage, there are two open cases before e_3 : both σ_1 and σ_2 expect the occurrence of activity B or activity C because both are enabled as per the process model. Therefore, e_3 can belong to each of those cases, as shown in Fig. 8(a). Considering the data constraints in Fig. 2(c), let us focus on C_1 , C_2 and C_4 in particular. They pertain to e_3 as C_1 verifies the equality of the Type attribute of an event with that of the preceding one (for all events), while C_2 and C_4 check that $\sigma(i).Act = B$ in their IF clause (and $e_3.Act = B$). As for C_1 (i.e., $\sigma(i).Type = \sigma(i-1).Type$) and its i -preassignment with e_3 , we observe that it is satisfied in σ_1 as $e_1.Type$ is Home like $e_3.Type$. Instead, the i -preassigned constraint is violated in σ_2 as $e_2.Type$ is Car, unlike $e_3.Type$. Constraint C_2 (i.e., IF $\sigma(i).Act = B \wedge \sigma(j).Act = A$ THEN $\sigma(i).Res = \sigma(j).Res$), once i -preassigned with e_3 , is satisfied in σ_1 as the IF and THEN clauses evaluate to true and $e_1.Res = e_3.Res = Kate$. Again, it is violated in σ_2 because $e_1.Res \neq e_3.Res$. As far as C_4 (IF $\sigma(i).Act = B$ THEN $30 \leq (\sigma(i).Ts - \sigma(i-1).Ts) \leq 120$) and its i -preassignment with e_3 are concerned, the execution duration of activity B in σ_1 is $e_3.Ts - e_1.Ts = 60$ min. Also, in σ_2 we have $e_3.Ts - e_2.Ts = 30$ min. Therefore, the i -preassigned C_4 is satisfied both in σ_1 and σ_2 .

Then, we compute the score as in Eq. (2) to rank the possible cases. As shown in Fig. 8(a), σ_1 gets the highest score as it supports the three constraints unlike σ_2 . Therefore, e_3 is assigned with σ_1 as graphically depicted in Fig. 6. Following this procedure, we observe that also e_6 is assigned with the same case later on.

To guarantee that all events are associated to a case, we consider all cases as assignable to the non-enabled events.

Again, we rank the cases based on their score to decide the assignment. For instance, e_8 is a non-enabled event, because neither of the three running cases σ_1 , σ_2 and σ_3 expects the execution of activity D. Therefore, e_8 is assigned based on the data constraints. Considering the constraints in Fig. 2(c), we focus on the i -preassignment of C_1 and C_5 with e_8 and evaluate them over σ_1 , σ_2 and σ_3 . As shown in Fig. 8(b), σ_1 gets the lowest score, as both $C_1[i/\bar{i}(\sigma_1, e_8)]$ and $C_5[i/\bar{i}(\sigma_1, e_8)]$ are violated. In σ_2 and σ_3 , instead, the i -preassigned C_1 is satisfied. Still, both $C_5[i/\bar{i}(\sigma_2, e_8)]$ and

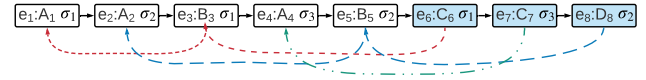
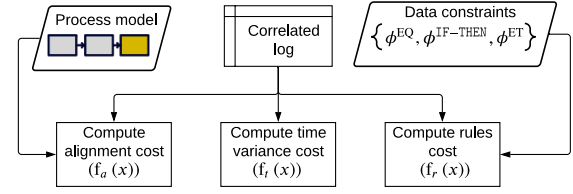

 Figure 9: New individual x' based on LG_1


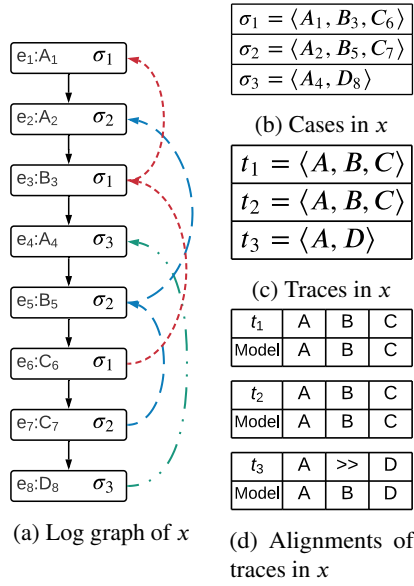
Figure 10: Energy cost functions

$C_5[i/\bar{i}(\sigma_3, e_8)]$ are violated as in both cases activity D exceeds the allowed maximum duration. Therefore, e_8 is randomly assigned with σ_3 as one of the cases with the highest score, as illustrated in Fig. 6.

4.4. Neighbor creation

As illustrated in Fig. 4, simulated annealing explores the search space by creating a new neighbor individual (x') based on the current individual (x) at the beginning of each iteration. SA typically generates x' by randomly updating some parts of x . In our case, we generate the neighbor x' by modifying the event-case assignments from a given event in UL on. We name that event the *changing point*.

The changing point is selected based on the current step S_{curr} , the increase of which corresponds to a decrease in the temperature, τ_{curr} . When $S_{curr} = 1$, τ_{curr} is at its maximum (τ_{init}), and the changing point is randomly selected among the first few events. Therefore, the events that follow the changing point are the large majority, and their reassignment allows for a wide exploration of the space. Instead, when $S_{curr} = S_{max} - 1$, the temperature is at its minimum during the execution of the algorithm and the changing point is randomly picked among the last few events. As a consequence, fewer events are going to be reassigned as we aim to converge towards a solution. The increment of S_{curr} (hence, the decrement of τ_{curr}) reduces the number of events to be re-evaluated at each iteration: this is in line with the cooling down mechanism of the annealing process. More in detail, EC-SA-Data selects a segment of events in the uncorrelated log starting at index $\left\lfloor \frac{S_{curr}}{S_{max}} \cdot |E| \right\rfloor$ and ending at index $\left\lceil \frac{S_{curr}+1}{S_{max}} \cdot |E| \right\rceil$, then the changing point is drawn randomly in that range. For instance, let $S_{max} = 2$ and $\tau_{init} = 100$. After generating the initial individual in Fig. 6, we have $S_{curr} = 1$. Notice that, at this iteration, $S_{curr} = S_{max} - 1$, so the changing point is randomly selected from the second half of the events in UL, i.e., from e_4 to e_8 . In this example, the changing point corresponds to e_6 . As shown in Fig. 9, a new individual x' is thus created and we explore the possible assignments for events e_6 , e_7 and e_8 , while the other events remain assigned as in x .


 Figure 11: Computation of the alignments for individual x

4.5. Energy cost function

Simulated annealing uses the energy function to model the optimization problem. We model the event correlation problem as a multi-level optimization problem, specifically with three levels of objectives: (i) minimizing the misalignment between the generated event log and the input process model, (ii) minimizing the constraints violations over the cases, and (iii) minimizing the activity execution time variance across the cases. Therefore, we define an energy function for each of these objectives, as shown in Fig. 10. The first energy function ($f_a(x)$) computes the cost of aligning x and the model. The second energy function ($f_r(x)$) computes the data rules violations cost within x . The third energy function ($f_t(x)$) computes the activity execution time variance within x . They are used to compute the energy cost function between x and x' , $\delta f_c(x, x')$, as shown in Fig. 4. In the following, we elaborate on the individual energy functions and the computation of $\delta f_c(x, x')$.

4.5.1. Alignment cost

To measure the model-log misalignment we use the well-established *alignment cost* function proposed by Adriansyah et al. [56]. The technique penalizes every asynchronous move between the log and the model, that is to say, it associates a cost to every event that occurs in the trace although the model would not allow for it, or every missing event that the model would require to continue the run though the trace does not contain it. Figure 11 shows an example of computing the log alignment cost $f_a(x)$ over individual x (depicted in Fig. 11(a)). The first step is to extract the cases from individual x as shown in Fig. 11(b). Then, EC-SA-Data deduces the traces from the cases by projecting them over the cases' activities as shown in Fig. 11(c). For each trace σ in the log, it computes the alignment cost of the trace ($\Delta_{\text{align}}(\text{Act}(\sigma))$) with respect to the

process model.¹ The log alignment cost is finally computed as the summation of the trace alignment costs, as shown in Eq. (3).

$$f_a(x) = \sum_{\sigma \in S(x)} \Delta_{\text{align}}(\text{Act}(\sigma)) \quad (3)$$

For example, Fig. 11(d) shows that the execution of activity D in t_3 is considered as an asynchronous move as it is not enabled by the model (indicated with a guillemet in the figure, \gg). Thus, $\Delta_{\text{align}}(t_3) = 1$. On the other hand, $\Delta_{\text{align}}(t_1) = \Delta_{\text{align}}(t_2) = 0$ as t_1 and t_2 are in complete alignment with the model. The log alignment cost of x (depicted in Fig. 11(a)) is $f_a(x) = \Delta_{\text{align}}(t_1) + \Delta_{\text{align}}(t_1) + \Delta_{\text{align}}(t_3) = 0 + 0 + 1 = 1$.

4.5.2. Rule cost

EC-SA-Data measures the cost of data constraint violations in a log (henceforth, rule cost, $f_r(x)$) by evaluating the constraints over the log's cases. A constraint is triggered by a case if (i) it is an equality constraint (hence, always triggered), or (ii) it is an IF-THEN constraint or an event-time constraint and the IF part is satisfied by at least an event in the case.

The second step evaluates the triggered constraints over the case. For every violated constraint, a penalty is added for the case. A constraint C is violated by a case σ if any of the following holds: (i) C is an equality constraint and at least an event violates it; (ii) C is an IF-THEN or an event-time constraint and the IF clause is satisfied while the THEN clause is violated by at least a pair of events in the case. We formalize these notions as follows.

Definition 13 (Rule cost). Let $S(L)$ represent the cases of log L as per Def. 5 and $\mathfrak{C} \supseteq \mathcal{C} \ni C$ be a universe of constraints as per Def. 10. Let $\text{trigger} : \mathfrak{C} \times S(L) \rightarrow \{\text{true}, \text{false}\}$ be a function that determines whether a constraint $C \in \mathfrak{C}$ is triggered by a case, as follows:

$$\text{trigger}(C, \sigma) = \begin{cases} \text{true} & \text{if } C \text{ is } \phi^{\text{EQ}} \\ \text{true} & \text{if } C \text{ is } \phi^{\text{IF-THEN}} \text{ and } \varphi_{\text{IF}} \equiv \text{true} \\ \text{true} & \text{if } C \text{ is } \phi^{\text{ET}} \text{ and } \varphi_{\text{IF}} \equiv \text{true} \\ \text{false} & \text{otherwise} \end{cases} \quad (4)$$

Let function $\text{trig}_C : 2^{\mathfrak{C}} \times S(L) \rightarrow 2^{\mathfrak{C}}$ return a subset of constraints in $\mathcal{C} \in 2^{\mathfrak{C}}$ that are triggered by $\sigma \in S(L)$ as follows:

$$\text{trig}_C(\mathcal{C}, \sigma) = \{C \in \mathcal{C} : \text{trigger}(C, \sigma) = \text{true}\} \quad (5)$$

Let $\text{eVio} : \mathfrak{C} \times E \times S(L) \rightarrow \{\text{true}, \text{false}\}$ be a function that determines whether an i -preassigned constraint C (see

¹We remark that Δ_{align} requires a trace *and* a process model to be computed. As the process model is given as background knowledge in this context, we omit its explicit mention as an input parameter for the sake of readability.

Table 1: Rule cost computation for individual x , considering the data constraints in $\mathcal{C} = \{C_1, \dots, C_5\}$ as per Fig. 2(c)

Cases	$\text{trig}_{\mathcal{C}}(\mathcal{C}, \sigma)$	$\text{vio}(\mathcal{C}, \sigma)$
$\sigma_1 = \langle A_1, B_3, C_6 \rangle$	C_1 as it is ϕ^{EQ}	$A_1.\text{Type} = B_3.\text{Type} \wedge B_3.\text{Type} = C_6.\text{Type}$ $\implies \text{vio}(C_1, \sigma_1) = 0$
	C_2 as ϕ^{IF} is satisfied by B_3 and A_1	Evaluate $\phi^{\text{THEN}} : B_3.\text{Res} = A_1.\text{Res}$ $\implies \text{vio}(C_2, \sigma_1) = 0$
	C_3 as ϕ^{IF} is satisfied by C_6 and B_3	Evaluate $\phi^{\text{THEN}} : C_6.\text{Res} \neq B_3.\text{Res}$ $\implies \text{vio}(C_3, \sigma_1) = 0$
	C_4 as ϕ^{IF} is satisfied by B_3	Evaluate $\phi^{\text{THEN}} : 30 \leq B_3.\text{Ts} - A_1.\text{Ts} \leq 120$ $\implies \text{vio}(C_4, \sigma_1) = 0$
$\sigma_2 = \langle A_2, B_3, C_7 \rangle$	C_1 as it is ϕ^{EQ}	$A_2.\text{Type} = B_3.\text{Type}$ and $B_3.\text{Type} = C_7.\text{Type}$ $\implies \text{vio}(C_1, \sigma_2) = 0$
	C_2 as ϕ^{IF} is satisfied by B_3 and A_2	Evaluate $\phi^{\text{THEN}} : B_3.\text{Res} = A_2.\text{Res}$ $\implies \text{vio}(C_2, \sigma_2) = 0$
	C_3 as ϕ^{IF} is satisfied by C_7 and B_3	Evaluate $\phi^{\text{THEN}} : C_7.\text{Res} \neq B_3.\text{Res}$ $\implies \text{vio}(C_3, \sigma_2) = 0$
	C_4 as ϕ^{IF} is satisfied by B_3	Evaluate $\phi^{\text{THEN}} : 30 \leq B_3.\text{Ts} - A_2.\text{Ts} \leq 120$ $\implies \text{vio}(C_4, \sigma_2) = 0$
$\sigma_3 = \langle A_4, D_8 \rangle$	C_1 as it is ϕ^{EQ}	$A_4.\text{Type} = D_8.\text{Type}$ $\implies \text{vio}(C_1, \sigma_3) = 0$
	C_5 as ϕ^{IF} is satisfied by D_8	Evaluate $\phi^{\text{THEN}} : 120 \leq D_8.\text{Ts} - A_4.\text{Ts} \leq 240$ $\implies \text{vio}(C_5, \sigma_3) = 1$

Def. 11) is violated given $e \in E$ in case $\sigma \in S(L)$ as follows:

$$\text{eVio}(\mathcal{C}, e, \sigma) = \begin{cases} \text{true} & \text{if } C \text{ is } \phi^{\text{EQ}} \text{ and } \phi^{\text{EQ}}[i/\tilde{\gamma}(\sigma, e)] \equiv \text{false} \\ & \text{if } C \text{ is } \phi^{\text{IF-THEN}} \text{ and } \phi_{\text{IF}}[i/\tilde{\gamma}(\sigma, e)] \equiv \text{true} \\ & \text{and } \phi_{\text{THEN}}[i/\tilde{\gamma}(\sigma, e)] \equiv \text{false} \\ \text{true} & \text{if } C \text{ is } \phi^{\text{ET}} \text{ and } \phi_{\text{IF}}[i/\tilde{\gamma}(\sigma, e)] \equiv \text{true} \\ & \text{and } \phi_{\text{THEN}}[i/\tilde{\gamma}(\sigma, e)] \equiv \text{false} \\ \text{false} & \text{otherwise} \end{cases} \quad (6)$$

Let $\text{vio} : \mathcal{C} \times S(L) \rightarrow \{0, 1\}$ be an indicator function that returns 1 if there exists at least an event $e \in \sigma \in S(L)$ such that $\text{eVio}(\mathcal{C}, e, \sigma) = \text{true}$ or 0 otherwise. The *rule cost*, $f_r(x)$, is a function computed as the sum of the ratios of triggered constraints that are violated by every case, divided by the number of cases in individual (log) x :²

$$f_r(x) = \frac{1}{|I|} \sum_{\sigma \in S(x)} \frac{\sum_{C \in \text{trig}_{\mathcal{C}}(\mathcal{C}, \sigma)} \text{vio}(\mathcal{C}, \sigma)}{|\text{trig}_{\mathcal{C}}(\mathcal{C}, \sigma)|} \quad (7)$$

To compute the rule cost over individual x , we first identify the constraints in set \mathcal{C} that are triggered by σ with $\text{trig}_{\mathcal{C}}(\mathcal{C}, \sigma_i)$. Then, we verify each constraint C in $\text{trig}_{\mathcal{C}}$ over σ . Finally, we take the average of violations over the log: $\text{RC} = 0$ if no violation occurs.

Table 1 shows how we evaluate constraints in our running example, having $\mathcal{C} = \{C_1, \dots, C_5\}$ (shown in Fig. 2(c)) over x (shown in Fig. 11(a)). Cases σ_1 and σ_2 trigger C_1 , C_2 , C_3 and C_4 , while σ_3 triggers C_1 and C_5 . We observe that σ_1 and σ_2 satisfy their triggered constraints, while σ_3 violates C_5 , as $D_8.\text{Ts} - A_4.\text{Ts} = 180 \text{ min}$ thus exceeding the maximum admissible duration of activity D (150 min). Therefore, $f_r(x) = \frac{1}{3} \left(\frac{0}{4} + \frac{0}{4} + \frac{1}{2} \right) = 0.167$.

²We remark that $\delta f_r(x)$ requires an event log and a set of data constraints to be computed. As the set of constraints is given as background knowledge in this context, we keep it as an implicit parameter for the sake of readability.

4.5.3. Execution time variation cost

The third objective is to minimize the activities' execution time variance over the correlated events. EC-SA-Data employs the Mean Square Error (MSE) [57] to measure the execution variance over the log for the $f_t(x)$ energy function. MSE measures the deviation between expected values and actual values. We assume that the activities tend to be carried out similarly across cases. Therefore, we use the activities' average execution time over the log to represent the expected duration. We use the events' elapsed time to represent the actual one. We formalize these concepts as follows.

Definition 14 (Time variance). Let $S(L)$ represent the cases of log L as per Def. 5. The event-time function $\text{ET} : E^* \times E \rightarrow \mathbb{R}^+$ computes the elapsed time of an event $\sigma(i) \in E$ based on the preceding event in the same case $\sigma(i-1) \in E$ as follows:

$$\text{ET}(\sigma, \sigma(i)) = \begin{cases} \sigma(i).\text{Ts} - \sigma(i-1).\text{Ts} & \text{if } 1 < i \leq n \\ 0 & \text{otherwise} \end{cases} \quad (8)$$

Let $\text{NSE}(L, a)$ be the set of non-starting events in the cases of $S(L)$ that report the execution of $a \in \text{Dom}_{\text{Act}}$: $\text{NSE}(L, a) = \bigcup_{\sigma \in S(L)} \{\sigma(i) \in \sigma : 1 < i \leq |\sigma| \text{ and } \sigma(i).\text{Act} = a\}$.

Let $T_{\text{avg}}(L, a)$ be the average activity execution duration of activity $a \in \text{Dom}_{\text{Act}}$ as per log L :

$$T_{\text{avg}}(L, a) = \frac{\sum_{e \in \text{NSE}(L, a)} \text{ET}(\sigma(e), e)}{|\text{NSE}(L, a)|} \quad (9)$$

Given an individual (log) x time variance is the mean square error of the activity execution duration over the events in x :

$$f_t(x) = \frac{\sum_{\sigma \in S(x)} \left(\sum_{i=2}^{|\sigma|} (T_{\text{avg}}(\sigma(i).\text{Act}) - \text{ET}(\sigma, \sigma(i)))^2 \right)}{|E| - |I|} \quad (10)$$

EC-SA-Data computes the time variance by measuring the mean square error (MSE) using $T_{\text{avg}}(a)$ as its expected values, and the events' elapsed time as its actual values. MSE is computed for all the events *except* the start events of the cases in log L . Therefore, the denominator of $f_t(x)$ is $(|E| - |I|)$ in order not to count the $|I|$ start events. For example, let us see how EC-SA-Data computes $f_t(x)$ for the x individual depicted in Fig. 11(a). The first step is computing the average execution time of activities in x to represent the expected values. The average execution times (in minutes) are $T_{\text{avg}}(B) = 75$, $T_{\text{avg}}(C) = 120$, and $T_{\text{avg}}(D) = 180$. Then, we compute the elapsed time of the events to represent the actual time and, thereupon, the time variance given the number of non-start events $(|E| - |I| = 8 - 3 = 5)$. As a result, $f_t(x) = 90 \text{ min}$.

4.5.4. Cost function computation

Simulated annealing evaluates the energy cost of changing from individual x to a new individual x' in order to

decide which one to keep in the next iteration. EC-SA-Data computes the *energy cost function*, $\delta f_c(x, x')$, based on the objective functions f_a (alignment cost), f_r (rule cost) and f_t (time variance), as shown in Eq. (11). We use these three energy functions to apply the multiple-level optimization as follows: (i) δf_c is computed based on f_a if the alignment cost of x is lower than that of x' ; else, we compute (ii) δf_c is based on f_r if x' violates more data rules than x and x' is better aligned with the model; otherwise, (iii) δf_c is computed based on f_t .

$$\delta f_c(x, x') = \begin{cases} f_a(x') - f_a(x) & \text{if } f_a(x') > f_a(x) \\ f_r(x') - f_r(x) & \text{if } f_a(x') \leq f_a(x) \text{ and } f_r(x') > f_r(x) \\ f_t(x') - f_t(x) & \text{otherwise.} \end{cases} \quad (11)$$

The energy cost function computes the cost of choosing the new neighbor individual x' over x . Therefore, $\delta f_c(x, x')$ is computed using an energy function where x' performs worse than x as per Eq. (11). For example, Fig. 12(a) depicts the initial individual, x , and the values of the energy functions applied to x . Figure 12(b) shows the new neighbor individual, x' . The energy cost function ($\delta f_c(x, x')$) is computed based on time variance f_t , as the new neighbor x' has a better alignment cost than that of x ($f_a(x') \leq f_a(x)$) and both the individuals have the same rule cost ($f_r(x') = f_r(x)$). Thus, $\delta f_c(x, x') = f_t(x') - f_t(x) = 15.3 - 7.3 = 8$.

4.6. Selection of the next individual

Algorithm 1: Selection of the solution for the next iteration

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

```

1 if  $f_a(x') < f_a(x)$  then return  $x'$ ;
2 else if  $f_a(x') = f_a(x)$  then
3     if  $f_r(x') < f_r(x)$  then return  $x'$ ;
4     else if  $f_r(x') = f_r(x)$  then
5         if  $f_t(x') < f_t(x)$  or  $\text{prob}(x') \geq \text{random}(0, 1)$ 
6             then return  $x'$ ;
7     else if
8          $f_r(x') > f_r(x)$  and  $\text{prob}(x') \geq \text{random}(0, 1)$ 
9         then
10             return  $x'$ 
11 else if  $f_a(x') > f_a(x)$  and  $\text{prob}(x') \geq \text{random}(0, 1)$ 
12     then return  $x'$ ;
13 return  $x$ 
    
```

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 f_a (first-level), f_r (second-level) and f_t (third-level), together with the acceptance probability, $\text{prob}(x')$. The latter is computed using $\delta f_c(x, x')$ and the current temperature (τ_{curr}) as shown in Eq. (12):

$$\text{prob}(x') = \exp \frac{-\delta f_c(x, x')}{\tau_{\text{curr}}} \quad (12)$$

EC-SA-Data compares the value of $\text{prob}(x')$ with a random value in a $[0, 1]$ interval to accept or reject the new neighbor, depending on whether $\text{prob}(x')$ is higher or lower than the random value, respectively. In this way, we simulate the annealing process, enforced by the fact that the decrease of temperature τ_{curr} also reduces the randomness of the choice. Furthermore, notice that the memory-less stochastic perturbation makes it possible to skip the local optimal.

If the new neighbor (x') has a lower alignment cost, then it is selected. If the new neighbor (x') and the current individual (x) have the same alignment cost, then we check the rule cost energy function. If the new neighbor (x') has a lower rule cost, then it is selected. If the new neighbor (x') and current individual (x) have the same rule cost, then we check the time variance energy function. The acceptance probability $\text{prob}(x')$ is computed using $\delta f_c(x, x')$ based on $f_t(x)$ and $f_t(x')$. Then, x' is selected either if x' has a lower time variance or if it is randomly selected based on $\text{prob}(x')$. On the other hand, if the new neighbor has a higher rule cost than the current individual, then we calculate $\delta f_c(x, x')$ based on $f_r(x)$ and $f_r(x')$. The same holds if the new neighbor has a higher alignment cost than the current individual. In that case, we calculate $\delta f_c(x, x')$ based on $f_a(x)$ and $f_a(x')$. We take the final decision based on a random selection weighed by $\text{prob}(x')$ which, in turn, is calculated on the basis of the current temperature, τ_{curr} , and $\delta f_c(x, x')$. This process is repeated for each individual within the population.

For example, Fig. 12 shows the results through the EC-SA-Data iterations. We assume that $S_{\text{max}} = 2$ and $\tau_{\text{init}} = 100$. Figure 12(a) depicts the initial individual, x , and its energy cost functions. Figure 12(b) shows the new individual, x' , generated on the basis of x . $\delta f_c(x, x')$ is computed considering f_t : $\delta f_c(x, x') = 8$. According to Algorithm 1, x' is selected and replaces x in the population as $f_a(x') \leq f_a(x)$.

4.7. Global solution update, cooling down and new iteration

As a final step, EC-SA-Data returns the global optimal solution x_G at S_{max} , namely the solution that has the best f_a , f_r and f_t over all iterations, as shown in Fig. 2. 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 [58] as per Eq. (13). The number of iterations that the logarithmic schedule goes through to cool down helps to skip the local optimum and explore a wider correlation search space especially in the early phases of the run:

$$\tau_{\text{curr}} = \frac{\tau_{\text{init}}}{\ln(1 + S_{\text{curr}})} \quad (13)$$

Following the EC-SA-Data steps in Fig. 4, the algorithm proceeds until $S_{\text{curr}} = S_{\text{max}}$ as shown in Fig. 12. At each iteration, it reassigns the events from different changing points in the log to explore the search space. We recall that accepting a worse solution than the current one in some iterations helps to skip the optimal local solution and reach an approximate optimal global solution.

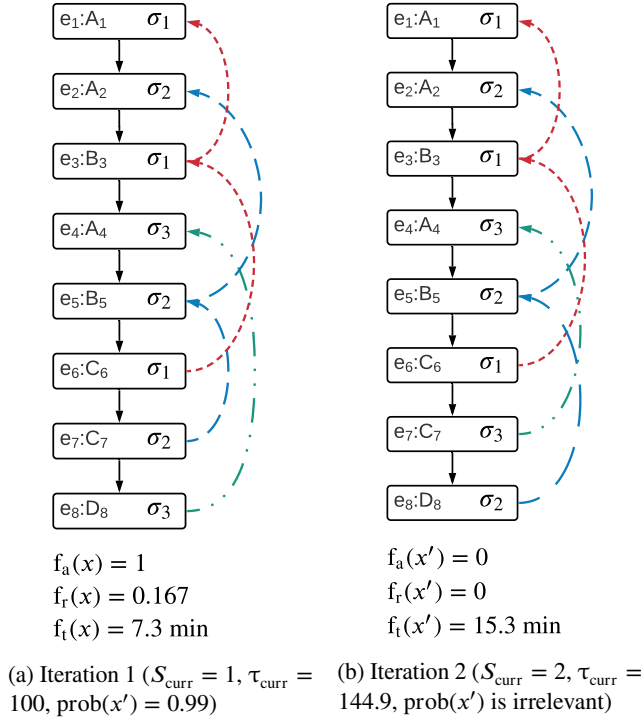


Figure 12: EC-SA-Data iterations, with $S_{\text{max}} = 2$ and $\tau_{\text{init}} = 100$

In the following section, we discuss the measures we introduce to evaluate the accuracy of our technique based on a pair of logs. We will see next in Section 6 that the two logs correspond to a golden standard and the one originated by EC-SA-Data.

5. Quality measures

To assess the quality of our technique, we defined measures that quantify the output accuracy. They are based on criteria that compare pairs of logs. In particular, we can group our measures in two categories. The first category is the *log-to-log similarity*, which takes into account trace-based and case-based distances. The second category is *log-to-log time deviation*, determining the temporal distance of events' elapsed times and cases' cycle times. The definition of a complete set of measures to compare logs goes beyond the scope of this paper. The measures we propose here are inspired by related work in the literature [59, 56, 60, 61, 27, 18] and provide a good trade-off between run-time computability and different level of details used in the comparison, experimental results evidenced. A refinement and enrichment of this set can be part of future investigations.

We use the two event logs in Fig. 13 to illustrate the different quality measures. We remark that our focus is on logs L and L' that stem from the same uncorrelated log and case id's, but possibly differ for the way in which events are correlated. Therefore, the cardinality of case sets $S(L)$ and $S(L')$ yet cases in $S(L)$ and $S(L')$ may differ.

Case Id	Event Id	Activity (Act)	Timestamp (Ts)
1	1	A	01/06/2020 09:00
1	4	B	01/06/2020 10:30
1	5	C	01/06/2020 11:00
2	2	A	01/06/2020 09:30
2	6	C	01/06/2020 12:00
2	7	B	01/06/2020 13:00
3	3	A	01/06/2020 10:00
3	8	B	01/06/2020 13:10
3	9	C	01/06/2020 13:50

(a) Log L

(b) Log L'

Figure 13: Event logs L and L' for quality measures example

5.1. Log-to-log similarity

The log-to-log similarity category focuses on measuring the structure similarity between two logs from trace- and case-structure perspectives. This category is comprised of six measures. The values of those measures range from 0 to 1, where 1 indicates the highest similarity. In the following, we describe the measures following an order given by a decreasing level of abstraction and aggregation with which the similarity between a pair of logs is established. The higher the level of detail, the more fine-granular differences are considered by the measure.

Inspired by the fitness measure proposed in [56], we define our first similarity measure, *trace-to-trace similarity*. It aims at assessing the extent to which two logs capture the same underlying control-flow through the string-edit distance of their traces. We formally define it as follows.

Definition 15 (Trace-to-trace similarity). Let $L = (UL, I, \ell)$ and $L' = (UL, I, \ell')$ be two event logs. Let $\Delta_{\text{del}}^{\text{ins}}$ be the string-edit distance based on insertions and deletions [59]. We denote with $T = \{t_1, t_2, \dots, t_{|T|}\}$ and $T' = \{t'_1, t'_2, \dots, t'_{|T'|}\}$ the set of *distinct* traces that are derived from event logs L and L' , respectively, i.e., $T = \bigcup_{\sigma \in S(L)} \text{Act}(\sigma)$ and $T' = \bigcup_{\sigma' \in S(L')} \text{Act}(\sigma')$. We indicate as the *trace-closest trace* to $t \in T$ a trace in $t'_\star \in T'$ derived as follows:

$$t'_\star = \arg \min_{t' \in T'} \left\{ \Delta_{\text{del}}^{\text{ins}}(t, t') \right\} \quad (14)$$

The *trace-to-trace similarity* $\text{L2L}_{\text{trace}}$ is computed as follows:

$$\text{L2L}_{\text{trace}} = 1 - \frac{\sum_{t \in T} \Delta_{\text{del}}^{\text{ins}}(t, t'_\star)}{\sum_{t \in T} (|t| + |t'_\star|)} \quad (15)$$

For example, Table 2(a) shows two distinct traces in event log L (depicted in Fig. 13(a)) and Table 2(b) shows two distinct traces in event log L' (Fig. 13(b)). The pairs of trace-closest traces (t, t'_\star) are illustrated in Table 2(c). We select the pairs that minimize the total distance Δ_{total} between traces (see the marked cells in Table 2(c)). For instance, for t_1 we select the trace-closest pair (t_1, t'_2) instead of (t_1, t'_1) because the distance of the former $\Delta_{\text{del}}^{\text{ins}}(t_1, t'_2) = 0$ is lower than

Table 2: Computation of $L2L_{\text{trace}}$ for logs L and L' in Fig. 13

(a) Distinct traces in L	(b) Distinct traces in L'
$t_1 = \langle A, B, C \rangle$	$t'_1 = \langle A, C, B \rangle$
$t_2 = \langle A, C, B \rangle$	$t'_2 = \langle A, B, C \rangle$
(c) Matching pair (t_*, t'_*) for each trace in L and L'	
$\Delta_{\text{del}}^{\text{ins}}(t_1, t'_1) = 1$	$\Delta_{\text{del}}^{\text{ins}}(t_1, t'_2) = 0$
$\Delta_{\text{del}}^{\text{ins}}(t_2, t'_1) = 0$	$\Delta_{\text{del}}^{\text{ins}}(t_2, t'_2) = 1$

$\Delta_{\text{del}}^{\text{ins}}(t_1, t'_1) = 1$. Finally, we compute $L2L_{\text{trace}}$ as a fraction having the sum of the string-edit distances between pairs of trace-closest traces ($0 + 0 = 0$) as the numerator and the length of the traces in L and their trace-closest traces in L' ($k = |t_1| + |t'_1| + |t_2| + |t'_2| = 12$) as the denominator.

The second measure we introduce is the *trace-to-trace frequency similarity*. It is more fine-granular than trace-to-trace similarity as it also takes into account the frequency with which traces occur. The formal definition follows.

Definition 16 (Trace-to-trace frequency similarity). Let $L = (\text{UL}, I, \ell)$ and $L' = (\text{UL}, I, \ell')$ be two event logs, whose cases are $S(L) = \{\sigma_1, \sigma_2, \dots, \sigma_{|I|}\}$ and $S(L') = \{\sigma'_1, \sigma'_2, \dots, \sigma'_{|I|}\}$ respectively. Let $\Delta_{\text{del}}^{\text{ins}}$ be the string-edit distance based on insertions and deletions [59]. Let $\text{tcc} : S(L) \rightarrow S(L')$ be a bijective function mapping every case in L to exactly one case in L' , TCC the set of all possible such bijective functions definable having $S(L)$ and $S(L')$ as domain and range respectively, and $\text{tcc}_* \in \text{TCC}$ be such that the total string-edit distance between tcc_* -mapped pairs of cases is minimal:

$$\text{tcc}_* = \arg \min_{\text{tcc} \in \text{TCC}} \left\{ \sum_{\sigma \in S(L)} \Delta_{\text{del}}^{\text{ins}}(\text{Act}(\sigma), \text{Act}(\text{tcc}(\sigma))) \right\} \quad (16)$$

Naming the tcc_* -mapped cases as trace-closest case pairs, let $\Delta_{\text{total}}(L, L')$ be the sum of all string-edit distances between trace-closest case pairs:

$$\Delta_{\text{total}} = \sum_{\sigma \in S(L)} \Delta_{\text{del}}^{\text{ins}}(\text{Act}(\sigma), \text{Act}(\text{tcc}_*(\sigma))) \quad (17)$$

The *trace-to-trace frequency similarity*, $L2L_{\text{freq}}$, is the opposite of the average of the total distances between trace-closest case pairs:

$$L2L_{\text{trace}}(L, L') = 1 - \frac{\Delta_{\text{total}}}{2 \times |E|} \quad (18)$$

To compute tcc_* as in Eq. (16) and thus find the trace-closest case pairs, we use the Hungarian Algorithm [62]. For example, Table 3(a) shows the traces that stem from event log L (depicted in Fig. 13(a)) and Table 2(b) shows the traces stemming from event log L' (depicted in Fig. 13(b)). Table 3(c) illustrates the pairs of trace-closest cases that we derive. The selected ones are colored in light blue: $\Delta_{\text{del}}^{\text{ins}}(\text{Act}(\sigma_1), \text{Act}(\sigma'_1)) = 0$, $\Delta_{\text{del}}^{\text{ins}}(\text{Act}(\sigma_2), \text{Act}(\sigma'_2)) = 0$, and $\Delta_{\text{del}}^{\text{ins}}(\text{Act}(\sigma_3), \text{Act}(\sigma'_3)) = 2$. These pairs lead to

 Table 3: Computation of $L2L_{\text{freq}}$ for logs L and L' in Fig. 13

(a) Traces in L

$\text{Act}(\sigma_1) = \langle A, B, C \rangle$
$\text{Act}(\sigma_2) = \langle A, C, B \rangle$
$\text{Act}(\sigma_3) = \langle A, B, C \rangle$

(b) Traces in L'

$\text{Act}(\sigma'_1) = \langle A, B, C \rangle$
$\text{Act}(\sigma'_2) = \langle A, C, B \rangle$
$\text{Act}(\sigma'_3) = \langle A, C, B \rangle$

(c) Matching pair $(\sigma_\star, \sigma'_\star)$ for each trace in L and L'

$\Delta_{\text{del}}^{\text{ins}}(\text{Act}(\sigma_1), \text{Act}(\sigma'_1)) = 0$	$\Delta_{\text{del}}^{\text{ins}}(\text{Act}(\sigma_2), \text{Act}(\sigma'_1)) = 2$	$\Delta_{\text{del}}^{\text{ins}}(\text{Act}(\sigma_3), \text{Act}(\sigma'_1)) = 0$
$\Delta_{\text{del}}^{\text{ins}}(\text{Act}(\sigma_1), \text{Act}(\sigma'_2)) = 2$	$\Delta_{\text{del}}^{\text{ins}}(\text{Act}(\sigma_2), \text{Act}(\sigma'_2)) = 0$	$\Delta_{\text{del}}^{\text{ins}}(\text{Act}(\sigma_3), \text{Act}(\sigma'_2)) = 2$
$\Delta_{\text{del}}^{\text{ins}}(\text{Act}(\sigma_1), \text{Act}(\sigma'_3)) = 0$	$\Delta_{\text{del}}^{\text{ins}}(\text{Act}(\sigma_2), \text{Act}(\sigma'_3)) = 0$	$\Delta_{\text{del}}^{\text{ins}}(\text{Act}(\sigma_3), \text{Act}(\sigma'_3)) = 2$

 Table 4: Intersect between cases with the same start event in L and L'

$\text{intersect}(\sigma_1, \sigma'_1) = 1$	$\text{intersect}(\sigma_2, \sigma'_2) = 1$	$\text{intersect}(\sigma_3, \sigma'_3) = 1$
---	---	---

the minimum $\Delta_{\text{total}} = 2$. Finally, we compute $L2L_{\text{freq}} = 1 - \frac{2}{2 \times 9} = 0.78$.

The following four measures investigate the similarity between logs at the level of events. Notice that we compare pairs of cases for which the first event correspond. The third measure we describe is the *partial case similarity*, which is based upon the number of events shared by cases that have the same first event.

Definition 17 (Partial case similarity). Let $L = (\text{UL}, I, \ell)$ and $L' = (\text{UL}, I, \ell')$ be two event logs, whose case sets are $S(L)$ and $S(L')$ respectively. We indicate with $\text{intersect} : E^* \times E^* \rightarrow \mathbb{N} \cup 0$ a function that takes two cases σ and σ' as input and returns the number of events that σ and σ' have in common.

$$\text{intersect}(\sigma, \sigma') = |\{e \in \sigma : e \in \sigma'\}| \quad (19)$$

The partial case similarity distance $L2L_{\text{first}}$ averages the number of events in common (except the first one) that cases in L and L' have when they share the same first event over the number of events in L (except the first ones of the cases), as follows:

$$L2L_{\text{first}}(L, L') = \frac{\sum_{\substack{\sigma \in S(L), \\ \sigma' \in S(L') : \\ \sigma(1) = \sigma'(1)}} \text{intersect}(\sigma[2, |\sigma|], \sigma'[2, |\sigma'|])}{|E| - |I|} \quad (20)$$

For instance, given the event logs L and L' in Figs. 13(a) and 13(b), we compute the elements of the sum in the numerator of $L2L_{\text{first}}(L, L')$ as depicted in Table 4. In the example, σ_1 and σ'_1 have the same start event A_1 , thus, we check the occurrence of events in $\sigma_1[2, 3]$ in $\sigma'_1[2, 3]$ and find that only B_4 occur in both cases, so $\text{intersect}(\sigma_1[2, 3], \sigma'_1[2, 3]) = 1$. Finally, $L2L_{\text{first}}$ is computed by averaging the non-start events in common over the total number of the non-start events: $L2L_{\text{first}} = \frac{1+1+1}{9-3} = 0.5$.

Table 5: Computation of $L2L_{2\text{gram}}$ for logs L and L' in Fig. 13

$\text{occurs2}(\langle A_1, B_4 \rangle, L') = 1$	$\text{occurs2}(\langle B_4, C_5 \rangle, L') = 0$
$\text{occurs2}(\langle A_2, C_6 \rangle, L') = 0$	$\text{occurs2}(\langle C_6, B_7 \rangle, L') = 0$
$\text{occurs2}(\langle A_3, B_8 \rangle, L') = 0$	$\text{occurs2}(\langle B_8, C_9 \rangle, L') = 0$

The fourth measure we describe is the *bigram similarity*. Inspired by [15], it is based on the number of sequences of two events (henceforth, bigrams, i.e., n-grams of length 2) that occur in both logs. We formally define it as follows.

Definition 18 (Bigram similarity). Let $L = (UL, I, \ell)$ and $L' = (UL, I, \ell')$ be two event logs, whose case sets are $S(L)$ and $S(L')$ respectively. We denote as $\text{occurs2}(\langle e, e' \rangle, L)$ the indicator function that returns 1 if there exists a case $\sigma \in S(L)$ such that $\langle e, e' \rangle$ is a segment of it:

$$\text{occurs2}(\langle e, e' \rangle, L) = \begin{cases} 1 & \text{if there exists } \sigma \in S(L) \text{ s.t. } \langle e, e' \rangle \subseteq \sigma \\ 0 & \text{otherwise} \end{cases} \quad (21)$$

The *bigram similarity* $L2L_{2\text{gram}}$ is computed dividing by the cardinality of $S(L)$ the average of bigrams in the cases of L that also occur in L' as follows:

$$L2L_{2\text{gram}}(L, L') = \frac{1}{|I|} \sum_{\sigma \in S(L)} \frac{1}{|\sigma|-1} \left(\sum_{i=1}^{|\sigma|-1} \text{occurs2}(\langle \sigma(i), \sigma(i+1) \rangle, L') \right) \quad (22)$$

For example, for every pair σ in L (depicted in Fig. 13(a)), we check if $\langle e, e' \rangle$ occurs in L' (depicted in Fig. 13(b)) as shown in Table 5. Notice that L and L' have only one bigram in common, that is $\langle A_1, B_4 \rangle$. Therefore, $L2L_{2\text{gram}} = \frac{1}{3} \left(\frac{1+0}{3-1} + \frac{0+0}{3-1} + \frac{0+0}{3-1} \right) = 0.167$.

The fifth measure we describe is the *trigram similarity*, which extends the bigram similarity by considering n-grams of length 3 (trigrams) in place of bigrams. We formally define it as follows.

Definition 19 (Trigram similarity). Let $L = (UL, I, \ell)$ and $L' = (UL, I, \ell')$ be two event logs, whose case sets are $S(L)$ and $S(L')$ respectively. We denote as $\text{occurs3}(\langle e, e', e'' \rangle, L)$ the indicator function that returns 1 if there exists a case $\sigma \in S(L)$ such that $\langle e, e', e'' \rangle$ is a segment of it.

$$\text{occurs3}(\langle e, e', e'' \rangle, L) = \begin{cases} 1 & \text{if there exists } \sigma \in S(L) \text{ s.t. } \langle e, e', e'' \rangle \subseteq \sigma \\ 0 & \text{otherwise} \end{cases} \quad (23)$$

The *trigram similarity* $L2L_{3\text{gram}}$ is computed dividing by the cardinality of $S(L)$ the average of trigrams in the cases of L that also occur in L' as follows:

$$L2L_{3\text{gram}}(L, L') = \frac{\sum_{\sigma \in S(L)} \left(\frac{1}{|\sigma|-2} \sum_{i=2}^{|\sigma|-1} \text{occurs3}(\langle \sigma(i-1), \sigma(i), \sigma(i+1) \rangle, L') \right)}{|I|} \quad (24)$$

Table 6: Computation of $L2L_{3\text{gram}}$ for logs L and L' in Fig. 13

$\text{occurs3}(\langle A_1, B_4, C_5 \rangle, L') = 0$
$\text{occurs3}(\langle A_2, C_6, B_7 \rangle, L') = 0$
$\text{occurs3}(\langle A_3, B_8, C_9 \rangle, L') = 0$

For example, for every trigram $\langle e, e', e'' \rangle$ in L (depicted in Fig. 13(a)), we check if it occurs in L' (depicted in Fig. 13(b)). As shown in Table 6, L and L' do not have trigrams in common. Therefore, $L2L_{3\text{gram}} = 0$. If the case assignments of e_5 and e_6 were swapped (i.e., e_5 and e_6 had been assigned with σ'_3 and σ'_2 , respectively), then the value of $L2L_{3\text{gram}}$ would be $\frac{1}{3}$.

The last measure we describe is the *case similarity*, which checks the extent to which a pair of logs identically correlate cases as a whole. We formally define it as follows.

Definition 20 (Case similarity, $L2L_{\text{case}}$). Let $L = (UL, I, \ell)$ and $L' = (UL, I, \ell')$ be two event logs, whose case sets are $S(L)$ and $S(L')$ respectively. The *case similarity*, $L2L_{\text{case}}$, amounts to the number of cases that are equal in L and L' divided by the total number of cases:

$$L2L_{\text{case}}(L, L') = \frac{|S(L) \cap S(L')|}{|I|} \quad (25)$$

Notice that we indicate with $S(L) \cap S(L')$ the cases in L that have an equal one in L' . As the number of cases is the same in L and L' , $L2L_{\text{case}}$ can be considered as a Sørensen-Dice coefficient for $S(L)$ and $S(L')$ [60]. In the example, given L (depicted in Fig. 13(a)) and L' (depicted in Fig. 13(b)), there are no equal cases occurring in L and L' . Therefore, $L2L_{\text{case}} = 0$. If the case assignments of e_5 and e_6 were swapped (i.e., $\ell(e_5) = \sigma'_3$ and $\ell(e_6) = \sigma'_2$), then the value of $L2L_{\text{case}}$ would be $\frac{1}{3}$.

Up to this point, we have presented the measures following an order given by the increasing amount of details they consider in the comparison. We shall refer to the ones at a higher level of abstraction as more *relaxed* (e.g., $L2L_{\text{trace}}$), as opposed to the *stricter* ones (e.g., $L2L_{\text{case}}$). In the following, we present measures that deal with time deviations.

5.2. Log-to-log time deviation

The log-to-log time deviation category consists of two measures which use the symmetric mean absolute percentage error (SMAPE) to compute the time deviation between a pair of logs. Their values range from 0 to 1, where 0 indicates the highest similarity between the two logs.

The first measure we describe here is the *event-time deviation*. It investigates the extent to which the two logs deviate in terms of the elapsed time of events. We formally define it as follows.

Table 7: Computation of SMAPE_{ET} for logs L and L' in Fig. 13

 (a) Elapsed time (in min) of events in L

$\text{ET}(\sigma_1, B_4) = 90$	$\text{ET}(\sigma_1, C_5) = 30$
$\text{ET}(\sigma_1, C_6) = 150$	$\text{ET}(\sigma_1, B_7) = 60$
$\text{ET}(\sigma_1, B_8) = 190$	$\text{ET}(\sigma_1, C_9) = 40$

 (b) Elapsed time (in min) of events in L'

$\text{ET}(\sigma'_1, B_4) = 90$	$\text{ET}(\sigma'_1, C_5) = 90$
$\text{ET}(\sigma'_1, C_6) = 120$	$\text{ET}(\sigma'_1, B_7) = 120$
$\text{ET}(\sigma'_1, B_8) = 70$	$\text{ET}(\sigma'_1, C_9) = 200$

Definition 21 (Event-time deviation). Let $L = (\text{UL}, I, \ell)$ and $L' = (\text{UL}, I, \ell')$ be two event logs defined over a common universe of events E . Let $\text{ET}(\sigma, e)$ be the elapsed time (ET) of event e in case σ as per Eq. (8). The *event-time deviation*, SMAPE_{ET} , is the Symmetric Mean Absolute Percentage Error (SMAPE) of the elapsed time of events between L and L' , computed as follows:

$$\text{SMAPE}_{\text{ET}}(L, L') = \frac{\sum_{e \in E} \frac{|\text{ET}(\sigma, e) - \text{ET}(\sigma', e)|}{|\text{ET}(\sigma, e)| + |\text{ET}(\sigma', e)|}}{|E| - |I|}$$

with $\sigma = L(\ell(e))$, $\sigma' = L'(\ell(e))$

(26)

For example, the elapsed time of C_5 in L , given that $L(\ell(C_5)) = \sigma_1$, is $C_5.Ts - B_4.Ts = 30$ min, whereas the elapsed time of C_5 in L' , given that $L'(\ell(C_5)) = \sigma'_1$, is $C_5.Ts - A_2.Ts = 90$ min. We compute SMAPE_{ET} using the elapsed time of the events in L (depicted in Fig. 13(a)), and the elapsed time of the events in L' (depicted in Fig. 13(b)) as shown in Table 7. As a result, $\text{SMAPE}_{\text{ET}} = 0.35$.

The second measure we describe is the *case cycle time deviation*, assessing the extent to which two logs differ in terms of the cases' cycle time. To compare pairs of cases, we consider those that begin with the same start event, as seen in Section 5.1 with $\text{L2L}_{\text{first}}$. We formally define the measure as follows.

Definition 22 (Case cycle time deviation). Let $L = (\text{UL}, I, \ell)$ and $L' = (\text{UL}, I, \ell')$ be two event logs defined over a common universe of events E . Let $\text{CT}(\sigma)$ be the cycle time of case σ , computed as follows [63, 64]:

$$\text{CT}(\sigma) = \sigma(|\sigma|).Ts - \sigma(1).Ts \quad (27)$$

The *case cycle time deviation* SMAPE_{CT} is the symmetric mean absolute percentage error of the cycle time between cases in L and L' :

$$\text{SMAPE}_{\text{CT}}(L, L') = \frac{1}{|I|} \times \sum_{\substack{\sigma \in L, \\ \sigma' \in L', \\ \sigma(1) = \sigma'(1)}} \frac{|\text{CT}(\sigma) - \text{CT}(\sigma')|}{|\text{CT}(\sigma)| + |\text{CT}(\sigma')|} \quad (28)$$

For example, we compute the cycle time of case σ_1 in L based on the first and the last events in the case: $\text{CT}(\sigma_1) = C_5.Ts - A_1.Ts = 1$ h. The cycle time of case σ'_1 in L' is

 Table 8: Computation of SMAPE_{CT} for logs L and L' in Fig. 13

 (a) Cycle time (in hours) of events in L

$\text{CT}(\sigma_1) = C_5.Ts - A_1.Ts = 1$
$\text{CT}(\sigma_2) = B_7.Ts - A_2.Ts = 3.5$
$\text{CT}(\sigma_3) = C_9.Ts - A_3.Ts = 3.8$

 (b) Cycle time (in hours) of events in L'

$\text{CT}(\sigma'_1) = C_9.Ts - A_1.Ts = 4.8$
$\text{CT}(\sigma'_2) = B_7.Ts - A_2.Ts = 3.5$
$\text{CT}(\sigma'_3) = B_8.Ts - A_3.Ts = 3.17$

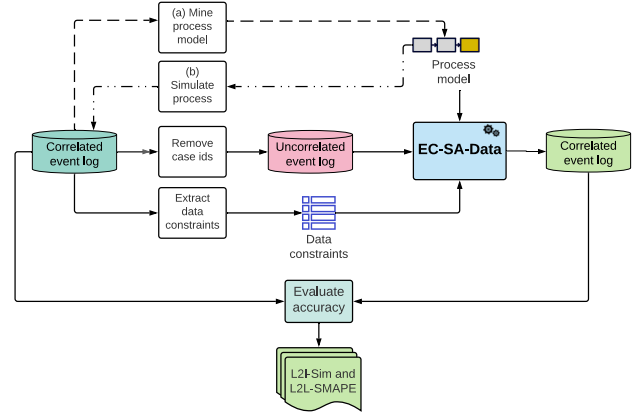


Figure 14: Evaluation steps

$\text{CT}(\sigma'_1) = C_9.Ts - A_1.Ts = 4.8$ h. We compute SMAPE_{CT} comparing the cycle time of the cases in event log L (depicted in Fig. 13(a)) and the ones in event log L' (depicted in Fig. 13(b)) having the same start event, as shown in Table 8. As a result, $\text{SMAPE}_{\text{CT}} = 0.25$.

Thus far, we have presented the measures we use to assess the outcome of our approach. Next, we illustrate our experimental results and evaluate them by means of the aforementioned measures.

6. Evaluation

We implemented a prototype tool for EC-SA-Data.³ Using this tool, we conducted six experiments to evaluate the accuracy and time performance of our approach. Furthermore, we compared the results with the state-of-the-art tools EC-SA [18], DCIc [17] and E-Max [16].

6.1. Design

Figure 14 illustrates our evaluation process. The primary input is a correlated event log with defined cases; we refer to it as the *original log*. We remove the case identifiers from it and thereby create an uncorrelated event log. Thereupon, we run our implemented technique and measure its accuracy using the two categories of measures defined in Section 5. The first category is the *log-to-log similarity*, which assesses the extent to which EC-SA-Data generates a correlated log (we refer to it as L' in Def.s 15 to 22) that is consistent with

³<https://github.com/DinaBayomie/EC-SA-Data/releases/tag/EC-SA-Data>

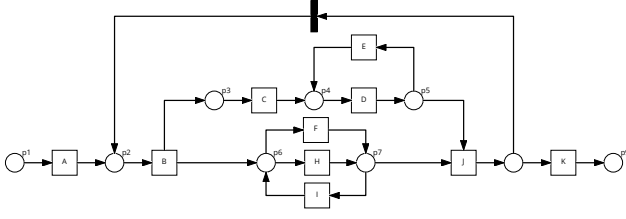


Figure 15: Process model used in the first experiment

Table 9: A sample of the data constraints used in the first experiment

$C_1 : \sigma(i-1).LoanType = \sigma(i).LoanType$
$C_2 : \sigma(i-1).Amount = \sigma(i).Amount$
$C_3 : \text{IF } \sigma(i).Act = B \wedge \sigma(j).Act = A \text{ THEN } \sigma(i).Resource \neq \sigma(j).Resource$
$C_4 : \text{IF } \sigma(i).Act = C \wedge \sigma(j).Act = B \text{ THEN } \sigma(i).Resource = \sigma(j).Resource$
$C_5 : \text{IF } \sigma(i).Act = D \wedge \sigma(j).Act = A \wedge \sigma(i).Status = \text{notifying} \text{ THEN } \sigma(i).Resource = \sigma(j).Resource$
$C_7 : \text{IF } \sigma(i).Act = B \text{ THEN } 5 \leq (\sigma(i).Ts - \sigma(i-1).Ts) \leq 50$
$C_7 : \text{IF } \sigma(i).Act = C \text{ THEN } 10 \leq (\sigma(i).Ts - \sigma(i-1).Ts) \leq 90$
$C_8 : \text{IF } \sigma(i).Act = D \text{ THEN } 20 \leq (\sigma(i).Ts - \sigma(i-1).Ts) \leq 70$

the original log (we refer to it as L in Def.s 15 to 22) in terms of trace-based and case-based distances. The second category is *log-to-log time deviation*, which considers the temporal distance of events' elapsed times and cases' cycle times.

We conduct six experiments (two of which are two-staged), which differ in terms of input and objectives. Overall, we aim to assess the effectiveness of our technique, taking into account the evaluation principles described in [65]. Next, we describe their setups and goals in detail. In the following subsection, we report on the attained results.

The first experiment performs a sensitivity analysis on the impact on the accuracy of the output log entailed by increasing (i) the number of overall cases (i.e., the volume of input data), and (ii) the average work-in-progress cases (WIP, i.e., the density of the overlapping cases at a point in time). We simulated the process depicted in Fig. 15 to generate 60 input logs. The process includes the fundamental behavior patterns, i.e., sequence, concurrency, exclusiveness, and cycles [64]. In addition to it, we utilized the data constraints shown in Table 9. We tuned the inter-arrival time between starting cases to set up the WIP. The produced logs, thus, present the following characteristics: (i) a varying number of cases, ranging between 100 and 1000 at steps of 100; (ii) different inter-arrival time based on the cycle time (CT) of the process (i.e., the time spent by a case from start to end), ranging between $CT/32$ and CT , at multiplicative steps of 2. We modified the simulated logs by adding extra data attributes. Table 10 illustrates a sample of an uncorrelated log we used.⁴

The second experiment performs a sensitivity analysis of the process structure's impact on the accuracy of the generated log. The starting point for the evaluation is a

⁴The full set of event logs we created is available at <https://doi.org/10.6084/m9.figshare.20736706.v1>.

Table 10: A sample of an uncorrelated log used in the first experiment

Event Id	Activity	Timestamp	Loan Type	Amount	Resource	Status
1	A	21/12/2019 09:00	Car	10000	R1	submitting
2	A	21/12/2019 09:05	Health	10000	R3	submitting
3	B	21/12/2019 09:15	Health	10000	R2	processing
4	A	21/12/2019 09:30	House	50000	R2	submitting
5	A	21/12/2019 09:35	Personal	66000	R2	submitting
6	B	21/12/2019 09:38	House	50000	R1	pricing
7	C	21/12/2019 09:42	Car	10000	R1	processing
8	B	21/12/2019 09:58	Personal	66000	R1	processing
9	D	21/12/2019 10:10	House	50000	R3	notifying
10	C	21/12/2019 10:20	House	50000	R1	pricing

collection of 18 process models previously used to assess the performance of algorithms for concept drift detection in process mining [66]. The reason for selecting this collection of models is that they contain a representative set of combinations of control-flow structures, such as a cycle nested inside a parallel block, a parallel block nested inside a conditional block, a composite process fragment nested inside a loop, etc. These models are obtained from a baseline model by systematically applying one out of twelve control-flow change patterns described in [67]. Every model in this existing collection contains at least one loop block. To include acyclic models in our evaluation, we modified three models in the existing collection by removing back-edges (and thus breaking the cycles). In total, we obtained 21 process models.⁵ We fed the 21 models into the BIMP simulator⁶ to generate an event log of a size of 100 cases from each of these models.⁷

The third experiment assesses the accuracy improvement our approach yields when data constraints are provided together with an input process model. For this experiment, we used four real-world datasets from the benchmark of Augusto et al. [72] based on the publicly available event logs in the BPIC repository. Table 11 shows descriptive statistics and complexity measures about the four logs. The first three categories focus on the size complexity regarding the number of cases and events and the trace length. The last category focuses on entropy and variance complexity measures from Augusto et al. [73] based on variant entropy (nvar-e), sequence entropy (nseq-e), and the number of acyclic paths (LOD). We mined the process models from the original logs using a state-of-the-art discovery technique, namely Split Miner [11]. We extracted the data constraints by visual inspection and analysis of those event logs – Table 13 summarizes our findings.⁸ Thereupon, we compared the results attained with our approach with those of EC-SA [18], as the latter does not provide the capability of including data

⁵The process models we used for simulation are publicly available at <https://doi.org/10.6084/m9.figshare.20732095>.

⁶<http://bimp.cs.ut.ee/simulator>

⁷The event logs we created via simulation for our tests are publicly available at <https://doi.org/10.6084/m9.figshare.20732512>.

⁸The full set of equality and IF-THEN constraints we used is available at <https://doi.org/10.6084/m9.figshare.20740912.v1>.

Table 11: Descriptive statistics of real-world logs

Event log	Cases		Events		Trace length			Complexity measures		
	Total	Dst.%	Total	Dst.%	Min	Avg	Max	nvar-e	nseq-e	LOD
BPIC13 _{cp} [68]	1487	12.3	6660	7	1	4	35	0.705	0.311	2.4
BPIC13 _{inc} [69]	7554	20.0	65 533	13	1	9	123	0.718	0.405	2.6
BPIC15 _{lf} [70]	902	32.7	21 656	70	5	24	50	0.652	0.419	24
BPIC17 _f [71]	21 861	40.1	714 198	41	11	33	113	0.777	0.358	14.4

Table 12: Fitness and precision of the process models mined from the real-world logs

Event log	SM-mined model		IM-mined model	
	Fitness	Precision	Fitness	Precision
BPIC13 _{cp} [68]	0.94	0.97	0.82	1.0
BPIC13 _{inc} [69]	0.91	0.98	0.92	0.54
BPIC15 _{lf} [70]	0.90	0.88	0.97	0.57
BPIC17 _f [71]	0.95	0.85	0.98	0.70

Table 13: Number of equality and IF-THEN constraints per real-world logs

Event log	Equality constraints	IF-THEN constraints
BPIC13 _{cp} [68]	3	3
BPIC13 _{inc} [69]	3	2
BPIC15 _{lf} [70]	5	0
BPIC17 _f [71]	3	7

constraints to steer the assignment of case identifiers to the events.

The fourth experiment performs a sensitivity analysis that investigates the effect of the constraints on the accuracy of the log generated by EC-SA-Data. We used the BPIC17 event log [71] as filtered by Augusto et al. [72] (hence the ‘f’ subscript in “BPIC17_f” in the tables and figures), as we observed that a relatively high number of data constraints define its behavior, compared to other real-world event logs (see Table 13). In particular, we inferred ten rules that regulate the behavior of the process behind the BPIC17 log. Their expressions are reported in Appendix A. There are three data-attribute equality rules and seven IF-THEN constraints. Six IF-THEN rules are based on the matching of the operating resources over some activities within a case. The seventh IF-THEN rule is a correlation rule based on the equality of two different data attributes over some activities within a case. This experiment is divided in two stages, each aimed at investigating a separate aspect. The first aspect pertains to the effect of an increase in the number of used constraints on the accuracy of the generated log. We gradually increase the number of used constraints from zero to ten. Notice that the order of the constraints does not affect the accuracy of the output. Thereby, we investigate the impact of increasing the knowledge about the data constraints on the accuracy of the generated logs.

The second aspect concerns the impact of the reliability of those constraints on the accuracy of the generated logs. We impersonate the business analysts in their iterative endeavor. While inspecting the data, some rules occur as evident. Other ones are less certain or harder to confirm. Nevertheless, they could use all of them in an attempt to drive the automated correlation, although some could possibly misrepresent the data, thereby misleading the technique. To mimic this situation, we run successive tests adding (a) four correct rules (three of which are data-attribute equality constraints and one is an IF-THEN constraint), and then (b) three inexact rules. Notice that we omit three correct rules out of the ten aforementioned ones to mimic the missing knowledge.

The fifth experiment performs a sensitivity analysis on the impact that the input process model’s quality (in terms of its ability to represent the behavior as reflected in the data) has on the accuracy of the generated log. For this experiment, we used the same logs and data constraints as in the third experiment (see Tables 11 and 13, respectively). This experiment is split in two stages, too.

First, we observe the effect of having models that do not guarantee full fitness and precision with respect to the data. A fitness of less than 100 % entails that some cases in the event log are not reflected in the model. Consequently, the model can potentially induce errors in the correlation by excluding execution paths that were instead plausible. A precision of less than 100 % indicates that the model allows for a wider execution behavior than the one observed in the event log. The effect is that the possible alternatives for the correlation decision unnecessarily increase. To this end, we used process models mined from the four real-world datasets by two state-of-the-art automated discovery methods: the Split Miner [11] and the Inductive Miner [74]. Table 12 shows the mined models’ fitness and precision as per the benchmark of Augusto et al. [72].

Second, we analyze the impact of a decreasing process model fitness on the output log’s accuracy. To this extent, we used the Inductive Miner to mine five models with different noise thresholds (0 % to 100 % at steps of 25 %). Raising values for the noise threshold determines that cases are discarded by the discovery algorithm based on their frequency, from the lowest to the highest. Notice that, consequently, the model fitness decreases. The five models generated, in order of noise threshold ascending from 0 % to 100 % have a fitness of 100 %, 82 %, 76 %, 67 % and 45 %.

Table 14: Number of event-time constraints used for the comparative evaluation

Event log	Event-time constraints
BPIC13 _{cp} [68]	4
BPIC13 _{inc} [69]	4
BPIC15 _{lr} [70]	70
BPIC17 _r [71]	18

Looking at the data constraints and control-flow model as normative rather than descriptive, the fourth and fifth experiment demonstrate the effect on the output's accuracy caused by non-compliant events [75]: the second stage of the former simulates the presence of events that do not satisfy an increasing number of rules and the second stage of the latter can reproduce the occurrence of events that deviate from the expected control flow.

The sixth experiment compares our approach against DCIc [17] and E-Max [16] with respect to accuracy and execution time. We used the four real-world logs in Table 11 and used as reference process models those mined using the Inductive Miner with default settings. We employed the equality and IF-THEN constraints already considered for the third experiment (see Table 13). E-Max does not require any input data except a sequence of event names; thus, we reshaped the input logs we fed into it accordingly. DCIc requires heuristic information about the activities' execution behavior as additional data. Therefore, we computed activity durations and used the quartiles to represent the ranges so that the first quartile is the lower bound and the third quartile is the upper bound. We translated the additional information required by DCIc into event-time constraints.⁹ A sample of those are reported in Appendix B. Table 14 reports the number of event-time constraints we defined for each event log.

6.2. Results

In the following, we describe in detail the outcome of our experiments, following the order we used in Section 6.1.

6.2.1. Impact of data volume and density on accuracy.

The first experiment studies the impact of increasing log size (volume) and WIP (density) on the accuracy of EC-SA-Data. As we have two varying variables (log size and WIP), we examine their effects individually. Figures 16 to 18 visually depict the separate trends having a single polyline per WIP value. In the text, we shall describe the effect of variations in the log size by averaging the obtained results over the WIP values. Vice-versa, we discuss the impact of altering the WIP by considering the average of the results over the different log sizes. In both situations, we will simply say "on average" for short. As we detail in the following, our results demonstrate that the stricter the measure is, the steeper the curve gets.

Figure 16 shows how log size and WIP affect log-to-log similarity measures. Markedly, a relaxed similarity measure such as $L2L_{trace}$ drops by around 4 % on average from the situation in which logs consist of 100 cases to when logs contain 1000 cases, as Fig. 16(a) illustrates. Also, $L2L_{trace}$ decreases by around 2 % on average when the inter-arrival rate goes from a value that equates the process cycle time (1 CT) to one thirty-second thereof ($1/32$ CT). As we see in Fig. 16(d), the stricter $L2L_{2gram}$ similarity measure decreases by around 5 % on average having logs whose cardinality increases from 100 cases to 1000 cases. Also, its value goes down by circa 3 % on average as the inter-arrival rate goes from 1 CT to $1/32$ CT. Figure 16(f) shows that the strictest similarity measure, $L2L_{case}$, falls by around 19 % on average when cases increase from 100 to 1000, and by approximately 18 % on average as the inter-arrival rate reaches a thirty-second of the process cycle time from the initial value of a whole cycle time.

Figure 17 illustrates how log size and WIP affect log-to-log time deviation measures. As Fig. 17(a) depicts, the drop of $SMAPE_{ET}$ and $SMAPE_{CT}$ averages to around 14 % and 13 %, respectively, when the number of cases in the logs rises from 100 to 1000. On average, $SMAPE_{ET}$ and $SMAPE_{CT}$ drop by about 8 % and 15 %, respectively, as the inter-arrival goes from 1 CT to $1/32$ CT, as shown in Fig. 17(b).

Figure 18 shows how log size and WIP affect execution time performance. The execution time increases by around 0.15 h on average when the cardinality of logs goes from 100 cases to 1000 cases. We observe that an increment of 100 cases implies an average increase from 0.02 h (having the WIP equal to 1 CT) to 0.05 h (when the WIP is $1/32$ CT). Considering the inter-arrival rate, execution time rises by circa 0.12 h on average when the WIP goes from 1 CT to $1/32$ CT. More in detail, halving the WIP induces an increase from about 0.02 h (with 100 cases) to 0.13 h (with 1000 cases).

These drops in performance occur because bigger logs bring more options to assign events, and the inter-arrival rate influences the number of overlapping cases. The combination of higher volumes of cases and their density increases the uncertainty of the correlation decision step, thereby affecting the accuracy of the technique. Moreover, it broadens the range of possible assignments per event, which affects the performance in terms of execution time.

6.2.2. Impact of the process structure on accuracy.

Figures 19 and 20 show the results of the second experiment, which studies the effect of process structure on the accuracy of the correlation process of EC-SA-Data.

Figure 19(a) illustrates how the process structure affects log-to-log similarity measures. Markedly, EC-SA-Data performs evenly well with the different behavioral structures such as nested cycles, short loops, parallel blocks, conditional blocks, and parallel blocks or conditional blocks nested inside a cyclic structure and vice versa. However, as we observe in the figure, models 4 and 15 determine the lowest values for similarity measures: among others, $L2L_{2gram}$ drops

⁹The full set of time constraints we used is available at <https://doi.org/10.6084/m9.figshare.20741419.v1>.

Event-Case Correlation for Process Mining using Probabilistic Optimization

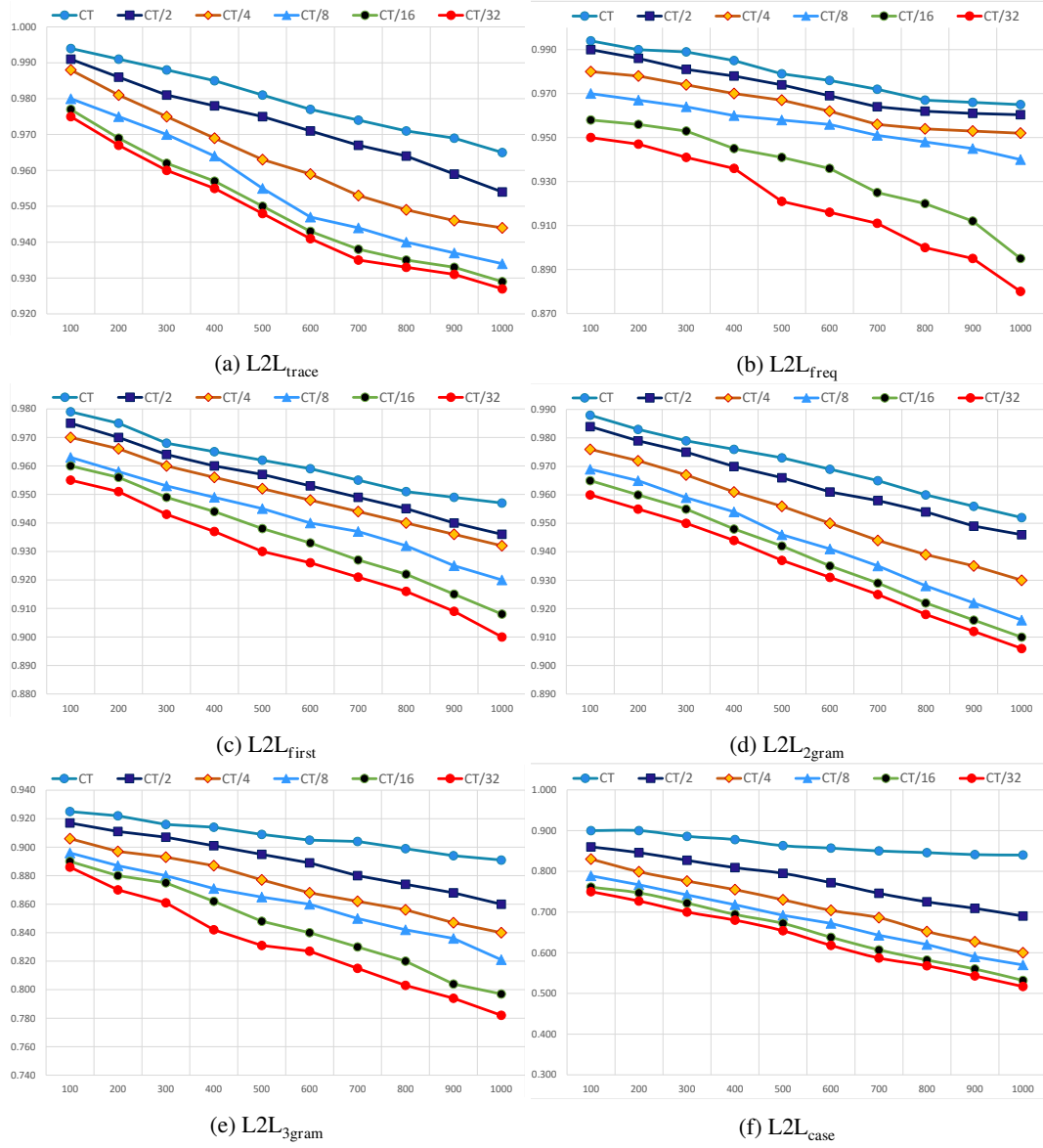


Figure 16: The impact of increasing log size (on the horizontal axis) and WIP (differentiated by the color and shape of the poly-lines) on log-to-log similarity measures (on the vertical axis)

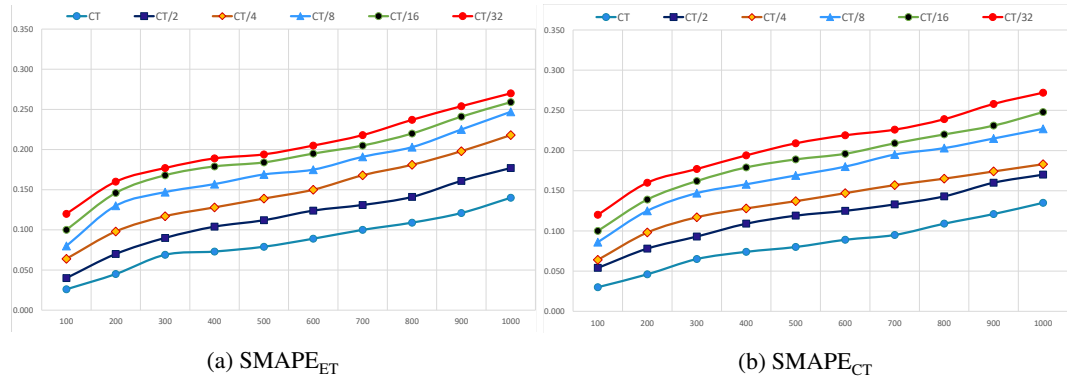


Figure 17: The impact of increasing log size (on the horizontal axis) and WIP (differentiated by the color and shape of the poly-lines) on log-to-log time deviation measures (on the vertical axis)

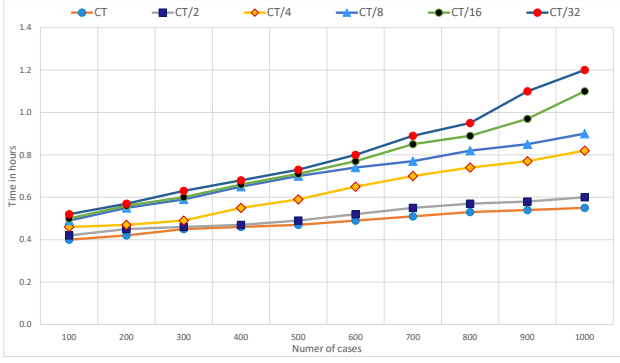


Figure 18: The impact of increasing log size (on the horizontal axis) and WIP (differentiated by the color and shape of the poly-lines) on execution time

to 90 % and 88 %, respectively, while $L2L_{case}$ goes down to 83 % in both circumstances. Figure 19(b) portrays how the process structure affects log-to-log deviation measures. Again, models 4 and 15 mark the worst results, also in terms of log-to-log time deviation measures: $SMAPE_{ET}$ rises to 16 % and 15 %, respectively, and $SMAPE_{CT}$ hits the peaks of 25 % and 26 %. The reason for these sub-standard outcomes is that models 4 and 15 include duplicated activities, i.e., activities occurring multiple times in different fragments of the model. Duplicated activities have the undesirable effect of misleading the assignment process as the events could be potentially associated with different cases that enable the same activity in separate stages of their execution.

Figure 20 shows how the process structure affects the computation time of EC-SA-Data. As it turns out, the acyclic models (19, 20, and 21) lead to a faster execution than the cyclic models – specifically, by 0.21 h on average. The reason is that cases of acyclic models normally consist of fewer events than those that record the enactment of cyclic ones. Also, similarly to what described above with duplicated activities, a cyclic behavior can lead to a wider spectrum of possible assignments per event, as different cases could enable the corresponding activity at different stages.

6.2.3. Impact of the data constraints on accuracy.

Figures 21 to 23 show the results of the third experiment, which studies the impact that adding data constraints has on the correlation accuracy. Thereby, it compares the results with EC-SA (which, unlike EC-SA-Data, does not allow for the input of data constraints). We can see that doing so improves the accuracy. Indeed, EC-SA-Data outperforms EC-SA.

Figure 21 shows that $L2L_{trace}$, $L2L_{2gram}$ and $L2L_{case}$ increase by around 6 %, 15 % and 28 % on average when constraints are in use, respectively. Notably, using data constraints with EC-SA-Data on the BPIC17 log dramatically improves the event correlation quality as it can be observed in Fig. 21(f) – notice that $L2L_{case}$ increases by 46 %. Figure 22 highlights that also the time deviation decreases when constraints are in use, as $SMAPE_{ET}$ and $SMAPE_{CT}$ rise by 19 % and 21 %, respectively. With the BPIC15_{lf} log, instead,

using the data constraints leads to a lesser improvement. $L2L_{2gram}$ and $L2L_{3gram}$ increase by 8 % and 5 %, respectively (see Figs. 21(d) and 21(e)). Nevertheless, it still increases $L2L_{case}$ by 13 %. Figure 22 evidences that also the time deviation decreases to a lower extent when constraints are in use with BPIC15_{lf}, as $SMAPE_{ET}$ and $SMAPE_{CT}$ get reduced by 9 % and 7 %, respectively. Possible factors leading to a less noticeable improvement with data constraints can be that (i) we specified only equality constraints to define its behavior beyond the control flow (see Table 13), and (ii) the five data attributes the constraints are exerted on have unevenly distributed values over the caes (e.g., the (Case)_Responsible_Actor attribute has 19 domain values, one of which is used in 43 % of the 902 cases).

Using constraints enhances the correlation process as they prune out the violating options for case assignment. Consequently, the uncertainty of the correlation decision step decreases and this positively affects the quality of the generated log. However, the usage of data constraints affects the performance of EC-SA-Data in terms of execution time, as shown in Fig. 23. The reason is, constraints must be verified at every assignment step. Therefore, the more the data constraints, the higher the overall computation time gets. For instance, EC-SA-Data ran for 13 h to complete the execution with the BPIC17 log using 10 constraints, in contrast with the 8.6 h needed in absence of constraints. The processing of the BPIC15_{lf} log required 6.2 h with 4 constraints and 5.5 h without constraints.

6.2.4. Impact of the quality of constraints on accuracy.

Figures 24 and 25 show the results of the fourth experiment, which studies the effect of the number and correctness of constraints on the accuracy of the correlation process of EC-SA-Data.

Figure 24(a) highlights that using more constraints improves the log-to-log similarity accuracy measures of the generated logs. For instance, $L2L_{trace}$, $L2L_{2gram}$ and $L2L_{case}$ increase by around 8 %, 24 % and 46 % when the constraints reach the peak of 10 correct ones. Figure 24(b) evidences that the log-to-log time deviation accuracy measures decrease too: in particular, $SMAPE_{ET}$ and $SMAPE_{CT}$ drop by up to 36 % and 40 %, respectively. These improvements materialize as more knowledge in terms of data constraints helps to discard case assignment possibilities and, therefore, decrease the uncertainty of the correlation decision.

Figures 24(c) and 24(d) illustrate the effect of including incorrect constraints in the decision process. These constraints are intentionally inconsistent with the data in order to study the impact of the knowledge quality on the accuracy of the technique. We observe that using 3 wrong constraints makes $L2L_{trace}$ reduce by 1 %, $L2L_{2gram}$ by 4 % and $L2L_{case}$ by 5 %. Also, it affects the time deviation measures as $SMAPE_{ET}$ and $SMAPE_{CT}$ increase by 4 % and 2 %, respectively. We observe that incorrect constraints affect the overall accuracy though not severely thanks to the positive impact of the other (correct) ones, and of the control-flow model. This scenario is meant to resemble a realistic situation in which

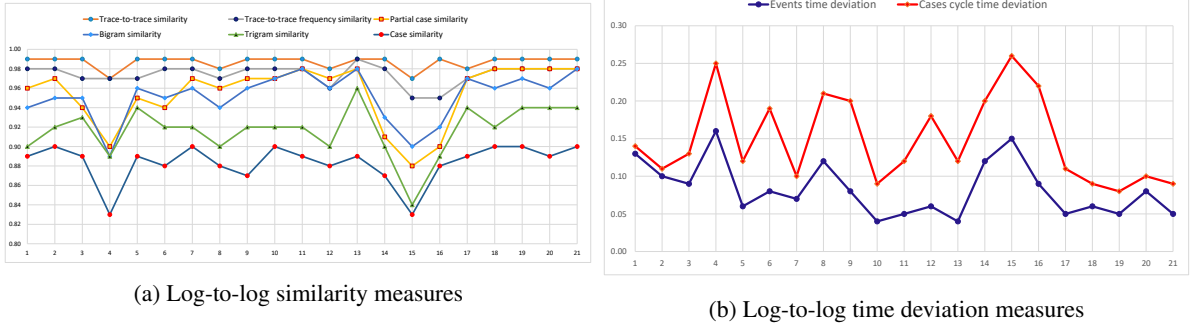


Figure 19: Impact of the process structure on accuracy measures

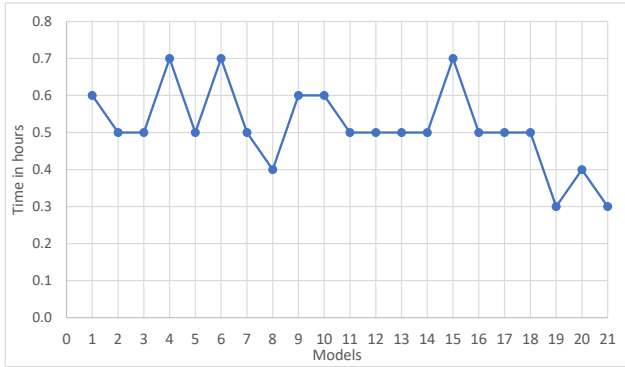


Figure 20: Impact of the process structure on execution time

constraints are available but without the certainty that all of those are consistent with the event data at hand. We can see that EC-SA-Data still provides a correlated log with sufficient accuracy.

Figure 25 shows the impact of increasing the number of used data constraints on the performance. The execution time increases by 4.2 h when the number of used constraints rises from zero to ten. The reason is that constraints must be verified at every event correlation step twice, as explained in Section 4: the first time for the case assignment decision, and the second time to compute the energy cost function.

6.2.5. Impact of the process model quality on accuracy.

Figures 26 and 27 depict the results of the first stage of the fifth experiment, which studies the impact of using models with different fitness and precision measures on the accuracy of the generated log.

Figure 26 shows that the log-to-log similarity measures change slightly based on the model's fitness and precision, as $L2L_{freq}$, $L2L_{2gram}$ and $L2L_{case}$ among others differ by around 0.25 %, 0.5 % and 1.13 % on average in our tests. We recall that the results stem from models mined by the Inductive Miner (henceforth, IM for short) [74] and the Split Miner (SM) [11]. Let us consider the BPIC13_{cp} log, for example. As reported in Table 12, the SM-mined model has a fitness of 94 % and a precision of 97 %, thus 12 % above and 3 % below the fitness and precision of the IM-mined model, respectively. The increase in fitness is four times higher than the absolute

value of the decrease in precision. As shown in Figure 26, $L2L_{case}$ and $L2L_{3gram}$ increase by 2.5 % and 1 %, respectively, when using the SM-mined model. The other four measures do not exhibit a significant change.

Figure 27 evidences that also the time deviation is mildly affected by the model's fitness and precision, as $SMAPE_{ET}$ and $SMAPE_{CT}$ change by 0.75 % and 1.5 % on average in our experiments, respectively. With the BPIC15_{lf} log, for instance, differences of -7 % in fitness and +31 % in precision occur between the SM-mined model and IM-mined model (see Table 12). The absolute value of the decrease in fitness is negligible with respect to the increase in precision, as the former amounts to less than a fourth of the latter. $SMAPE_{ET}$ and $SMAPE_{CT}$ increase by 2 % and 3 %, respectively, using the SM-mined model.

Figures 28 and 29 show the results of the second stage of the fifth experiment, which studies the impact of increasing the noise threshold of the Inductive Miner to create the control-flow model. We recall that an increase in the noise threshold implies a decrease in the model fitness. Figure 28(a) confirms that the effect is a lower accuracy of the output log. For instance, $L2L_{trace}$, $L2L_{2gram}$ and $L2L_{case}$ drop by around 3 %, 7 % and 14 %, respectively, when the noise threshold increases from 0 to 1 (hence, the fitness lowers from 100 % to 45 %). Figure 28(b) evidences that $SMAPE_{ET}$ and $SMAPE_{CT}$ rise by up to 3 % and 6 %, respectively, under the same conditions.

We remark that the wide oscillations in the quality measures of the control-flow models are reverberated in the output log accuracy, yet the effect is reduced as differences are narrower. We motivate this mitigation with the use of data constraints, which lessen the otherwise more severe impact of using inaccurate models.

Finally, Fig. 29 shows that the execution time is slightly affected by the decrease in the model fitness, as it rises by 0.9 h. This effect is determined by the fact that the events that cannot be assigned as their activities are not enabled as per the control-flow model are more numerous, and thus checks against the data constraints are more frequent.

6.2.6. Comparative evaluation.

Figures 30 to 32 show the results of the sixth experiment, which compares our approach with DCIc [17] and E-Max [16]

Event-Case Correlation for Process Mining using Probabilistic Optimization

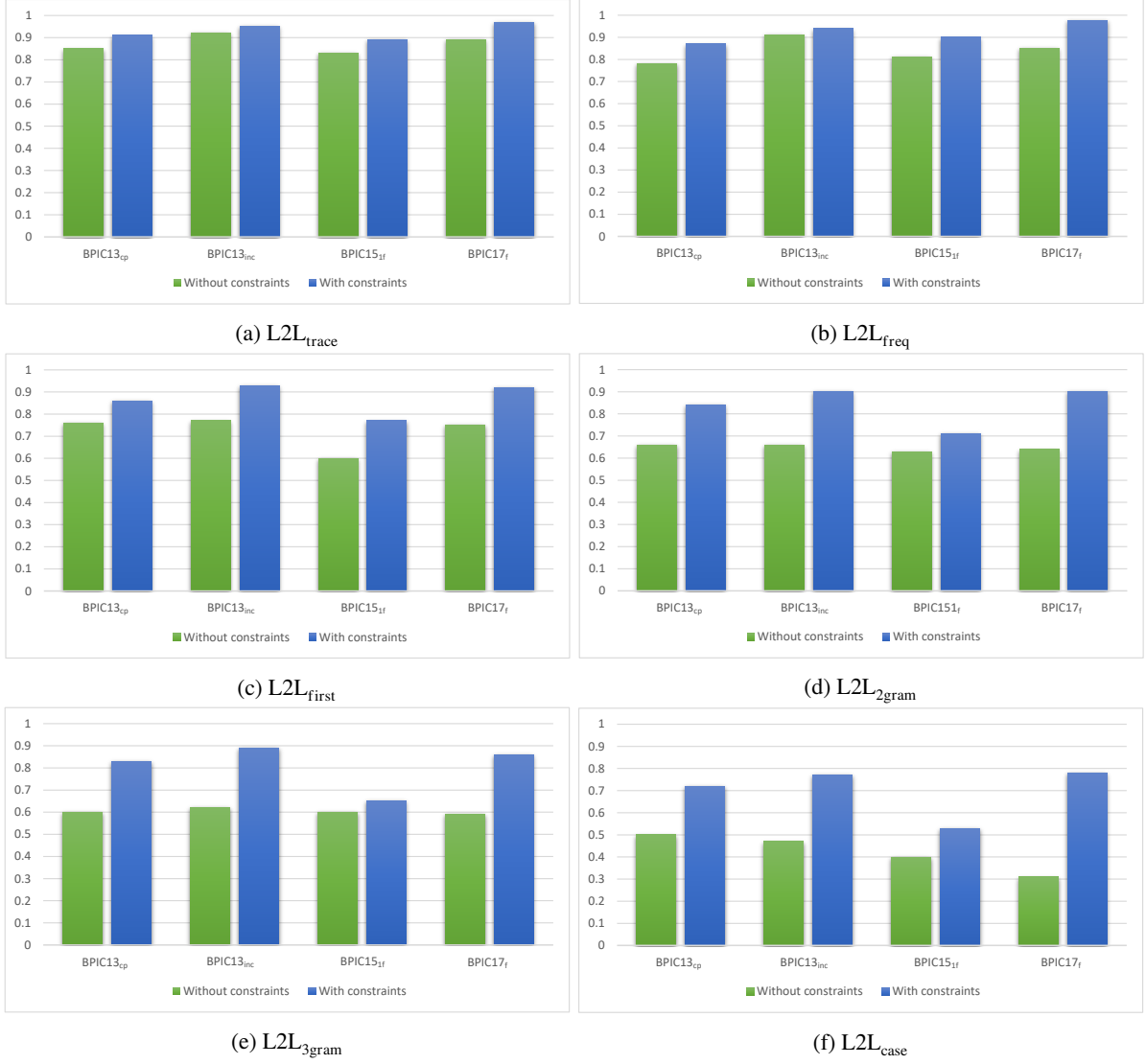


Figure 21: Impact of using data constraints with real-world logs on log-to-log similarity measures

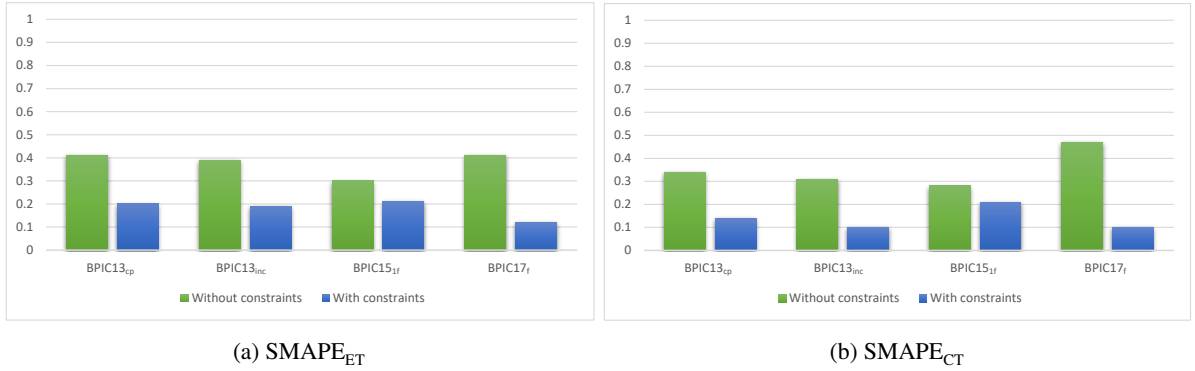


Figure 22: Impact of using data constraints with real-world logs on log-to-log time deviation measures

as the correlation techniques' baseline. We can see that EC-SA-Data outperforms DCIc and particularly E-Max in terms of accuracy.

Figure 30 focuses on the log-to-log similarity measures. Among others, with EC-SA-Data the $L2L_{trace}$, $L2L_{2gram}$ and

$L2L_{case}$ exhibit an increase of about 7 %, 18 % and 31 % against DCIc and, more prominently, of around 81 %, 72 % and 66 % against E-Max, respectively (see Figs. 30(a), 30(d) and 30(f)).

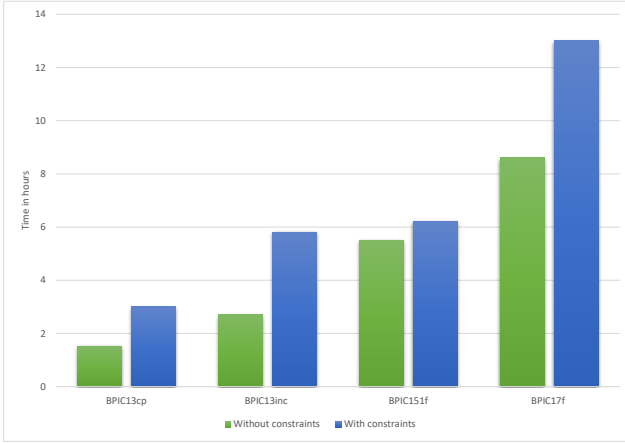


Figure 23: Impact of using data constraints with real-world logs on execution time

An analogous improvement can be observed with the log-to-log time deviation measures, as Fig. 31 highlights. EC-SA-Data yields an average reduction of SMAPE_{ET} and SMAPE_{CT} by around 3 % and 8 % against DCIc and 47 % and 50 % against E-Max, respectively.

Using the data constraints in addition to the process knowledge is the key feature that allows EC-SA-Data to outperform DCIc and E-Max. However, E-Max is the fastest one between the three correlation techniques, as shown in Fig. 32.

6.3. Discussion

Based on the different sensitivity analyses we conducted, we found that using constraints improves the correlation accuracy (see Section 6.2.3). The comparative evaluation reported on in Section 6.2.6 corroborates this statement as EC-SA-Data outperforms state-of-the-art techniques. Remarkably, the use of data constraints mitigates the effect of flawed process models, as Section 6.2.5 highlights. On the other hand, the experimental results discussed in Section 6.2.4 demonstrate that verified rules and a reliable control-flow model diminish the negative effect of rules that are partially incorrect. This dual contribution makes the techniques robust with respect to the presence of noise in the event log too, as rule support and model fitness can equally measure the extent to which the event log is faulty with respect to normative specifications [75, 41].

The accuracy of our approach tends to be partially prone to a worsening when the cases' density and log size increase, as highlighted in Section 6.2.1, because these parameters increase the number of options available for correlation. EC-SA-Data can handle cyclic, exclusive, and parallel behaviors, largely present in the real-world event logs we analyzed, as opposed to other proposed techniques [15, 16, 26]. However, based on the experimental results presented in Section 6.2.2, we notice that EC-SA-Data is sensitive to duplicated activities, which may lead to incorrect assignments. Overcoming these limitations paves the path for future work,

which we discuss in the next section alongside the concluding remarks.

7. Conclusion

The research presented in this paper addresses the event correlation problem. To automatically correlate the events with their cases, our approach (EC-SA-Data) resorts to data constraints to model domain knowledge, in addition to process models that define the control flow of the original process. Our approach uses multi-level objective simulated annealing to map every event to a case. We use trace alignment cost, support of data constraints, and activity execution time variance for optimization. Our evaluation on real-world event logs demonstrates that using data constraints as input in addition to the process model improves the generated log quality positively.

These findings suggest numerous directions for future research. A first objective to pursue is the improvement of the technique by making it more robust in presence of duplicated activities and more scalable, especially with respect to inter-arrival rate and log size. We are also interested in the extension of the technique towards new features. A first option is to support a broader spectrum of data constraints, such as the inter-case rules [76]. At present, EC-SA-Data employs the process model and the data rules at different stages in a multi-level scheme. Integrating approaches that tackle the two perspectives in an ensemble, such as [77], is an intriguing alternative that is worth investigating. Also, extending the set of quality measures for correlated event logs, encompassing both the perspectives of data rules and control-flow model [78, 79] is an objective that can be pursued in future endeavors. Other opportunities for investigations in the field are the analysis of event logs wherein multiple events may register the execution of an activity, to record the stage reached in their life cycle [80], or one event reports on the unfolding of multiple cases, as in the case of batch processes [81]. Finally, an interesting avenue for upcoming work is to explore the possibility of correlating the events solely based on constraints, without prior knowledge of the whole process model and thus by means of declarative process rules that specify its behavior [82].

Acknowledgements

The work of Claudio Di Ciccio was partly supported by the Italian Ministry of University and Research (MUR) under the PRIN programme, grant B87G22000450001 (PIN-POINT), and grant "Dipartimenti di eccellenza 2018-2022" of the Department of Computer Science at Sapienza University of Rome. The research by Jan Mendling was supported by the Einstein Foundation Berlin.

A. BPIC-2017 rules

Table 15 reports the data constraints we retrieved from visual inspection of the BPIC17_f event log to run our experiments as described in Sections 6.1 and 6.2.4.

Event-Case Correlation for Process Mining using Probabilistic Optimization

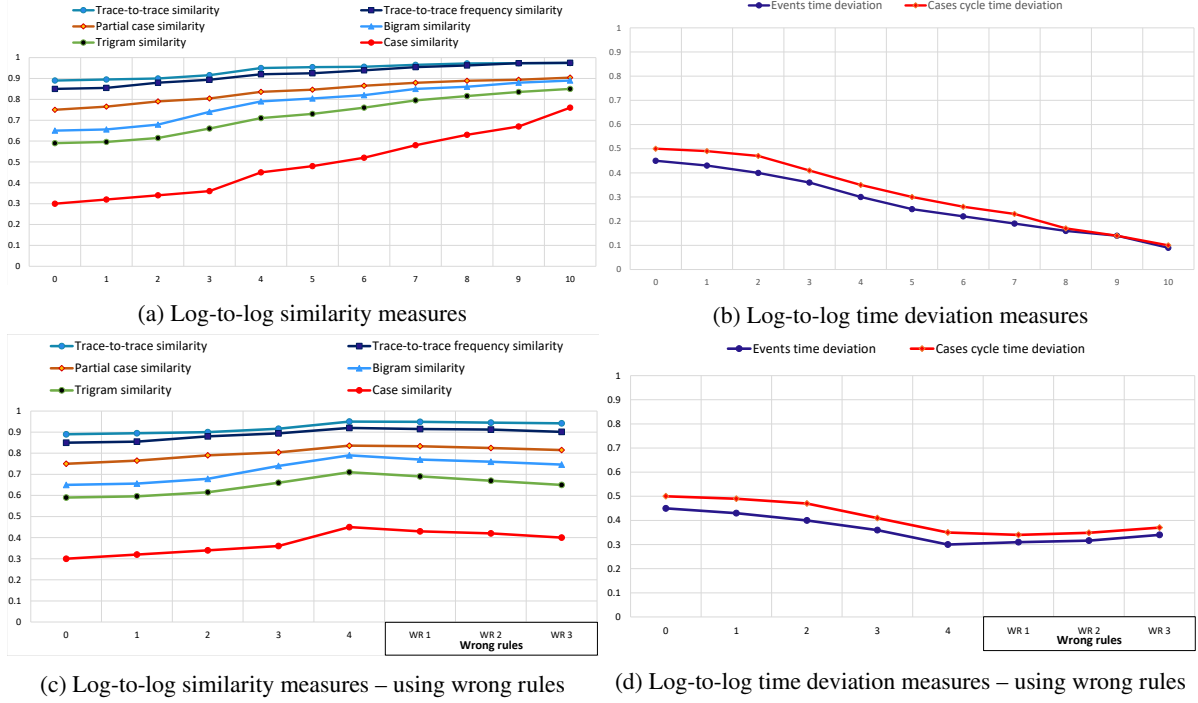


Figure 24: Impact of the quality and number of constraints on accuracy measures

Table 15: Observed rules in the BPIC17_f event log

C_1 : $\sigma(i-1).ApplicationType = \sigma(i).ApplicationType$
C_2 : $\sigma(i-1).LoanGoal = \sigma(i).LoanGoal$
C_3 : $\sigma(i-1).RequestedAmount = \sigma(i).RequestedAmount$
C_4 : IF $\sigma(i).EventOrigin = \text{"Offer"} \wedge \sigma(i).Act \neq \text{"O_Create offer"} \wedge \sigma(j).Act = \text{"O_Create offer"}$ THEN $\sigma(i).OfferID = \sigma(j).EventId$
C_5 : IF $\sigma(i).Act = \text{"A_Complete"} \wedge \sigma(j).Act = \text{"W_Call after offers"}$ THEN $\sigma(i).Resource = \sigma(j).Resource$
C_6 : IF $\sigma(i).Act = \text{"W_Call after offers"} \wedge \sigma(i).Act = \text{"O_Sent (mail and online)"}$ THEN $\sigma(i).Resource = \sigma(j).Resource$
C_7 : IF $\sigma(i).Act = \text{"O_Returned"} \wedge \sigma(i).Act = \text{"A_Validating"}$ THEN $\sigma(i).Resource = \sigma(j).Resource$
C_8 : IF $\sigma(i).Act = \text{"A_Pending"} \wedge \sigma(i).Act = \text{"O_Accepted (mail and online)"}$ THEN $\sigma(i).Resource = \sigma(j).Resource$
C_9 : IF $\sigma(i).Act = \text{"O_Refused"} \wedge \sigma(i).Act = \text{"A_Denied"}$ THEN $\sigma(i).Resource = \sigma(j).Resource$
C_{10} : $\sigma(i).Act = \text{"O_Create offer"} \wedge \sigma(i).Act = \text{"A_Accepted"}$ THEN $\sigma(i).Resource = \sigma(j).Resource$

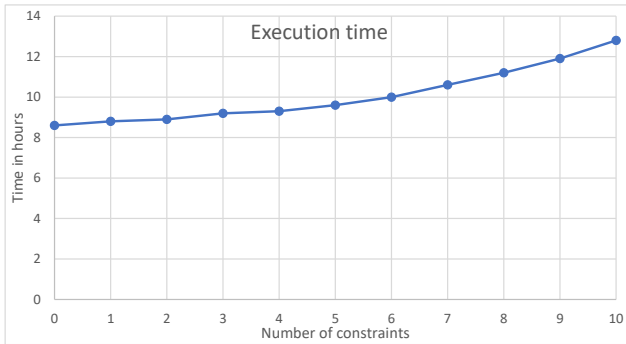


Figure 25: Impact of increasing the data constraints on execution time

B. Sample of the event-time constraints used in the sixth experiment

Table 16 reports the event-time constraints we extracted by computing the quartiles that represent the activity execution time ranges of the BPIC13_{cp} event log. We used these values to run our experiments as described in Sections 6.1 and 6.2.6.

Table 16: Event-time constraints of the BPIC13_{cp} event log

C_1 : IF $\sigma(i).Act = \text{"Accepted"}$ THEN $275 \leq (\sigma(i).Ts - \sigma(i-1).Ts) \leq 13583261$
C_2 : IF $\sigma(i).Act = \text{"Completed"}$ THEN $120 \leq (\sigma(i).Ts - \sigma(i-1).Ts) \leq 4588415$
C_3 : IF $\sigma(i).Act = \text{"Queued"}$ THEN $151 \leq (\sigma(i).Ts - \sigma(i-1).Ts) \leq 75314$
C_4 : IF $\sigma(i).Act = \text{"Unmatched"}$ THEN $729 \leq (\sigma(i).Ts - \sigma(i-1).Ts) \leq 622157$

Event-Case Correlation for Process Mining using Probabilistic Optimization

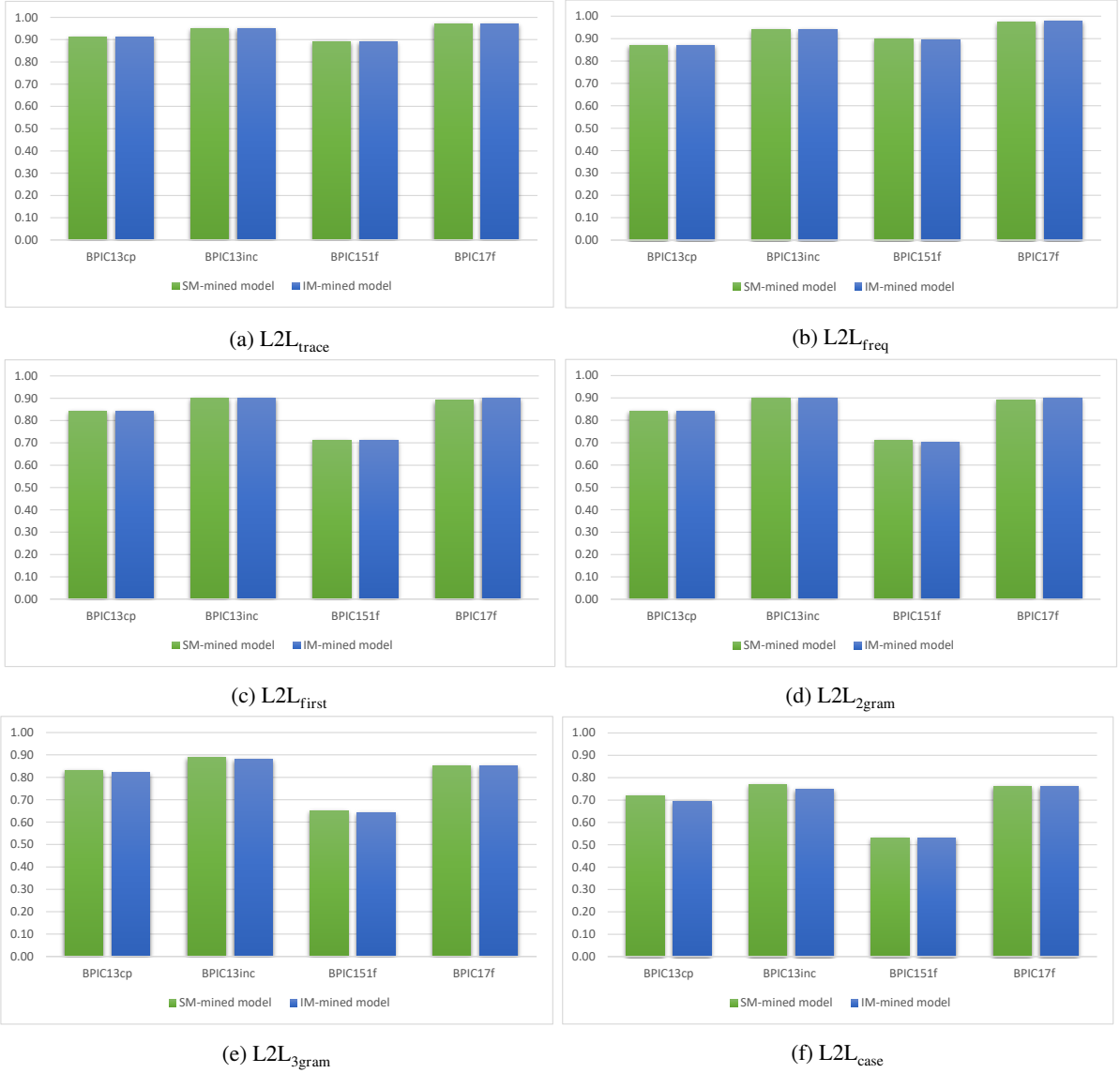


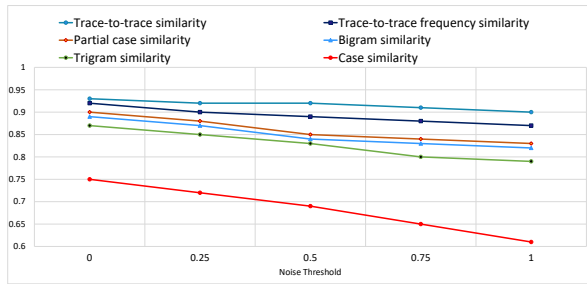
Figure 26: Impact of using models with different fitness and precision on log-to-log similarity measures



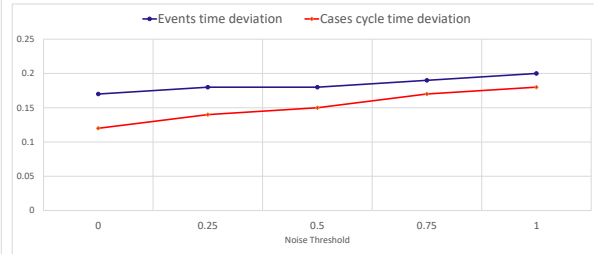
Figure 27: Impact of using models with different fitness and precision on log-to-log time deviation measures

References

- [1] S.-M.-R. Beheshti, B. Benatallah, H. R. Motahari-Nezhad, Scalable graph-based olap analytics over process execution data, Distributed and Parallel Databases 34 (2016) 379–423.
- [2] B. Benatallah, S. Sakr, D. Grigori, H. R. Motahari-Nezhad, M. C. Barukh, A. Gater, S. H. Ryu, et al., Process analytics: concepts and techniques for querying and analyzing process data, Springer, 2016.



(a) Log-to-log similarity measures



(b) Log-to-log time deviation measures

Figure 28: Sensitivity analysis on the impact that the use of models mined by the inductive miner [74] with different noise thresholds has on the accuracy measures, taking the BPIC13_{cp} log as input

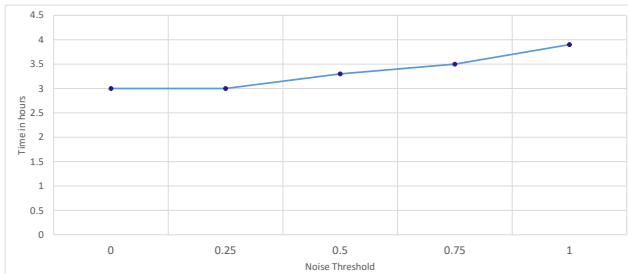


Figure 29: Sensitivity analysis on the impact that the use of models mined by the inductive miner [74] with different noise thresholds has on the execution time, taking the BPIC13_{cp} log as input

- [3] P. Soffer, A. Hinze, A. Koschmider, H. Ziekow, et al., From event streams to process models and back: Challenges and opportunities, *Inf. Syst.* 81 (2019) 181–200.
- [4] W. M. P. van der Aalst, A. Adriansyah, A. K. A. de Medeiros, F. Arcieri, T. Baier, T. Blickle, R. P. J. C. Bose, P. van den Brand, R. Brandtjen, J. C. A. M. Buijs, A. Burattin, J. Carmona, M. Castellanos, J. Claes, J. Cook, N. Costantini, F. Curbra, E. Damiani, M. de Leoni, P. Delias, B. F. van Dongen, M. Dumas, S. Dustdar, D. Fahland, D. R. Ferreira, W. Gaaloul, F. van Geffen, S. Goel, C. W. Günther, A. Guzzo, P. Harmon, A. H. M. ter Hofstede, J. Hoogland, J. E. Ingvaldsen, K. Kato, R. Kuhn, A. Kumar, M. L. Rosa, F. M. Maggi, D. Malerba, R. S. Mans, A. Manuel, M. McCreesh, P. Mello, J. Mendling, M. Montali, H. R. M. Nezhad, M. zur Muehlen, J. Munoz-Gama, L. Pontieri, J. Ribeiro, A. Rozinat, H. S. Pérez, R. S. Pérez, M. Sepúlveda, J. Sinur, P. Soffer, M. Song, A. Sperduti, G. Stilo, C. Stoel, K. D. Swenson, M. Talamo, W. Tan, C. Turner, J. Vanthienen, G. Varvaressos, E. Verbeek, M. Verdonk, R. Vigo, J. Wang, B. Weber, M. Weidlich, T. Weijters, L. Wen, M. Westergaard, M. T. Wynn, Process mining manifesto, in: F. Daniel, K. Barkaoui, S. Dustdar (Eds.), *Business Process Management Workshops - BPM 2011 International Workshops, Clermont-Ferrand, France, August 29, 2011, Revised Selected Papers, Part I*, volume 99 of *Lecture Notes in Business Information Processing*, Springer, 2011, pp. 169–194. URL: https://doi.org/10.1007/978-3-642-28108-2_19. doi:10.1007/978-3-642-28108-2_19.
- [5] W. van der Aalst, *Process Mining – Data science in action*, 2nd Edition, Springer, 2016.
- [6] Q. Guo, L. Wen, J. Wang, Z. Yan, S. Y. Philip, Mining invisible tasks in non-free-choice constructs, in: *International Conference on Business Process Management*, Springer, 2015, pp. 109–125.
- [7] S. J. J. Leemans, D. Fahland, W. M. P. van der Aalst, Discovering block-structured process models from event logs containing infrequent behaviour, in: N. Lohmann, M. Song, P. Wohed (Eds.), *Business*

Process Management Workshops - BPM 2013 International Workshops, Beijing, China, August 26, 2013, Revised Papers, volume 171 of *Lecture Notes in Business Information Processing*, Springer, 2013, pp. 66–78. URL: https://doi.org/10.1007/978-3-319-06257-0_6. doi:10.1007/978-3-319-06257-0_6.

- [8] J. C. A. M. Buijs, B. F. van Dongen, W. M. P. van der Aalst, On the role of fitness, precision, generalization and simplicity in process discovery, in: R. Meersman, H. Panetto, T. S. Dillon, S. Rinderle-Ma, P. Dadam, X. Zhou, S. Pearson, A. Ferscha, S. Bergamaschi, I. F. Cruz (Eds.), *On the Move to Meaningful Internet Systems: OTM 2012, Confederated International Conferences: CoopIS, DOA-SVI, and ODBASE 2012, Rome, Italy, September 10-14, 2012. Proceedings, Part I*, volume 7565 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 305–322. URL: https://doi.org/10.1007/978-3-642-33606-5_19. doi:10.1007/978-3-642-33606-5_19.
- [9] S. K. vanden Broucke, J. De Weerd, Fodina: a robust and flexible heuristic process discovery technique, *Decision Support Systems* (2017).
- [10] A. Augusto, R. Conforti, M. Dumas, M. La Rosa, G. Bruno, Automated discovery of structured process models from event logs: The discover-and-structure approach, *Data & Knowledge Engineering* 117 (2018) 373–392.
- [11] A. Augusto, R. Conforti, M. Dumas, M. La Rosa, A. Polyvyanyy, Split Miner: Automated discovery of accurate and simple business process models from event logs, *Knowledge and Information Systems* (2018).
- [12] S. J. van Zelst, B. F. van Dongen, W. M. P. van der Aalst, ILP-Based Process Discovery Using Hybrid Regions, in: *International Workshop on Algorithms & Theories for the Analysis of Event Data, ATAED 2015*, volume 1371 of *CEUR Workshop Proceedings*, CEUR-WS.org, 2015, pp. 47–61.
- [13] S. Bala, J. Mendling, M. Schimak, P. Queteschiner, Case and activity identification for mining process models from middleware, in: R. A. Buchmann, D. Karagiannis, M. Kirikova (Eds.), *The Practice of Enterprise Modeling - 11th IFIP WG 8.1. Working Conference, PoEM 2018, Vienna, Austria, October 31 - November 2, 2018, Proceedings*, volume 335 of *Lecture Notes in Business Information Processing*, Springer, 2018, pp. 86–102. URL: https://doi.org/10.1007/978-3-030-02302-7_6. doi:10.1007/978-3-030-02302-7_6.
- [14] G. Meroni, C. D. Ciccio, J. Mendling, An artifact-driven approach to monitor business processes through real-world objects, in: E. M. Maximilien, A. Vallecillo, J. Wang, M. Oriol (Eds.), *Service-Oriented Computing - 15th International Conference, ICSOC 2017, Malaga, Spain, November 13-16, 2017, Proceedings*, volume 10601 of *Lecture Notes in Computer Science*, Springer, 2017, pp. 297–313. URL: https://doi.org/10.1007/978-3-319-69035-3_21. doi:10.1007/978-3-319-69035-3_21.
- [15] S. Pourmirza, R. Dijkman, P. Grefen, Correlation Miner: Mining business process models and event correlations without case identifiers, *IJICIS* 26 (2017).
- [16] D. R. Ferreira, D. Gillblad, Discovering process models from unlabelled event logs, in: U. Dayal, J. Eder, J. Koehler, H. A. Reijers

Event-Case Correlation for Process Mining using Probabilistic Optimization



Figure 30: Log-to-log similarity measures: Comparing EC-SA-Data, DCIc, and E-Max

- (Eds.), Business Process Management, 7th International Conference, BPM 2009, Ulm, Germany, September 8-10, 2009. Proceedings, volume 5701 of *Lecture Notes in Computer Science*, Springer, 2009, pp. 143–158. URL: https://doi.org/10.1007/978-3-642-03848-8_11. doi:10.1007/978-3-642-03848-8_11.
- [17] D. Bayomie, A. Awad, E. Ezat, Correlating unlabeled events from cyclic business processes execution, in: S. Nurcan, P. Soffer, M. Bajec, J. Eder (Eds.), *Advanced Information Systems Engineering - 28th International Conference, CAiSE 2016, Ljubljana, Slovenia, June 13-17, 2016. Proceedings*, volume 9694 of *Lecture Notes in Computer*

- Science*, Springer, 2016, pp. 274–289. URL: https://doi.org/10.1007/978-3-319-39696-5_17. doi:10.1007/978-3-319-39696-5_17.
- [18] D. Bayomie, C. D. Ciccio, M. L. Rosa, J. Mendling, A probabilistic approach to event-case correlation for process mining, in: A. H. F. Laender, B. Pernici, E. Lim, J. P. M. de Oliveira (Eds.), *Conceptual Modeling - 38th International Conference, ER 2019, Salvador, Brazil, November 4-7, 2019. Proceedings*, volume 11788 of *Lecture Notes in Computer Science*, Springer, 2019, pp. 136–152. URL: https://doi.org/10.1007/978-3-030-33223-5_12. doi:10.1007/978-3-030-33223-5_12.

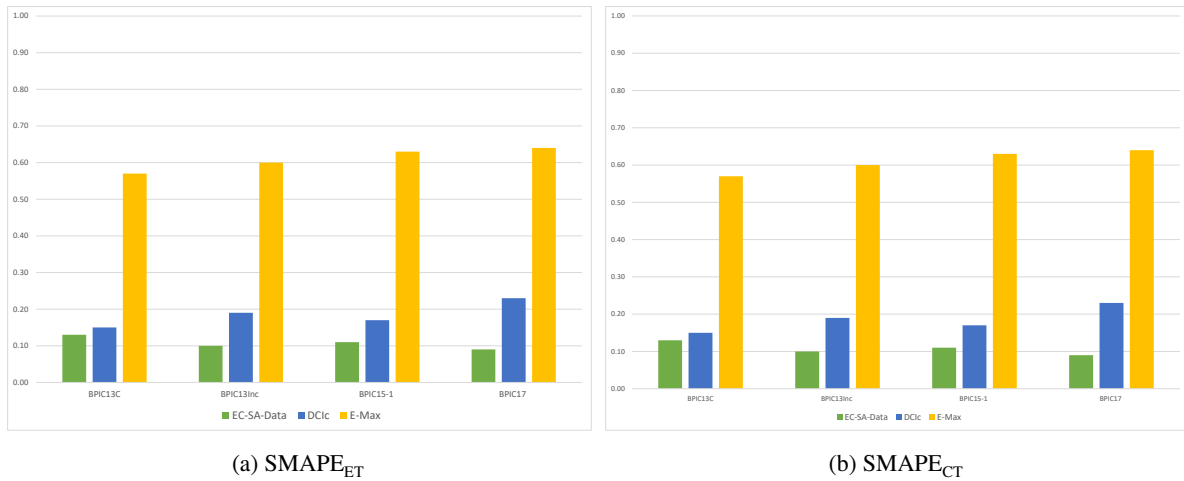


Figure 31: Log-to-log time deviation measures: Comparing EC-SA-Data, DCIc, and E-Max

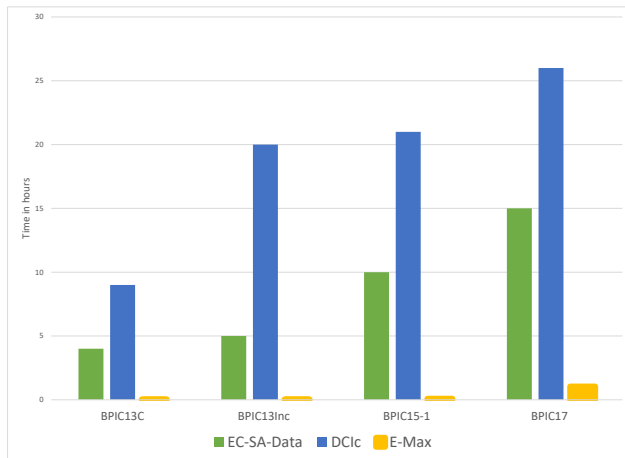


Figure 32: Execution time: Comparing EC-SA-Data, DCIc [17] and E-Max[16]

- [19] B. F. van Dongen, J. De Smedt, C. Di Ciccio, J. Mendling, Conformance checking of mixed-paradigm process models, *Information Systems* (2020) 101685.
- [20] C. S. Calude, G. Longo, The deluge of spurious correlations in big data, *Foundations of Science* 22 (2017) 595–612.
- [21] A. L'heureux, K. Grolinger, H. F. Elyamany, M. A. Capretz, Machine learning with big data: Challenges and approaches, *Ieee Access* 5 (2017) 7776–7797.
- [22] E. Achtert, C. Böhm, J. David, P. Kröger, A. Zimek, Global correlation clustering based on the hough transform, *Statistical Analysis and Data Mining: The ASA Data Science Journal* 1 (2008) 111–127.
- [23] C. Jermaine, Finding the most interesting correlations in a database: how hard can it be?, *Information Systems* 30 (2005) 21–46.
- [24] P. G. Brown, P. J. Haas, - bhunt: Automatic discovery of fuzzy algebraic constraints in relational data, in: J.-C. Freytag, P. Lockemann, S. Abiteboul, M. Carey, P. Selinger, A. Heuer (Eds.), *Proceedings 2003 VLDB Conference*, Morgan Kaufmann, San Francisco, 2003, pp. 668–679. URL: <https://www.sciencedirect.com/science/article/pii/B9780127224428500653>. doi:<https://doi.org/10.1016/B978-012722442-8/50065-3>.
- [25] K. Diba, K. Batoulis, M. Weidlich, M. Weske, Extraction, correlation, and abstraction of event data for process mining, *Wiley Interdisciplinary Reviews: Data Mining and Knowledge Discovery* (2019) 1–24.
- [26] M. Walicki, D. Ferreira, Sequence partitioning for process mining with unlabeled event logs, *DKE* 70 (2011).
- [27] S. Pourmirza, R. Dijkman, P. Grefen, Correlation mining: Mining process orchestrations without case identifiers, in: A. Barros, D. Grigori, N. C. Narendra, H. K. Dam (Eds.), *Service-Oriented Computing*, Springer Berlin Heidelberg, Berlin, Heidelberg, 2015, pp. 237–252.
- [28] D. Bayomie, I. M. A. Helal, A. Awad, E. Ezat, A. ElBastawissi, Deducing case ids for unlabeled event logs, in: M. Reichert, H. A. Reijers (Eds.), *Business Process Management Workshops*, Springer International Publishing, Cham, 2016, pp. 242–254.
- [29] H. Nezhad, R. Saint-Paul, F. Casati, B. Benatallah, Event correlation for process discovery from web service interaction logs, *VLDB J.* 20 (2011).
- [30] R. Engel, W. Krathu, M. Zapletal, C. Pichler, R. P. J. C. Bose, W. van der Aalst, H. Werthner, C. Huemer, Analyzing inter-organizational business processes, *Information Systems and e-Business Management* 2015 14:3 14 (2015) 577–612.
- [31] H. Reguieg, F. Toumani, H. R. M. Nezhad, B. Benatallah, Using mapreduce to scale events correlation discovery for business processes mining, in: A. Barros, A. Gal, E. Kindler (Eds.), *Business Process Management - 10th International Conference, BPM 2012, Tallinn, Estonia, September 3-6, 2012. Proceedings*, volume 7481 of *Lecture Notes in Computer Science*, Springer, 2012, pp. 279–284. URL: https://doi.org/10.1007/978-3-642-32885-5_22. doi:10.1007/978-3-642-32885-5_22.
- [32] H. Reguieg, B. Benatallah, H. R. Nezhad, F. Toumani, Event correlation analytics: Scaling process mining using mapreduce-aware event correlation discovery techniques, *IEEE Transactions on Services Computing* 8 (2015) 847–860.
- [33] L. Cheng, B. F. V. Dongen, W. M. V. D. Aalst, Efficient event correlation over distributed systems, *Proceedings - 2017 17th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing, CCGRID 2017* (2017) 1–10.
- [34] E. G. L. de Murillas, W. M. P. van der Aalst, H. A. Reijers, Process mining on databases: Unearthing historical data from redo logs, in: H. R. Motahari-Nezhad, J. Recker, M. Weidlich (Eds.), *Business Process Management*, Springer International Publishing, Cham, 2015, pp. 367–385.
- [35] E. G. L. de Murillas, H. A. Reijers, W. M. van der Aalst, Case notion discovery and recommendation: automated event log building on databases, *Knowledge and Information Systems* (2019).
- [36] A. Djedović, A. Karabegović, E. Žunić, D. Alić, A rule based events correlation algorithm for process mining, in: *IAT*, Springer, Cham, 2020, pp. 587–605.
- [37] A. Abbad Andaloussi, A. Burattin, B. Weber, Toward an Automated Labeling of Event Log Attributes, in: *EMMSAD/BPMDS*, Springer,

- 2018, pp. 82–96.
- [38] A. Burattin, R. Vigo, A framework for semi-automated process instance discovery from decorative attributes, in: 2011 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), 2011, pp. 176–183. doi:10.1109/CIDM.2011.5949450.
- [39] W. M. P. van der Aalst, The application of petri nets to workflow management, *Journal of Circuits, Systems, and Computers* 8 (1998) 21–66.
- [40] A. Adriansyah, J. Munoz-Gama, J. Carmona, B. F. van Dongen, W. M. P. van der Aalst, Measuring precision of modeled behavior, *Inf. Syst. E-Business Management* 13 (2015) 37–67. One-alignment and best-alignment precision.
- [41] J. Carmona, B. F. van Dongen, A. Solti, M. Weidlich, *Conformance Checking - Relating Processes and Models*, Springer, 2018. doi:10.1007/978-3-319-99414-7.
- [42] S. Kirkpatrick, C. D. Gelatt, M. Vecchi, Optimization by simulated annealing, *Science* 220 (1983) 671–680.
- [43] A. Eiben, J. Smith, *Introduction to evolutionary computing*, volume 2, 2004.
- [44] G. W. Stewart, Stochastic perturbation theory, <http://dx.doi.org/10.1137/103212132> 32 (2006) 579–610.
- [45] D. Henderson, S. H. Jacobson, A. W. Johnson, The theory and practice of simulated annealing, in: F. W. Glover, G. A. Kochenberger (Eds.), *Handbook of Metaheuristics*, volume 57 of *International Series in Operations Research & Management Science*, Kluwer / Springer, 2003, pp. 287–319. URL: https://doi.org/10.1007/0-306-48056-5_10. doi:10.1007/0-306-48056-5_10.
- [46] R. Kolisch, S. Hartmann, Experimental investigation of heuristics for resource-constrained project scheduling: An update, *European Journal of Operational Research* 174 (2006) 23–37.
- [47] K. Bouleimen, H. Lecocq, A new efficient simulated annealing algorithm for the resource-constrained project scheduling problem and its multiple mode version, *Eur. J. Oper. Res.* 149 (2003) 268–281.
- [48] M. A. Arostegui, S. N. Kadipasaoglu, B. M. Khumawala, An empirical comparison of tabu search, simulated annealing, and genetic algorithms for facilities location problems, *International Journal of Production Economics* 103 (2006) 742–754.
- [49] L. Angelis, E. Bora-Senta, C. Moysiadiis, Optimal exact experimental designs with correlated errors through a simulated annealing algorithm, *Computational Statistics & Data Analysis* 37 (2001) 275–296.
- [50] A. Askarzadeh, L. dos Santos Coelho, C. E. Klein, V. C. Mariani, A population-based simulated annealing algorithm for global optimization, in: 2016 IEEE International Conference on Systems, Man, and Cybernetics, SMC 2016, Budapest, Hungary, October 9–12, 2016, IEEE, 2016, pp. 4626–4633. URL: <https://doi.org/10.1109/SMC.2016.7844961>. doi:10.1109/SMC.2016.7844961.
- [51] Y. Nourani, B. Andresen, A comparison of simulated annealing cooling strategies, *Journal of Physics A: Mathematical and General* 31 (1998) 8373.
- [52] G. J. Pace, *Mathematics of Discrete Structures for Computer Science*, Springer, 2012. doi:10.1007/978-3-642-29840-0.
- [53] F. M. Maggi, R. P. J. C. Bose, W. M. P. van der Aalst, Efficient discovery of understandable declarative process models from event logs, in: *Advanced Information Systems Engineering - 24th International Conference, CAiSE 2012*, Gdansk, Poland, June 25–29, 2012. Proceedings, 2012, pp. 270–285. doi:10.1007/978-3-642-31095-9_18.
- [54] C. Di Ciccio, F. M. Maggi, M. Montali, J. Mendling, On the relevance of a business constraint to an event log, *Inf. Syst.* 78 (2018) 144–161.
- [55] A. Ceconi, C. Di Ciccio, G. De Giacomo, J. Mendling, Interestingness of traces in declarative process mining: The janus ltlp approach, in: *Business Process Management - 16th International Conference, BPM 2018*, Sydney, NSW, Australia, September 9–14, 2018, Proceedings, Springer, 2018, pp. 121–138. doi:10.1007/978-3-319-98648-7_8.
- [56] A. Adriansyah, B. F. van Dongen, W. M. P. van der Aalst, Conformance checking using cost-based fitness analysis, in: *Proceedings of the 15th IEEE International Enterprise Distributed Object Computing Conference, EDOC 2011*, Helsinki, Finland, August 29 - September 2, 2011, IEEE Computer Society, 2011, pp. 55–64. URL: <https://doi.org/10.1109/EDOC.2011.12>. doi:10.1109/EDOC.2011.12.
- [57] J. E. Gentle, *Computational statistics* (2009).
- [58] S. Geman, D. Geman, Stochastic relaxation, gibbs distributions, and the bayesian restoration of images, *IEEE Transactions on Pattern Analysis and Machine Intelligence PAMI-6* (1984) 721–741.
- [59] G. Navarro, A guided tour to approximate string matching, *ACM Comput. Surv.* 33 (2001).
- [60] E. Deza, M. Deza, *Dictionary of distances*, North-Holland, 2006.
- [61] J. L. Devore, K. N. Berk, *Modern mathematical statistics with applications* (2012).
- [62] R. Jonker, T. Volgenant, Improving the hungarian assignment algorithm, *Operations Research Letters* 5 (1986) 171–175.
- [63] A. Van Looy, A. Shafagatova, Business process performance measurement: a structured literature review of indicators, measures and metrics, *SpringerPlus* 5 (2016) 1797.
- [64] M. Dumas, M. L. Rosa, J. Mendling, H. A. Reijers, *Fundamentals of Business Process Management*, Second Edition, Springer, 2018.
- [65] J. Mendling, B. Depaire, H. Leopold, Theory and practice of algorithm engineering, *CoRR abs/2107.10675* (2021).
- [66] A. Maaradji, M. Dumas, M. La Rosa, A. Ostovar, Fast and accurate business process drift detection, in: *LNCS*, volume 9253, Springer, 2015, pp. 406–422.
- [67] B. Weber, M. Reichert, S. Rinderle-Ma, Change patterns and change support features - enhancing flexibility in process-aware information systems, *Data Knowl. Eng.* 66 (2008) 438–466.
- [68] W. Steeman, BPI Challenge 2013, closed problems, 2013. URL: https://data.4tu.nl/articles/dataset/BPI_Challenge_2013_closed_problems/12714476. doi:10.4121/uuid:c2c3b154-ab26-4b31-a0e8-8f2350ddac11.
- [69] W. Steeman, BPI Challenge 2013, incidents, 2013. URL: https://data.4tu.nl/articles/dataset/BPI_Challenge_2013_incidents/12693914. doi:10.4121/uuid:500573e6-acc6-4b0c-9576-aa5468b10cee.
- [70] B. van Dongen, BPI Challenge 2015 Municipality 1, 2015. URL: https://data.4tu.nl/articles/dataset/BPI_Challenge_2015_Municipality_1/12709154. doi:10.4121/uuid:a0addfda-2044-4541-a450-fdc9fe16d17.
- [71] B. van Dongen, BPI Challenge 2017 - Offer log, 2021. URL: https://data.4tu.nl/articles/dataset/BPI_Challenge_2017_-_Offer_log/12705737. doi:10.4121/12705737.v2.
- [72] A. Augusto, R. Conforti, M. Dumas, M. La Rosa, F. Maggi, A. Marrella, M. Mecella, A. Soo, Automated discovery of process models from event logs: Review and benchmark, *IEEE TKDE* 31 (2019).
- [73] A. Augusto, J. Mendling, M. Vidgof, B. Wurm, The connection between process complexity of event sequences and models discovered by process mining, *Inf. Sci.* 598 (2022) 196–215.
- [74] S. Leemans, D. Fahland, W. van der Aalst, Discovering block-structured process models from event logs containing infrequent behaviour, in: *Proc. of BPM Workshops*, Springer, 2014.
- [75] A. Rogge-Solti, A. Senderovich, M. Weidlich, J. Mendling, A. Gal, In log and model we trust? A generalized conformance checking framework, in: *BPM, Springer*, 2016, pp. 179–196.
- [76] K. Winter, F. Stertz, S. Rinderle-Ma, Discovering instance and process spanning constraints from process execution logs, *Inf. Syst.* 89 (2020) 101484.
- [77] F. Mannhardt, M. de Leoni, H. A. Reijers, W. M. P. van der Aalst, Balanced multi-perspective checking of process conformance, *Computing* 98 (2016) 407–437.
- [78] B. F. van Dongen, J. De Smedt, C. Di Ciccio, J. Mendling, Conformance checking of mixed-paradigm process models, *Information Systems* 102 (2021) 101685.
- [79] P. Felli, A. Gianola, M. Montali, A. Rivkin, S. Winkler, Cocomot: Conformance checking of multi-perspective processes via SMT, in: *Business Process Management - 19th International Conference, BPM 2021*, Rome, Italy, September 06–10, 2021, Proceedings, volume 12875 of *Lecture Notes in Computer Science*, Springer, 2021, pp. 217–234. doi:10.1007/978-3-030-85469-0_15.
- [80] T. Baier, C. Di Ciccio, J. Mendling, M. Weske, Matching events and activities by integrating behavioral aspects and label analysis, *SoSyM* 17 (2018) 573–598.

- [81] N. Martin, L. Pufahl, F. Mannhardt, Detection of batch activities from event logs, *Inf. Syst.* 95 (2021) 101642.
- [82] C. Di Ciccio, M. Montali, Declarative process specifications: Reasoning, discovery, monitoring, in: W. M. P. van der Aalst, J. Carmona (Eds.), *Process Mining Handbook*, volume 448 of *Lecture Notes in Business Information Processing*, Springer, 2022, pp. 108–152. doi:10.1007/978-3-031-08848-3_4.

This document is a pre-print copy of the manuscript
([Bayomie, Di Ciccio, and Mendling 2023](#))
published by Elsevier.

The final version of the paper is identified by DOI: [10.1016/j.is.2023.102167](#)

References

Bayomie, Dina, Claudio Di Ciccio, and Jan Mendling (2023). “Event-case correlation for process mining using probabilistic optimization”. In: *Information Systems* 114, p. 102167. ISSN: 0306-4379. DOI: [10.1016/j.is.2023.102167](#).

BibTeX

```
@Article{
  author      = {Bayomie, Dina and Di Ciccio, Claudio and Mendling, Jan},
  journal     = {Information Systems},
  title       = {Event-case correlation for process mining using
    probabilistic optimization},
  year        = {2023},
  issn        = {0306-4379},
  pages       = {102167},
  volume      = {114},
  doi         = {10.1016/j.is.2023.102167},
  keywords    = {Process mining; Event correlation; Simulated annealing;
    Constraints; Association rules},
  publisher   = {Elsevier}
}
```