Progress in Artificial Intelligence
https://doi.org/10.1007/s13748-022-00282-6

REGULAR PAPER

Feature recommendation for structural equation model discovery in process mining

Mahnaz Sadat Qafari[1] · **Wil M. P. van der Aalst**[1]

Received: 13 March 2021 / Accepted: 3 May 2022

Abstract
Process mining techniques can help organizations to improve their operational processes. Organizations can benefit from process mining techniques in finding and amending the root causes of performance or compliance problems. Considering the volume of the data and the number of features captured by the information system of today's companies, the task of discovering the set of features that should be considered in causal analysis can be quite involving. In this paper, we propose a method for finding the set of (aggregated) features with a possible causal effect on the problem. The causal analysis task is usually done by applying a machine learning technique to the data gathered from the information system supporting the processes. To prevent mixing up correlation and causation, which may happen because of interpreting the findings of machine learning techniques as causal, we propose a method for discovering the structural equation model of the process that can be used for causal analysis. We have implemented the proposed method as a plugin in ProM, and we have evaluated it using real and synthetic event logs. These experiments show the validity and effectiveness of the proposed methods.

Keywords Process mining · Causality inference · Root cause analysis

1 Introduction

Organizations aim to improve operational processes to serve customers better and to become more profitable. To this goal, they can utilize process mining techniques in many steps, including identifying friction points in the process, finding what causes a friction point, estimating the possible impact of changing each factor on the process performance, and also planning process enhancement actions. Finding the set of features that cause (effect) another feature, i.e., the set of features that change the value of any of the former ones can change the value of the latter one, is the core of the process improvement process. Today, there are several robust techniques for process monitoring and finding their friction points, but little work on *causal analysis*. In this paper, we focus on causal analysis and investigating the impact of interventions.

Processes are complicated entities involving many steps, where each step itself may include many influential factors and features. Moreover, not just the steps but also the order of the steps that are taken for each process instance may vary, which results in several process instance variants. This makes it quite hard to identify the set of features that influence a problem. However, in the literature related to causal analysis in the field of process mining, it is usually assumed that the user provides the set of features that have a causal relationship with the observed problem in the process (see for example [1,2]). To overcome this issue, we investigate the application of feature selection methods to find the set of features that may have a causal relationship with the problem. Moreover, we use a method based on information gain to identify the values of these features that are more prone to causing the problem.

Traditionally, the task of finding the root cause of a problem in a process is done in two steps; first gathering process data, and then applying data mining and machine learning techniques. It is easy to find correlations, but very hard to determine *causation*. Although the goal is to perform causal analysis, a naive application of such approaches often leads to a mix-up of correlation and causation. Consequently,

✉ Mahnaz Sadat Qafari
m.s.qafari@pads.rwth-aachen.de

Wil M. P. van der Aalst
wvdaalst@pads.rwth-aachen.de

[1] Process and Data Science Chair (PADS), RWTH Aachen University, 52074 Aachen, North Rhine-Westphalia, Germany

Published online: 25 June 2022

process enhancement based on the results of such approaches does not always lead to any process improvements.

Consider the following three scenarios:

(i) In an online shop, it has been observed that in many delayed deliveries certain employees were responsible.
(ii) In a consultancy company, there are deviations in those cases done by the most experienced employees.
(iii) In an IT company, it has been observed that the higher the number of resources assigned to a project, the longer it takes.

The following possibly incorrect conclusions can be made by considering the observed correlations as causal relationships.

- In the online shop scenario, the responsible employees are causing the delays.
- In the second scenario, we may conclude that over time the employees get more and more reckless, and consequently the rate of deviations increases.
- In the IT company, we may conclude that the more people working on a project, the more time is spent on team management and communication, which prolongs the project unnecessarily.

However, correlation does not mean causation. We can have a high correlation between two events when they both are caused by a possibly unmeasured (hidden) common cause (set of common causes), which is called a *confounder*. For example, in the first scenario, the delayed deliveries are mainly for the bigger size packages which are usually assigned to specific employees. Or, in the second scenario, the deviations happen in the most complicated cases that are usually handled by the most experienced employees. In the third scenario, maybe both the number of employees working on a project and the duration of a project are highly dependent on the complexity of the project. As it is obvious from these examples, changing the process based on the observed correlations not only leads to no improvement but also may aggravate the problem or create new ones.

Two general frameworks for finding the causes of a problem are (*1*) randomized experiments and the (*2*) theory of causality [3,4]. A randomized experiment provides the most robust and reliable method for making causal inferences and statistical estimates of the effect of an intervention, i.e., intentionally changing the value of a feature. This method involves randomly setting the values of the features that have a causal effect on the observed problem and monitoring the effects. Applying randomized experiments in the context of processes is usually too expensive (and sometimes unethical) or simply impossible. The other option for anticipating the effect of any intervention on the process is using a *structural equation model* [3,4]. In this method, first, the causal mechanism of the process features is modeled by a conceptual model, and then this model is used for studying the effect of changing the value of a process feature.

The main benefit of modeling the relationships among the process feature using a structural equation model, over those methods that are based on mere correlations, is the possibility to investigate the distribution of unseen data. Using a structural equation model, we can study the effect of interventions on one of the process features on the other features.

This paper is an extension of [5], where we have proposed a method for *causal analysis using structural equation modeling*. Here we address one of the main issues in this method. *Finding the features that may have a causal effect on the problem often requires substantial domain knowledge*. Moreover, considering the variety of the feature values in a process, even in the presence of extensive domain knowledge, it may not be easy to determine those values of the features that have the strongest effect on the problem. So, we propose a simple yet effective method for finding a set of features and feature value pairs that may contribute the most to the problem. Applying causal inference on a smaller set of features not only increases the time efficiency of the causal inference but also results in simpler and consequently more understandable structural equation models. Moreover, we add aggregated features to the features that can be extracted from the event log, which makes the method capable of analyzing more scenarios. The method explained in this paper includes the following five steps:

1. As a preprocessing step, the event log is enriched by several process-related features. These features are derived from different data sources like the event log, the process model, and the conformance checking results. Also, here we consider the possibility of adding aggregated features to the event log regarding the time window provided by the user.
2. In the second step, based on the class and descriptive features (denoted in Fig. 1 as class situation feature and descriptive situation features) provided by the user a class-dependent data table is created, which we call *situation feature table*.
3. A set of pairs of the form (feature, feature value) are recommended to the user where the set of recommended features is a subset of descriptive features in the situation feature table created in the second step. Such pairs include the descriptive features that might have a causal relationship with the problem and those values of them that possibly contribute more to the problem. Users can modify this set of features that have been identified automatically or simply ignore it and provide another set of features to create a new situation feature table (or trim the situation feature table).

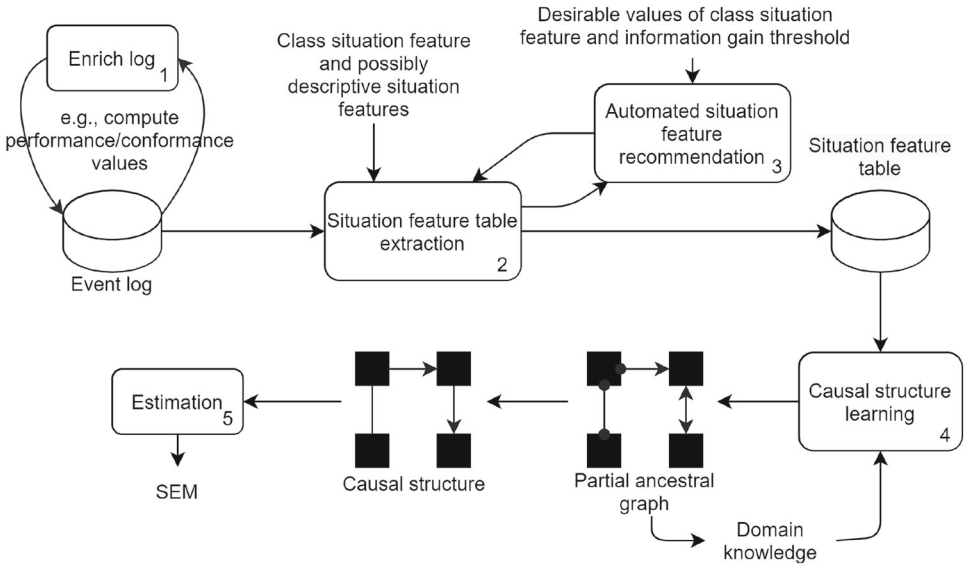


Fig. 1 The general structural equation model discovery approach. Each step is annotated with a number which is corresponding to the number of the item dedicated to explaining that step in the overview of the method in Sect. 1

4. The fourth step involves generating a graphical object encoding the structure of causal relationships among the process features. This graphical object can be provided by the customer or be inferred from the observational data (the situation feature table from the previous step) using a causal structure learning algorithm, also called *search algorithm*. The user can modify the resulting graphical object by adding domain knowledge as input to the search algorithm and repeating this step or by modifying the discovered graph.
5. The last step involves estimating the strength of each discovered causal relationship and the effect of an intervention on any of the process features on the identified problem.

In Fig. 1, the general overview of the proposed approach for structural equation model discovery is presented. The remainder of the paper is organized as follows. In Sect. 2, we start with an example. We use this example as the running example throughout this paper. In Sect. 3, we present some of the related work. In Sect. 4, we present the approach at a high level. The corresponding process mining and causal inference theory preliminaries are presented in Sect. 5 and, in Sect. 6, an overview of the proposed approaches for feature recommendation and causal equation model discovery is presented. In Sect. 7, the assumptions and the design choices in the implemented plugin and the experimental results of applying it on synthetic and real event logs are presented. Finally, in Sect. 8, we summarize our approach and its applications.

2 Motivating example

As the running example, we use an imaginary IT company that implements software for its customers. However, they do not do the maintenance of the released software. Here, each process instance is corresponding to the process of implementing one software. This process involves the following activities: business case development, feasibility study, product backlog, team charter, development, test, and release. The Petri net for this process is shown in Fig. 2. We refer to the sub-model including two transitions "Development" and "Test" (the two blue activities in Fig. 2) as the *implementation phase*.

The manager of the company is concerned about the duration of the implementation phase of projects. She wants to know what features determine the implementation phase duration. And also, if there is any way to reduce the implementation phase duration. If so, what would be the effect

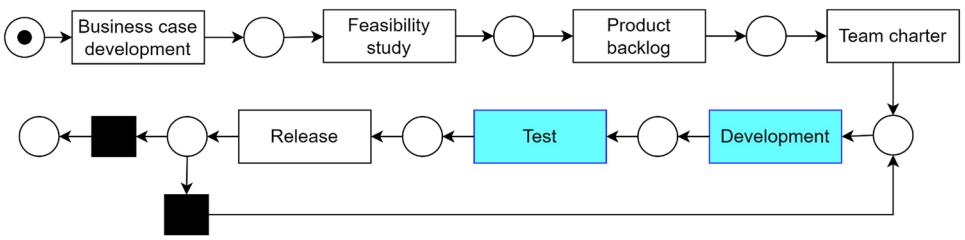


Fig. 2 The Petri net model of the process of IT company

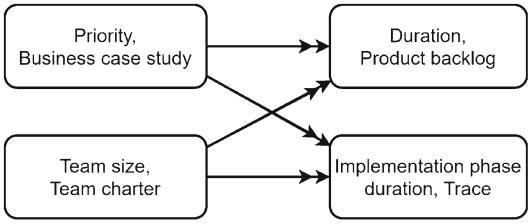


Fig. 3 A possible causal structure for the IT company

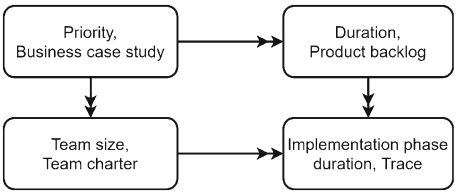


Fig. 4 Another possible causal structure for the IT company

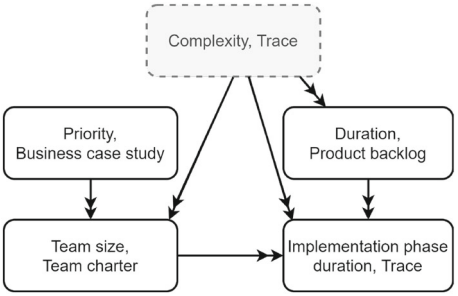


Fig. 5 Another possible causal structure for the IT company

of changing each feature? These are valid questions to be asked before planning for re-engineering and enhancing the process. The manager believes that the following features of a project might have a causal effect on the duration of its implementation phase (i.e., she believes that by changing the value of these features, the duration of the implementation phase will also change.):

- *"Priority"* which is a feature of business case development indicating how urgent the software is for the customer,
- *"Team size"* which is a feature of team charter specifying the number of resources working on a project,
- *"Duration"* of product backlog activity, a feature of product backlog activity, which indicates its duration.

Analyzing the historical data from the company shows that there is a high correlation between every one of these three features and the duration of the implementation phase. We consider *"Complexity"* (the complexity and hardness of a project) as another feature that is not recorded in the event log but has a causal effect on the duration of the implementation phase.

The structure of the causal relationship among the features has a high impact on designing the steps to enhance the process. In Figs. 3, 4, and 5, three possible structures of the causal relationship among the features of the IT company are depicted[1].

According to Fig. 3, just team size and priority of a project have a causal effect on the duration of the implementation phase. But the duration of product backlog does not have any causal effect on it even though they are highly correlated. Consequently, changing product backlog duration does not have any impact on the duration of the implementation phase.

According to Fig. 4, all three features priority, product backlog duration, and team size influence the duration of the implementation phase. Thus, by changing each of these three features, one can influence the duration of the implementation phase.

Based on Fig. 5, we can conclude that the complexity, which is a hidden feature in the model (depicted by the gray dashed rectangle), causally influences both implementation phase duration and product backlog duration. Therefore, the correlation among them is because of having a common cause. Grounded in this causal structure, it is not possible to influence the duration of the implementation phase by forcing product backlog activity to take place in a shorter or longer amount of time.

It is worth noting that not all the features are actionable, i.e., in reality, it is not possible to intervene on some of the features. For example, in the context of this IT company, we can imagine that the manager intervenes on team size by assigning more or fewer people to a project; but he cannot intervene in the complexity of a project. Judging whether a feature can be intervened requires using common sense and domain knowledge.

In the rest of this paper, we show how to answer such questions posed by the manager of our imaginary IT company. We first mention how to extract data in a meaningful way regarding the class feature (implementation phase duration in this example), and then we show how to discover the causal relationships between the process features and the structural equation model of the features that may affect the class feature using our method. Finally, we demonstrate how questions related to investigating the effect of an intervention on the class feature are answered in this framework. In Sect. 7.2, we show the results of applying our method for answering the questions of the IT company manager in this example.

[1] In these three figures and other figures in this paper that visualize networks of feature, labels of the nodes (vertices) are either of form *Attribute name, Trace* if the attribute name is related to a trace-level attribute, or of form *Attribute name, Activity name* if the attribute name is related to an event-level attribute. In the former case, the activity name indicated the activity name of the event that the attribute belongs to.

3 Related work

In the literature, there is plenty of work in the area of process mining dedicated to finding the root causes of a performance or compliance problem. The causal analysis approach of the proposed methods usually involves classification [1,6], and rule mining [7]. The main problem of these approaches is that the findings of these methods are based on correlation which does not necessarily imply causation.

The theory of causation based on the structural causal model has been studied deeply [4]. Also, a variety of domains benefit from applying causal inference methods (e.g. [8,9]). However, there is little work on the application of the theory of causality in the area of process mining. There are some works in process mining that use causality theory. These include:

- In [10], the authors propose an approach for discovering causal relationships between a range of business process characteristics and process performance indicators based on time-series analysis. The idea is to generate a set of time series using the values of performance indicators, and then apply the Granger causality test on them, to investigate and discover their causal relationships. Granger test is a statistical hypothesis test to detect predictive causality; consequently, the causal relationships using this approach might not be true cause-and-effect relationships.
- In [11], the authors use the event log and the BPMN model of a process to discover the structural causal model of the process features. They first apply loop unfolding on the BPMN model of the process and generate a partial order of features. They use the generated partial order to guide the search algorithm. In this work, it is assumed that the BPMN model of a process is its accurate model, which is not always the case.

There is also some work devoted to the case level causal analysis [2,12]. In [12] a method for case-level treatments recommendation has been proposed. The authors identify treatments using an action rule mining technique, and then they use uplift trees to discover subgroups of cases for which a treatment has a high causal effect. In [2], the authors have utilized the causal structure model of the whole process and an optimization method to explain the reason for an undesirable outcome in the case level.

It is worth mentioning that all the process-level causal inference approaches that have been presented above are based on statistical tests for discovering causal relationships. Consequently, these approaches are not feasible when there are a huge number of features. However, none of them provides a method for feature recommendation. Yet, there is some work on influence analysis that aims at finding feature values that correlate with a specific property of the process [13–15]. These methods utilize the frequency of the concurrence of each one of the process feature values with the problematic cases to determine their influence.

4 Overview of the method

In this section, we informally describe the method.

4.1 Data extraction

Process mining techniques start from an *event log* extracted from an information system. The building block of an event log is an *event*. An event indicates that a specific activity happened at a point in time for a process instance. A set of events that are related to a specific process instance are called a *trace*. We can look at an event log as a collection of traces. An event log may include three different levels of attributes: log-level attributes, trace-level attributes, and event-level attributes. However, there is much more performance and conformance-related information encoded in the event log that might be helpful for causal inference and root cause analysis. For example, we can add deviation information, the number of log moves, and the trace duration as additional trace-level features and event duration and next activity as event-level features to enrich the event log. If we are interested in the aggregated feature, such as average trace duration or process workload, then we need to enrich the event log with aggregated features. For that, given $k \in \mathbb{N}$ as the number of time intervals, we divide the time-span of the event log into k consecutive equal length time intervals. The value of an aggregated feature for an event (trace) is the value of that aggregated feature in the time interval that includes its timestamp (the timestamp of its last event).

Example 1 An event log with two traces for the IT company in Sect. 2 is presented in Table 1. This event log has been enriched by adding the duration attribute to its events (duration of each event) and the implementation phase duration attribute to the traces.

One of the fundamental rules of cause and effect is their time precedence. So, assuming negligible recording time while gathering the data by the information system of the companies, we have to extract the data from the part of the trace that happens before the occurrence of the class feature. Thus that data should be extracted from a prefix of the trace which has been recorded before the occurrence of the class feature. We call a prefix of a trace and its (trace-level) attributes a *situation*. Depending on the type of class feature, we can define different types of situations such as:

Table 1 An event log with two traces for the IT company in Sect. 2

Case ID	Activity name	Timestamp	Priority	Team size	Duration	Responsible	Implementation phase duration
1	Business case development	20,10,2018	2			Alice	324
1	Feasibility study	15.1.2019			87	Alice	324
1	Product backlog	19.2.2019			35	Alice	324
1	Team charter	19.3.2019		21	28	Alice	324
1	Development	19.11.2019			245	Alice	324
1	Test	6.2.2020			79	Alice	324
1	Release	8.2.2020			2	Alice	324
2	Business case development	20.2.2019	1			Alex	807
2	Feasibility study	22.2.2019			33	Alex	807
2	Product backlog	26.4.2019			63	Alex	807
2	Team charter	3.5.2019		33	7	Alex	807
2	Development	3.2.2020			276	Alex	807
2	Test	17.4.2020			74	Alex	807
2	Release	25.4.2020			8	Alex	807
2	Development	31.3.2021			340	Alex	807
2	Test	26.7.2021			117	Alex	807
2	Release	29.7.2021			3	Alex	807

- *Trace situation*, when the class feature is one of the trace features, e.g., trace delay, and each situation is a trace.
- *Event situation*, when the class feature is one of the event features, e.g., the duration of activity "Test" (in the context of IT company in Sect. 2), and each situation is a prefix of a trace and its trace-level attributes. In this example, each situation includes a prefix of a trace in the IT company event log ending with an event with the activity name "Test" and the trace-level attributes of that trace.

An interesting subclass of event situations includes those when the class feature refers to the decision in one of the choice places of the process. In this case, each situation would be a prefix of a trace where the last event is the one that happened before the chosen choice place (and the trace-level attributes of that trace). Such a situation is suitable for analyzing the causes of the decision made in a choice place.

To extract the data, we need to know the exact features. However, it is possible to have the same attribute names in several events of the same trace. For example, we might be interested in the "timestamp" of the event with the activity name "Test" and not other events. To overcome this hurdle, we use *situation feature* notation, which is identified by a pair including an attribute name and a group of events for which we are interested in the attribute value. The group of events is determined in terms of the property that they have in common. However, if we are interested in a trace-level attribute, we leave the second element of the situation feature empty. For example, a situation feature may refer to:

- the duration of the trace,
- the timestamp of events with activity name "Test",
- the duration of the events with activity name "Development", or
- the resource of the events that took longer than 80 days.

Moreover, given a situation and a situation feature, we assign the corresponding trace-level attribute value to the situation feature if it is a trace-level situation feature (i.e., the second term of the situation feature is empty). In the case of the event-level situation feature, we assign the corresponding event-level attribute value of the latest event in that situation that belongs to the specified event group (satisfies the required event group properties) to it. For example, considering the second trace of the event log in Table 1 as a situation, we have:

- the duration of the trace is 807 days,
- the timestamp of the event with activity name "Test" is 117,
- the duration of the event with activity name "Development" is 340 days, and
- the resource of the events that took longer than 80 is Alex which is the one for activity "Test" with duration 117 days.

Knowing the situation feature that represents the class feature (which in the rest of the paper we call *class situation feature*), we can turn an event log into a collection of situations with respect to that class situation feature. Here we

consider those collection of all situations extracted from an event log in which each situation is:

- a trace if the second term of the class situation feature is empty. In other words, the first element of the class situation feature is a trace-level attribute name.
- a prefix of a trace and its trace-level attributes in the event log ending with an event that belongs to the event group specified by the second term of the class situation feature. In this case, class situation feature is an event-level situation feature.

Having the set of descriptive situation features and the class situation feature, we can simply map each situation to a data point which we call an *instance*. We call a data table which is a collection of instances driven from a subset of situations (of an event log) a *situation feature table*.

4.2 Feature recommendation

We can extract an astronomical number of situation features from a given event log. Thus a piece of valuable information that can help the process owners in causal inference is the set of situation features that may have causal relationships with the class situation feature. Moreover, considering the variety of the values that can be assigned to a situation feature, another advantageous piece of information is those values of the selected situation features that contribute more to the problem. To provide the stakeholders with such information, We use a method based on information gain for situation feature and value recommendation. Knowing which values of the class situation feature are undesirable, we compute the information gain of each descriptive situation feature value (we use binning technique in case of numerical situation features). Then those pair of situation features and values are recommended to the user with information gain bigger than a given threshold.

Information gain quantifies the amount of information gained about the class situation feature from a descriptive situation features. It measures the reduction in information entropy of class situation feature by conditioning on a descriptive situation feature. The number of situation features considered as possible causes of the class situation feature is more using information gain than other more enhanced techniques for feature recommendation. However, based on our experiments, more enhanced techniques are more prone to fail to discover the whole set of parents and ancestors of the class situation feature (descriptive situation features connected to class situation feature by a directed path longer than one).

4.3 Causal inference

We can encode the causal relationships among the situation features of a given situation feature table, in the form of a set of equations which are called *Structural Equation Model (SEM)*. In other words, a SEM encodes how the data has been generated and hence the observational distribution (the distribution that the data come from). To discover the SEM of the data, we need to know the structure of the causal relationships among the situation features, as well as the strength of each causal relationship.

The structure of the causal relationships among the situation features in a situation feature table can be captured and presented in the form of a graph, which we call a *causal structure*. A causal structure is a Directed Acyclic Graph (DAG) in which each vertex is corresponding to a situation feature. If a situation feature is a direct cause of another situation feature then there should be a directed edge from the corresponding vertex of the former situation feature to the corresponding vertex of later one in the causal structure. The causal structure of the data can be provided by the user. However, if the user does not have such information, then we can approach the causal structure discovery in a data-driven manner by analyzing statistical properties of situation features in the situation feature table (observational data).

The number of potential causal structures grows exponentially with the number of situation features which indicate the hardness of the causal structure discovery problem [16]. Yet, several algorithms have been proposed in the literature with this purpose. We can do causal structure inference using a *causal structure learning algorithm* which (also called a *search algorithm*). The search algorithms use partial correlation tests to determine the existence of a potential causal relationship between two features[2]. There are two main types of search algorithms [17]:

- Score-based methods, where the goal is finding a DAG (as the causal structure of the data) that maximizes the likelihood of the data, according to a fitness score indicating how good the DAG describes the data. Refer to [18–20] for some examples of score-based search algorithms.
- Constraint-based methods, where conditional independence tests on data are used as constraints to construct the DAG structure. Refer to [21–23] for some examples of construct based search algorithms.

[2] Loosely speaking, a partial correlation test is a statistical test designed to measure the degree of association between two features (random variables) where the effect of a set of other features (random variables) has been removed.

The output of the search algorithm is not always a DAG, but a *Partial Ancestral Graph (PAG)* which is a graphical object encoding all statistically supported causal structures by the data. A PAG is simply a graph with four types of edges. Each edge type has semantics and encodes a piece of information about the causal structure of the corresponding situation features in its ends. To turn the discovered PAG into a causal structure, we can use domain knowledge and common sense. For example we can guide the search algorithm by adding *required* and *forbidden directions*. A required direction indicates a causal relationship that must exist in the causal structure whereas, a forbidden direction indicates a causal relationship that should not exist in the causal structure. Having the causal structure of the data, discovering the SEM is simply an estimation problem.

Using SEM of the data, we can predict the effect of the interventions on the process. An (atomic) intervention on a process is done by forcefully setting the value of one of its situation features to a specific value. An example of an intervention in the context of the IT company in Sect. 2 would be setting "Team size" to 5 for all the projects regardless of their other properties. To predict the effect of an intervention on a process, we need to replace the equation corresponding to the situation feature that we intervene on (the equation for "Team size" in the mentioned example) with the fixed value assignment (i.e., with "Team size $= 5$"). The distribution induced by the modified SEM is the interventional distribution describing the behavior of the process under intervention. If the SEM is the correct model, then all the deduced interventional distributions correspond to distributions that we would obtain from randomized experiments [4].

5 Preliminaries

In this section, we describe some of the basic notations and concepts of the process mining and causal inference theory in a more formal way.

5.1 Process mining

In the following section, we follow two goals: first, we describe the basic notations and concepts of the process mining, and second, we show the steps involved in converting a given event log into a situation feature table.

We start by explicitly defining an event, trace, and event log in a way that reflects reality and, at the same time, is suitable for our purpose. But first, we need to define the following universes and functions:

- $\mathcal{U}_{att}$ is the universe of *attribute names*, where $\{actName, timestamp, caseID\} \subseteq \mathcal{U}_{att}$. $actName$ indicates the activity name, $timestamp$ indicates the timestamp of an event, and $caseID$ is an identifier indicating the trace (process instance) that the event belongs to.
- $\mathcal{U}_{val}$ is the universe of *values*.
- $values \in \mathcal{U}_{att} \mapsto \mathbb{P}(\mathcal{U}_{val})$ is a function that returns the set of all possible values of a given attribute name[3].
- $\mathcal{U}_{map} = \{m \in \mathcal{U}_{att} \not\mapsto \mathcal{U}_{val} \mid \forall at \in dom(m) : m(at) \in values(at)\}$ is the universe of all mappings from a set of attribute names to attribute values of the correct type.

Also, we define $\perp$ as a member of $\mathcal{U}_{val}$ such that $\perp \notin values(at)$ for all $at \in \mathcal{U}_{att}$. We use this symbol to indicate that the value of an attribute is unknown, undefined, or is missing.

Now, we define an event as follows:

Definition 1 (Event) An *event* is an element of $e \in \mathcal{U}_{map}$, where $e(actName) \neq \perp$, $e(timestamp) \neq \perp$, and $e(caseID) \neq \perp$. We denote the universe of all possible events by $\mathcal{E}$ and the set of all non-empty chronologically ordered sequences of events that belong to the same case (have the same value for $caseID$) by $\mathcal{E}^+$. If $\langle e_1, \ldots, e_n \rangle \in \mathcal{E}^+$, then for all $1 \leq i < j \leq n$, $e_i(timestamp) \leq e_j(timestamp) \wedge e_i(caseID) = e_j(caseID)$.

Example 2 The events in the following table are some of the possible events for the IT company in Sect. 2.

Each event may have several attributes which can be used to group the events. For $at \in \mathcal{U}_{att}$, and $V \subseteq values(at)$, we define a group of events as the set of those events in $\mathcal{E}$ that assign a value of V to the attribute at; i.e.

$$group(at, V) = \{e \in \mathcal{E} \mid e(at) \in V\}.$$

Some of the possible groups of events are:

- the set of events with specific activity names,
- the set of events which are done by specific resources,
- the set of events that start in a specific time interval during the day, or,
- the set of events with a specific duration.

[3] In this paper, it is assumed that the reader is familiar with sets, multi-sets, and functions. $\mathbb{P}(X)$ is the set of non-empty subsets of set $X \neq \emptyset$. Let X and Y be two sets. $f : X \not\mapsto Y$ is a partial function. The domain of f is a subset of X and is denoted by $dom(f)$. We write $f(x) = \perp$ if $x \notin dom(f)$.

e_1:={($caseID$, 1), ($Responsible$, $Alice$), ($actName$, "Business case development"), ($timestamp$, 20.10.2018), ($Priority$, 2)}
e_2:={($caseID$, 1), ($actName$, "Feasibility study"), ($timestamp$, 15.1.2019)}
e_3:={($caseID$, 1), ($Responsible$, $Alice$), ($actName$, "Product backlog"), ($timestamp$, 19.2.2019), ($Duration$, 35)}
e_4:={($caseID$, 1), ($Responsible$, $Alice$), ($actName$, "Team charter"), ($timestamp$, 19.3.2019), ($Team\ size$, 21)}
e_5:={($caseID$, 1), ($Responsible$, $Alice$), ($actName$, "Development"), ($timestamp$, 19.11.2019), ($Duration$, 245)}
e_6:={($caseID$, 1), ($Responsible$, $Alice$), ($actName$, "Test"), ($timestamp$, 6.2.2020), ($Duration$, 79)}
e_7:={($caseID$, 1), ($Responsible$, $Alice$), ($actName$, "Release"), ($timestamp$, 8.2.2020)}
e_8:={($caseID$, 2), ($Responsible$, $Alex$), ($actName$, "Business case development"), ($timestamp$, 20.2.2019), ($Priority$, 1)}
e_9:={($caseID$, 2), ($Responsible$, $Alex$), ($actName$, "Feasibility study"), ($timestamp$, 22.2.2019)}
e_{10}:={($caseID$, 2), ($Responsible$, $Alex$), ($actName$, "Product backlog"), ($timestamp$, 26.4.2019), ($Duration$, 63)}
e_{11}:={($caseID$, 2), ($Responsible$, $Alex$), ($actName$, "Team charter"), ($timestamp$, 3.5.2019), ($Team\ size$, 33)}
e_{12}:={($caseID$, 2), ($Responsible$, $Alex$), ($actName$, "Development"), ($timestamp$, 3.2.2020), ($Duration$, 276)}
e_{13}:={($caseID$, 2), ($Responsible$, $Alex$), ($actName$, "Test"), ($timestamp$, 17.4.2020), ($Duration$, 74)}
e_{14}:={($caseID$, 2), ($Responsible$, $Alex$)($actName$, "Release"), ($timestamp$, 25, 4, 2020)}
e_{15}:={($caseID$, 2), ($Responsible$, $Alex$), ($actName$, "Development"), ($timestamp$, 31.3.2021), ($Duration$, 340)}
e_{16}:={($caseID$, 2), ($Responsible$, $Alex$), ($actName$, "Test"), ($timestamp$, 26.7.2021), ($Duration$, 117)}
e_{17}:={($caseID$, 2), ($Responsible$, $Alex$), ($actName$, "Release"), ($timestamp$, 29.7.2021)}

We denote the universe of all event groups by $\mathcal{G} = \mathbb{P}(\mathcal{E})$.

For the sake of simplicity, we restrict the group of events to be defined based on conditions expressed on a single attribute. However, in principle, it is not a restriction. It is possible to define a group of events via multi-attribute conditions. To do so, the user may enrich the event log by adding a derivative attribute considering the required conditions on multi-attributes of interest and then use the new derivative attribute to define a group of events.

Example 3 Here are some possible event groups based on the IT company in Sect. 2.

G_1:=$group(actName$, {"Business case development"})
G_2:=$group(actName$, {"Product backlog"})
G_3:=$group(actName$, {"Team charter"})
G_4:=$group(actName$, {"Development"})
G_5:=$group(team\ size$, {33, 34, 35})

G_1, G_2, G_3, and G_4 group the events based on their activity name. For example, event group G_1 is the set of events with activity name "Business case development" (i.e., $G_1 = \{e \in \mathcal{E} \mid e(actName) =$ "Business case development"$\}$.). However, event group G_5 represents the group of events for which the value of attribute $team\ size$ is 33, 34, or 35.

Based on the definition of an event, we define an event log as follows:

Definition 2 (Event Log) We define the universe of all event logs as $\mathcal{L} = \mathcal{E}^+ \times \mathcal{U}_{map}$. We call each element $(\sigma, m) \in L$ where $L \in \mathcal{L}$ a *trace* in which σ represent the sequence of events of the trace and m is a mapping from the trace-level attribute names to their values (possibly with an empty domain).

One of our assumptions in this paper is the uniqueness of events in event logs; i.e., given an event log $L \in \mathcal{L}$, we have $\forall(\sigma_1, m_1), (\sigma_2, m_2) \in L : e_1 \in \sigma_1 \wedge e_2 \in \sigma_2 \wedge e_1 = e_2 \implies (\sigma_1, m_1) = (\sigma_2, m_2)$ and $\forall(\langle e_1, \ldots, e_n\rangle, m) \in L : \forall 1 \leq i < j \leq n : e_i \neq e_j$. This property can easily be ensured by adding an extra identity attribute to the events.

Also, we assume that the uniqueness of the "caseID" value for traces in a given event log L. In other words, $\forall(\sigma_1, m_1), (\sigma_2, m_2) \in L : e_1 \in \sigma_1 \wedge e_2 \in \sigma_2 \wedge e_1(caseID) = e_2(caseID) \implies (\sigma_1, m_1) = (\sigma_2, m_2)$.

Example 4 $L_{IT} = \{\lambda_1, \lambda_2\}$ is a possible event log for the IT company in Sect. 2. L_{IT} includes two traces λ_1 and λ_2, where:

λ_1:=($\langle e_1, \ldots e_7\rangle$, {($caseID$, 1), ($Responsible$, $Alice$), ($implementation\ phase\ duration$, 324)})

and

λ_2:=($\langle e_8, \ldots e_{17}\rangle$, {($caseID$, 2), ($Responsible$, $Alex$), ($implementation\ phase\ duration$, 807)}).

As a preprocessing step, we enrich the event log by adding many derived features to its traces and events. There are many different derived features related to any of the process perspectives; the time perspective, the data flow-perspective, the control-flow perspective, the conformance perspective, or the resource/organization perspective of the process. We can compute the value of the derived features from the event log or possibly other sources.

Moreover, we can enrich the event log by adding aggregated attributes to its events and traces. We define an aggregated feature as follows:

Definition 3 (Aggregated Attribute) Let $\mathcal{L}$ be the universe of event logs, $\mathbb{N}^+$ a non-zero natural number (which indicates

the number of time windows), $\mathcal{U}_{att}$ the universe of attribute names, and $values(timestamp)$ the domain of timestamp. We call an attribute name in $\mathcal{U}_{att}$ an *aggregated attribute* if its value is determined using a function

$$\xi \in \mathcal{L} \times \mathcal{U}_{att} \times \mathbb{N}^{+} \times values(timestamp) \rightarrow \mathbb{R}.$$

Function $\xi(L, ag, k, t)$ where $L \in \mathcal{L}$, $ag \in \mathcal{U}_{att}$, $k \in \mathbb{N}^{+}$, and $t \in values(timestamp)$, returns the corresponding aggregated value of attribute ag at time t where we partition the time between the minimum and maximum timestamp in L to k consecutive time windows with equal width. To compute the value of an aggregated attribute ag for and event $e \in L$ (in the event-level) considering k time windows, we use $\xi(L, ag, k, e(timestamp))$ while for a trace $t = (\sigma, m) \in L$ (in the event-level) considering k time windows, we use $\xi(L, ag, k, t)$ where $t = \max\{e(timestamp) \mid e \in \sigma\}$.

Some of the possible aggregated attributes are: the number of waiting customers, process workload, average service time, average waiting time, number of active events with a specific activity name, number of waiting events with a specific activity name, average service time, average waiting time of a resource.

While extracting the data from an event log, we assume that the event recording delays by the information system of the process were negligible. Moreover, we assume that all the trace-level features were recorded before the execution of the trace. Considering the time order of cause and effect, we have that only the features that have been recorded before the occurrence of a specific feature can have a causal effect on it. So the relevant part of a trace to a given feature is a prefix of that trace and its trace-level attributes, which we call a *situation*. Let $prfx(\langle e_1, \ldots, e_n\rangle) = \{\langle e_1, \ldots, e_i\rangle \mid 1 \leq i \leq n\}$, a function that returns the set of non-empty prefixes of a given sequence of events. Using *prfx* function we define a situation as follows:

Definition 4 (Situation) Let $\mathcal{L}$ be the universe of all event logs. We define the universe of all situations as $\mathcal{U}_{situation} = \bigcup_{L \in \mathcal{L}} S_L$ where $S_L = \{(\sigma, m) \mid \sigma \in prfx(\sigma') \wedge (\sigma', m) \in L\}$ is the set of situations of an event log $L \in \mathcal{L}$. Moreover, we call each element $(\sigma, m) \in \mathcal{U}_{situation}$ a *situation*.

Among the possible subsets of S_L of a given event log L, we distinguish two important situation subset types of S_L. The first type is the *G-based situation subset* of L where $G \in \mathcal{G}$ and includes those situations in S_L that their last event (the event with maximum timestamp) belongs to G. The second type is the *trace-based situation subset*, which includes the set of all traces of L[4].

[4] If a process includes decision points, then one of the derived attributes that can be added to the event log when enriching the event log is the *choice* attribute. A choice attribute is added to the activity that happens before the decision point and its value indicates which activity has been enabled as the result of the decision that has been made. So we can use an added choice attribute and its values to group the events in an event log and extract a situation subset based on the occurrence of that specific choice. We already defined two important types of situation subsets; group-based situation subsets and trace-based situation subsets. We can also distinguish the *choice-based situation subsets* where the situation subset is extracted based on events that have a specific choice attribute. These situation subsets are important as they are conceptually related to a decision point. However, we do not emphasise on them as they are a subset of *G-based situation subset* of L where $G \in \mathcal{G}$ and can be handled the same way.

Definition 5 (Situation Subset) Let $S_L \subseteq \mathcal{U}_{situation}$ be the set of situations for $L \in \mathcal{L}$, and $G \in \mathcal{G}$, we define

- *G-based situation subset* of L as $S_{L,G} = \{(\langle e_1, \ldots, e_n\rangle, m) \in S_L \mid e_n \in G\}$, and
- *trace-based situation subset* of L as $S_{L,\perp} = L$.

Example 5 Three situations s_1, s_2, and s_3, where $s_1, s_2, s_3 \in S_{L1T,G_4}$ (G_4 in Example 3, generated using the traces in Example 4 are as follows:

$$\begin{aligned}
s_1{:=}(\langle e_1, \ldots e_5\rangle, \{&(caseID, 1), (Responsible, Alice),\\
&(implementation\ phase\ duration, 324)\})\\
s_2{:=}(\langle e_8, \ldots e_{12}\rangle, \{&(caseID, 2), (Responsible, Alex),\\
&(implementation\ phase\ duration, 807)\})\\
s_3{:=}(\langle e_8, \ldots e_{15}\rangle, \{&(caseID, 2), (Responsible, Alex),\\
&(implementation\ phase\ duration, 807)\})
\end{aligned}$$

Note that $G_4{:=}group(actName, \{\text{“Development”}\})$ and we have $\{e_5, e_{12}, e_{15}\} \subseteq G_4$. In other words $e_5(actName) = e_{12}(actName) = e_{15}(actName) = \text{“Development”}$.

When extracting the data, we need to distinguish trace-level attributes from event-level attributes. We do that by using *situation features* which is identified by a group of events, G (possibly $G = \perp$), and an attribute name, at. Each situation feature is associated with a function defined over the situations. This function returns the proper value for the situation feature regarding at and G extracted from the given situation. More formally:

Definition 6 (Situation Feature) We define $\mathcal{U}_{sfeature} = \mathcal{U}_{att} \times (\mathcal{G} \cup \{\perp\})$ as the universe of *situation features*. Each situation feature $sf = (at, G)$ where $at \in \mathcal{U}_{att}$, and $G \in \mathcal{G} \cup \{\perp\}$ is associated with a function $\#_{sf} : \mathcal{U}_{situation} \not\rightarrow \mathcal{U}_{val}$ such that:

- if $G = \perp$, then $\#_{(at,G)}((\sigma, m)) = m(at)$ and
- if $G \in \mathcal{G}$, then $\#_{(at,G)}((\sigma, m)) = e(at)$ where $e = argmax_{e' \in G \cap \{e'' \in \sigma\}} e'(timestamp)$

for $(\sigma, m) \in \mathcal{U}_{situation}$. We denote the universe of the situation features as $\mathcal{U}_{sfeature}$.

We can consider a situation feature as an analogy to the feature (a variable) in tabular data. Also, we can look at the corresponding function of a situation feature as the function that determines the mechanism of extracting the value of the situation feature from a given situation. Given a situation (σ, m) and a situation feature (at, G), if $G = \perp$, its corresponding function returns the value of at in trace-level (i.e., $m(at)$). However, if $G \neq \perp$, then the function returns the value of at in $e \in \sigma$ that belongs to G and happens last (has the maximum timestamp) among those events of σ that belong to G.

Example 6 We can define the following situation features using the information provided in the previous examples:

$$sf_1{:=}(Team\ size, G_3)$$
$$sf_2{:=}(Duration, G_2)$$
$$sf_3{:=}(Priority, G_1)$$
$$sf_4{:=}(Duration, G_4)$$
$$sf_5{:=}(Implementation\ phase\ duration, \perp),$$

where event groups G_1, G_2, G_3, and G_4 has been defined in Example 3.

Also, considering s_1 (Example 5), we have:

$$\#_{sf_1}(s_1) = 21 \qquad \#_{sf_3}(s_1) = 2$$
$$\#_{sf_2}(s_1) = 35 \qquad \#_{sf_4}(s_1) = 245$$
$$\#_{sf_5}(s_1) = 324$$

where s_1 is one of the situations in 5.

We interpret a nonempty set of situation features, which we call it a *situation feature extraction plan*, as an analog to the schema of tabular data. More formally;

Definition 7 (Situation Feature Extraction Plan) We define a *situation feature extraction plan* as $\boldsymbol{SF} \subseteq \mathcal{U}_{sfeature}$ where $\boldsymbol{SF} \neq \emptyset$.

Example 7 A possible situation feature extraction plan for the IT company in Sect. 2 is as follows:

$$\begin{aligned}\boldsymbol{SF}_{IT} =& \{(Team\ size, G_3), (Duration, G_2),\\ & (Priority, G_1), (Duration, G_4)\}\\ =& \{sf_1, sf_2, sf_3, sf_4\}.\end{aligned}$$

We can map each situation to a data point according to a given situation feature extraction plan. We do that as follows:

Definition 8 (Instance) Let $s \in \mathcal{U}_{situation}$ and $\boldsymbol{SF} \subseteq \mathcal{U}_{sfeature}$ where $\boldsymbol{SF} \neq \emptyset$. We define the *instance* $inst_{\boldsymbol{SF}}(s)$ as $inst_{\boldsymbol{SF}}(s) \in \boldsymbol{SF} \not\rightarrow \mathcal{U}_{val}$ such that $\forall sf \in \boldsymbol{SF} : (inst_{\boldsymbol{SF}}(s))(sf) = \#_{sf}(s)$.

An instance is a set of pairs where each pair is composed of a situation feature and a value. With a slight abuse of notation, we define $values(sf) = values(at)$ where $sf = (at, G)$ is a situation feature.

Example 8 Considering $\boldsymbol{SF}_{IT}$ from Example 6 and the situations from Example 5. We have:

$$\begin{aligned}&inst_{\boldsymbol{SF}_{IT}}(s_1) = \{((Team\ size, G_3), 21), ((Duration, G_2), 35),\\ &\quad ((Priority, G_1), 2), ((Duration, G_4), 245)\} = \{(sf_1, 21),\\ &\quad (sf_2, 35), (sf_3, 2), (sf_4, 245)\}\\ &inst_{\boldsymbol{SF}_{IT}}(s_2) = \{((Team\ size, G_3), 33), ((Duration, G_2), 63),\\ &\quad ((Priority, G_1), 1), ((Duration, G_4), 276)\}\\ &\quad = \{(sf_1, 33), (sf_2, 63), (sf_3, 1), (sf_4, 276)\}\\ &inst_{\boldsymbol{SF}_{IT}}(s_3) = \{((Team\ size, G_3), 33), ((Duration, G_2), 63),\\ &\quad ((Priority, G_1), 1), ((Duration, G_4), 340)\}\\ &\quad = \{(sf_1, 33), (sf_2, 63), (sf_3, 1), (sf_4, 340)\}\end{aligned}$$

Given a situation feature extraction plan $\boldsymbol{SF}$, we consider one of its situation features as the class situation feature, denoted as csf and $\boldsymbol{SF} \setminus \{csf\}$ as descriptive situation features. Given $\boldsymbol{SF} \subseteq \mathcal{U}_{sfeature}$, $csf \in \boldsymbol{SF}$ where $csf = (at, G)$, and an event log L, we can generate a class situation feature sensitive tabular data-set. We call such a tabular data set a *situation feature table*. To do that, we first generate $S_{L,G}$ and then we generate the situation feature table which is the bag of instances derived from the situations in $S_{L,G}$, regarding $\boldsymbol{SF}$. Note that choosing $S_{L,G}$ such that G is the same group in the class situation feature (where we have $csf = (at, G)$), ensures the sensitivity of the extracted data to the class situation feature. More formally;

Definition 9 (Situation Feature Table) Let $L \in \mathcal{L}$ be an event log, $\boldsymbol{SF} \subseteq \mathcal{U}_{sfeature}$ a situation feature extraction plan, and $csf = (at, G) \in \boldsymbol{SF}$. We define a *situation feature table* $T_{L,\boldsymbol{SF},(at,G)}$ (or equivalently $T_{L,\boldsymbol{SF},csf}$) as follows:

$$T_{L,\boldsymbol{SF},(at,G)} = [inst_{\boldsymbol{SF}}(s) \mid s \in S_{L,G}].$$

Note that if $csf = (at, G)$ where $G \in \mathcal{G}$, then the situation feature table $T_{L,\boldsymbol{SF},csf}$ includes the instances derived from the situations in G-based situation subset $S_{L,G}$. However, if $G = \perp$, then it includes the situations derived from the situations in trace-based situation subset $S_{L,\perp}$.

Example 9 Based on Example 8 we have

$$\begin{aligned}&T_{L_{IT},\boldsymbol{SF}_{IT},(\text{Duration},G_4)}\\ &\quad = [inst_{\boldsymbol{SF}_{IT}}(s_1), inst_{\boldsymbol{SF}_{IT}}(s_2), inst_{\boldsymbol{SF}_{IT}}(s_3)].\end{aligned}$$

Note that in this example, the class situation feature is $csf = sf_4 = (\text{Duration}, G_4)$. Another way to present $T_{L_{IT},\boldsymbol{SF}_{IT},(\text{Duration},G_4)}$ is as follows:

In this table, each row includes the values of the situation features in $\boldsymbol{SF}_{IT}$ (Example 7) extracted from one of the situations s_1, s_2, and s_3. The first row is corresponding to the $inst_{\boldsymbol{SF}_{IT}}(s_1)$, the second row is corresponding to the $inst_{\boldsymbol{SF}_{IT}}(s_2)$, and the third row is corresponding to the $inst_{\boldsymbol{SF}_{IT}}(s_3)$.

$sf_1 = (Team\ size, G_3)$	$sf_2 = (Duration, G_2)$	$sf_3 = (Priority, G_1)$	$sf_4 = (Duration, G_4)$
21	35	2	245
33	63	1	276
33	63	1	340

5.2 Structural equation model

A structural equation model is a data-generating model in the form of a set of equations. Each equation encodes how the value of one of the situation features is determined by the value of other situation features[5]. It is worth noting that these equations are a way to determine how the observational and the interventional distributions are generated and should not be considered as normal equations. More formally[6];

Definition 10 (Structural Equation Model (SEM)) Let $T_{L,\boldsymbol{SF},csf}$ be a situation feature table, in which $L \in \mathcal{L}$, $\boldsymbol{SF} \subseteq \mathcal{U}_{sfeature}$, and $csf \in \boldsymbol{SF}$. The SEM of $T_{L,\boldsymbol{SF},csf}$ is defined as $\mathcal{EQ} \in \boldsymbol{SF} \rightarrow Expr(\boldsymbol{SF})$ where for each $sf \in SF$, $Expr(sf)$ is an expression of the situation features in $\boldsymbol{SF} \setminus \{sf\}$ and possibly some noise N_{sf}. It is needed that the noise distributions N_{sf} of $sf \in \boldsymbol{SF}$ be mutually independent.

We need $\boldsymbol{SF}$ to be *causal sufficient*, which means $\boldsymbol{SF}$ includes all relevant situation features. We assume that SEMs are acyclic; i.e., given a SEM $\mathcal{EQ}$ over the $\boldsymbol{SF}$ of a situation feature table $T_{L,\boldsymbol{SF},csf}$, for each $sf \in \boldsymbol{SF}$, the right side of expression $sf = Expr(\boldsymbol{SF})$ in $\mathcal{EQ}$ does not include sf.

Given $\mathcal{EQ}$ over the $\boldsymbol{SF}$ of a situation feature table $T_{L,\boldsymbol{SF},csf}$, the *parents* of the $sf \in \boldsymbol{SF}$ is the set of situation features that appear in the right side of expression $\mathcal{EQ}(sf)$. The set of parents of a situation feature includes those situation features with a direct causal effect on it.

Example 10 A possible SEM for the situation feature table shown in Example 9 is as follows:

$(Priority, G_1) = N_{(Priority,G_1)}$	$N_{(Priority,G_1)} \sim Uniform(1, 3)$
$(Team\ size, G_3) = 10(Priority, G_1) + N_{(Team\ size,G_3)}$	$N_{(Team\ size,G_3)} \sim Uniform(1, 15)$
$(Duration, G_2) = 2(Team\ size, G_3) + N_{(Duration,G_2)}$	$N_{(Duration,G_2)} \sim Uniform(-5, 5)$
$(Duration, G_4) = 5(Duration, G_2) + 10(Priority, G_1) + (Team\ size, G_3) + N_{(Duration,G_4)}$	$N_{(Duration,G_4)} \sim Uniform(-100, 100)$

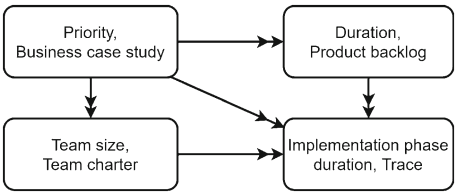


Fig. 6 The causal structure of the SEM in Example 10

The structure of the causal relationships between the situation features in a SEM can be encoded as a directed acyclic graph, which is called *causal structure*. Given a SEM $\mathcal{EQ}$ on a set of situation features $\boldsymbol{SF}$, each vertex in its corresponding causal structure is analogous to one of the situation features in $\boldsymbol{SF}$. Let $sf_1, sf_2 \in \boldsymbol{SF}$, there is a directed edge from sf_1 to sf_2 if sf_1 appears in the right side of expression $\mathcal{EQ}(sf_2)$. More formally,

Definition 11 (Causal Structure) Let $\mathcal{EQ}$ be the SEM of the situation feature table $T_{L,\boldsymbol{SF},csf}$. We define the corresponding *causal structure* of $\mathcal{EQ}$ as a directed acyclic graph $(\boldsymbol{U}, \twoheadrightarrow)$ where $\boldsymbol{U} = \boldsymbol{SF}$ and $(sf_1, sf_2) \in \twoheadrightarrow$ if $sf_1, sf_2 \in \boldsymbol{SF}$ and sf_1 appears in the right side of expression $\mathcal{EQ}(sf_2)$.

In the rest of this paper, we use $sf_1 \twoheadrightarrow sf_2$ instead of $(sf_1, sf_2) \in \twoheadrightarrow$.

Having a situation feature table $T_{L,\boldsymbol{SF},csf}$, the structural equation model of its situation features can be provided by a customer who possesses the process domain knowledge or in a data-driven manner.

Example 11 The causal structure of the SEM in Example 10 is as depicted in Fig. 6.

To predict the effect of manipulating one of the situation features on the other situation features, we need to intervene on the SEM by actively setting the value of one (or more) of its situation features to a specific value (or a distribution). Here, we focus on atomic interventions where the intervention is done on just one of the situation features by actively forcing its value to be a specific value.

Definition 12 (Atomic Intervention) Given an SEM $\mathcal{EQ}$ over $\boldsymbol{SF}$ where $sf \in \boldsymbol{SF} \setminus \{csf\}$, and $c \in values(sf)$, the SEM $\mathcal{EQ}'$ after the intervention on sf is obtained by replacing $\mathcal{EQ}(sf)$ by $sf = c$ in $\mathcal{EQ}$.

Note that the corresponding causal structure of $\mathcal{EQ}'$ (after intervention on sf) is obtained from the causal structure of $\mathcal{EQ}$

[5] In the case of nominal situation features, each equation determines the distributions of one of the situation feature values based on the value of other situation features.

[6] Definition 10 and 12 are based on [4].

by removing all the incoming edges to sf [4]. When we intervene on a situation feature, we just replace the equation of that situation feature in the SEM and the others do not change as causal relationships are autonomous under interventions [4].

Example 12 We can intervene on the SEM introduced in Example 10 by forcing the team size to be 13. For this case, the SEM under the intervention is as follows:

$(Priority, G_1) = N_{(Priority,G_1)}$	$N_{(Priority,G_1)} \sim Uniform(1, 3)$
$(Team\ size, G_3) = 13$	
$(Duration, G_2) = 2(Team\ size, G_3) + N_{(Duration,G_2)}$	$N_{(Duration,G_2)} \sim Uniform(-5, 5)$
$(Duration, G_4) = 5(Duration, G_2) + 10(Priority, G_1) + (Team\ size, G_3) + N_{(Duration,G_4)}$	$N_{(Duration,G_4)} \sim Uniform(-100, 100)$

Please note that in Definition 12 (and consequently in Example 12), we just consider atomic interventions in the sense of forcefully setting one of the situation features to a fixed value regardless of the value of other features. In general, it is possible to intervene on a situation feature by intentionally assigning values from a specific distribution. As an example, in Example 12, it is possible to replace the equation for $(Team\ size, G_3)$ (in the SEM presented in Example 11) by

$$(Team\ size, G_3) = 20(Priority, G_1).$$

The above intervention captures the situation where the number of resources assigned to a project is 20 times its priority.

6 Approach

Observing a problem in the process, we need to find a set of situation features $\boldsymbol{SF}$ which not only include csf (the situation feature capturing the problem) but also be causal sufficient (i.e., no hidden confounder exists). The expressiveness of the discovered SEM is highly influenced by $\boldsymbol{SF}$ (even though SEMs, in general, can deal with latent variables). Considering the variety of the possible situation features captured by the event log and the derived ones, finding the proper set $\boldsymbol{SF}$ and also those values of the situation features (or combination of values) that contribute more to the problem is a complicated task and needs plenty of domain knowledge.

We know that correlation does not mean causation. On the other hand, if a situation feature is caused by another situation feature (set of situation features), this implies that there is a correlation between the given situation feature and its parents. We use this simple fact for the automated situation feature recommendation. It is worth noting that there are many situation recommendation methods possible. The automated situation feature recommendation method and the SEM discovery process are described in the following:

6.1 Automated situation feature recommendation

Given an event log $L \in \mathcal{L}$ and the class situation feature name $csf = (at, G)$, we consider a situation feature name sf a possible cause of csf if there exists a value $v \in values(sf)$ that appears more in the situations with the undesirable (problematic) result for csf. In other words, we are looking for those descriptive situation features such that at least for one of the values in their domain the probability of having an undesirable result for class situation feature increases. To identify such a situation feature and situation feature values, we use the information gain concept. But first, we need to turn the situation feature table into a binary situation feature table in which the class situation feature is binary (based on being a desirable or an undesirable outcome).

Let $T_{L,\boldsymbol{SF},csf}$ be a situation feature table where $csf = (at, G)$ and $\boldsymbol{SF} = (sf_1, \ldots, sf_n, csf)$. Moreover, let $values(csf)_{\downarrow}$ be the set of undesirable values of csf. We can define $Tb_{L,\boldsymbol{SF},csf,values(csf)_{\downarrow}}$ as a situation feature table with binary class situation feature as follows:

$$\begin{aligned} &Tb_{L,\boldsymbol{SF},csf,values(csf)_{\downarrow}} \\ &= [\{\#_{sf_1}(s), \ldots, \#_{sf_n}(s), 1\} \mid s \in S_{L,G} \wedge \#_{csf}(s) \\ &\quad \notin values(csf)_{\downarrow}] \uplus \\ &[\{\#_{sf_1}(s), \ldots, \#_{sf_n}(s), 0\} \mid s \in S_{L,G} \wedge \#_{csf}(s) \\ &\quad \in values(csf)_{\downarrow}]. \end{aligned}$$

We can derive this binary situation feature table from $T_{L,\boldsymbol{SF},csf}$ by replacing the class situation feature value in every instance by 0 or 1 depending on being desirable or undesirable. Now, we define the potential intervention set of situation feature and situation value pairs as follows:

Definition 13 (potential Intervention pairs) Let $L \in \mathcal{L}$ be an event log, $\boldsymbol{SF} \subseteq \mathcal{U}_{sfeature}$ a nonempty set of descriptive features, and $csf = (at, G)$ the class situation feature where $csf \in \boldsymbol{SF}$ and $G \in \mathcal{G} \cup \{\perp\}$. Moreover, consider α as a threshold where $0 < \alpha \leq 1$ and $values(csf)_{\downarrow} \subset val(csf)$ as the set of undesirable values for csf. We call a pair (sf, v) where $sf \in \boldsymbol{SF} \setminus \{csf\}$ and $v \in values(sf)$ a *potential intervention pair* if

$$IG_{L,\boldsymbol{SF},csf,values(csf)_{\downarrow}}(sf, v) \geq \alpha$$

in which $IG_{L,\boldsymbol{SF},csf,values(csf)_{\downarrow}}(sf, v)$ is the information gain of splitting the instances in the binary situation feature table $Tb_{L,\boldsymbol{SF},csf,values(csf)_{\downarrow}}$ by $sf = v$ and $sf \neq v$.

We present the set of the potential causes to the user as a set of tuples (sf, v) where $sf \in \mathcal{U}_{sfeature}$ and $v \in values(sf)$ in the descending order regarding the information gain of splitting the binary situation feature table by $sf = v$ and $sf \neq v$. This way, the first tuples in the order are those values of those situation features that intervention on them may have (potentially) the most effect on the value of the class situation feature. The choice of the value α depends on the application and is determined by the user.

The selected set of situation features by this method is the set of situation features for which the information gain is more than the given threshold α. The user can use this set as the descriptive set of situation features in the situation feature extraction plan to generate a situation feature table with fewer situation features. Let's call such a situation feature table which contains just the selected set of situation features a *trimmed situation feature table*.

6.2 SEM inference

Here we show how to infer the SEM of a given situation feature table in two steps:

- The first step is *causal structure discovery*, which involves discovering the causal structure of the situation feature table. This causal structure encodes the existence and the direction of the causal relationships among the situation features in the situation extraction plan of the given situation feature table.
- The second step is *causal strength estimation*, which involves estimating a set of equations describing how each situation feature is influenced by its immediate causes. Using this information we can generate the SEM of the given situation feature table.

6.2.1 Causal structure discovery

The causal structure of the situation features in a given situation feature table can be determined by an expert who possesses domain knowledge about the underlying process and the causal relationships between its features. But having access to such knowledge is quite rare. Hence, we support discovering the causal structure in a data-driven manner.

Several search algorithms have been proposed in the literature (e.g., [22,24,25]). The input of a search algorithm is observational data in the form of a situation feature table (and possibly knowledge) and its output is a graphical object that represents a set of causal structures that cannot be distinguished by the algorithm. One of these graphical objects is *Partial Ancestral Graph (PAG)* introduced in [26].

A PAG is a graph whose vertex set is $\boldsymbol{V} = \boldsymbol{SF}$ but has different edge types, including $\rightarrow, \leftrightarrow, \circ\!\!\rightarrow, \circ\!\!-\!\!\circ$. Similar to $\twoheadrightarrow$, we use infix notation for $\rightarrow, \leftrightarrow, \circ\!\!\rightarrow, \circ\!\!-\!\!\circ$. Each edge type has a specific meaning. Let $sf_1, sf_2 \in \boldsymbol{V}$. The semantics of different edge types in a PAG are as follows:

- $sf_1 \rightarrow sf_2$ indicates that sf_1 is a direct cause of sf_2.
- $sf_1 \leftrightarrow sf_2$ means that neither sf_1 nor sf_2 is an ancestor of the other one, even though they are probabilistically dependent (i.e., sf_1 and sf_2 are both caused by one or more hidden confounders).
- $sf_1 \circ\!\!\rightarrow sf_2$ means sf_2 is not a direct cause of sf_1.
- $sf_1 \circ\!\!-\!\!\circ sf_2$ indicates that there is a relationship between sf_1 and sf_2, but nothing is known about its direction.

The formal definition of a PAG is as follows [26]:

Definition 14 (Partial Ancestral Graph (PAG)) Let $\boldsymbol{SF} \subseteq \mathcal{U}_{sfeature}$ be a situation feature extraction plan. A PAG is a tuple $(\boldsymbol{V}, \rightarrow, \leftrightarrow, \circ\!\!\rightarrow, \circ\!\!-\!\!\circ)$ in which $\boldsymbol{V} = \boldsymbol{SF}$ and $\rightarrow, \leftrightarrow, \circ\!\!\rightarrow, \circ\!\!-\!\!\circ \subseteq \boldsymbol{V} \times \boldsymbol{V}$ such that $\rightarrow, \leftrightarrow, \circ\!\!\rightarrow$, and $\circ\!\!-\!\!\circ$ are mutually disjoint.

The discovered PAG by the search algorithm represents a class of causal structures that satisfies the conditional independence relationships discovered in the situation feature table and ideally, includes its true causal structure. Each edge in the discovered PAG indicates a statistically supported (potential) causal relationship among the situation features in the situation feature table. This graph can be used as initial insight into the causal relationships of the situation features by the user.

Example 13 Two possible PAGs for the SEM in Example 10 are shown in Fig. 7.

Now, it is needed to modify the discovered PAG to a compatible causal structure. To transform the output PAG to a compatible causal structure, which represents the causal structure of the situation features in the situation feature table, domain knowledge of the process and common sense can be used. This information can be applied by directly modifying the discovered PAG or by adding them to the search algorithm, as an input, in the form of *required directions* or *forbidden directions* denoted as D_{req} and D_{frb}, respectively. $D_{req}, D_{frb} \subseteq \boldsymbol{V} \times \boldsymbol{V}$ and $D_{req} \cap D_{frb} = \emptyset$. Required directions and forbidden directions influence the discovered PAG as follows:

- If $(sf_1, sf_2) \in D_{req}$, then we have $sf_1 \rightarrow sf_2$ or $sf_1 \circ\!\!\rightarrow sf_2$ in the output PAG.
- If $(sf_1, sf_2) \in D_{frb}$, then in the discovered PAG it should not be the case that $sf_1 \rightarrow sf_2$.

We assume no hidden common confounder exists, so we expect that in the PAG, relation $\leftrightarrow$ be empty. If $\leftrightarrow \neq \emptyset$, the user can restart the procedure after adding more situation features to the situation feature table. We can define the compatibility of a causal structure with a PAG as follows:

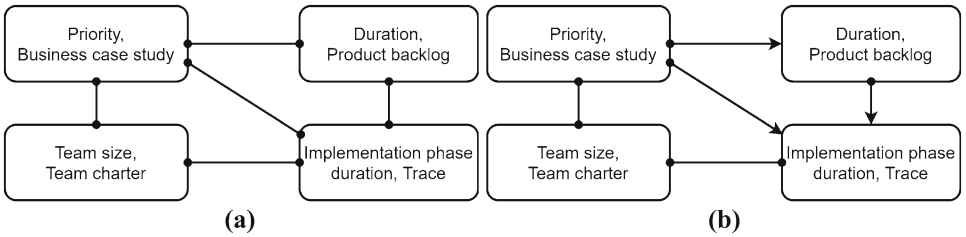


Fig. 7 Two possible PAGs for the SEM presented in Example 10

Definition 15 (Compatibility of a Causal Structure With a Given PAG) Given a PAG $(\boldsymbol{V}, \rightarrow, \leftrightarrow, \bullet\!\!\rightarrow, \bullet\!\!-\!\!\bullet)$ in which $\leftrightarrow = \emptyset$, we say a causal structure $(\boldsymbol{U}, \twoheadrightarrow)$ is compatible with the given PAG if $\boldsymbol{V} = \boldsymbol{U}$, $(sf_1 \rightarrow sf_2 \vee sf_1 \bullet\!\!\rightarrow sf_2) \implies sf_1 \twoheadrightarrow sf_2$, and $sf_1 \bullet\!\!-\!\!\bullet sf_2 \implies (sf_1 \twoheadrightarrow sf_2 \oplus sf_2 \twoheadrightarrow sf_1)$, where $\oplus$ is the XOR operation and $sf_1, sf_2 \in \boldsymbol{V}$ [7].

It is worth noting that the assumption of the absence of hidden confounders plays an important role in the definition of compatibility of a causal structure with a PAG. For example, it enables us to infer $sf_1 \twoheadrightarrow sf_2$ from $sf_1 \bullet\!\!\rightarrow sf_2$ while this implication might not be true in general as it may signifies the existence of a confounder (one or more) which causes both sf_1 and sf_2.

Example 14 The causal structure shown in Fig. 6 is compatible with both PAGs demonstrated in Fig. 7.

6.2.2 Causal strength estimation

The final step of discovering the causal model is estimating the strength of each direct causal effect using the observed data. We do that by estimating each situation feature by a function of its parents and a noise function. We can estimate the strength of the causal relationships in the following manner. Let D be the causal structure of a situation feature table $T_{L,\boldsymbol{SF},csf}$. As D is a directed acyclic graph, we can sort its vertices in a topological order γ. Now, we can statistically model each situation feature as a function of the noise terms of those situation features that appear earlier in the topological order γ. In other words, $sf = f\left((N_{sf'})_{sf':\gamma(sf')\leq\gamma(sf)}\right)$ [4]. The set of these functions, for all $sf \in \boldsymbol{SF}$, is the SEM of $\boldsymbol{SF}$. Note that the set of situation features that appear earlier than a situation feature in the topological order γ of D includes the parents of that situation feature and none of its descendants.

Finally, we want to answer questions about the effect of an intervention on any of the situation features on the class situation feature. We can do the intervention as described in Definition 12. The resulting SEM (after intervention) demonstrates the effect of the intervention on the situation features.

Note that, if there is no directed path between $sf \in \boldsymbol{SF}$ and csf, in the causal structure of a situation feature table $T_{L,\boldsymbol{SF},csf}$, they are independent and consequently, intervention on sf by forcing $sf = c$ has no effect on csf.

7 Experimental results

We have implemented the proposed approach as a plugin in ProM that is available in the nightly build of ProM under the name *Causal Inference Using Structural Equation Model*. ProM is an open-source and extensible platform for process mining [27]. The inputs of the implemented plugin are an event log, the Petri net model of the process, and the conformance checking results of replaying the given event log on the given model. In the rest of this section, first, we mention some of the implementation details and design choices that we have made, and then we present the results of applying the plugin on a synthetic and several real-life event logs.

7.1 Implementation notes

In the implemented plugin, we first enrich the event log by adding some attributes. Some of the features that can be extracted from the event log using the implemented plugin are as follows:

- Time perspective: timestamp, activity duration, trace duration, trace delay, sub-model duration.
- Control-flow perspective: next activity, previous activity.
- Conformance perspective: deviation, number of log moves, number of model moves.
- Resource organization perspective: resource, role, group.
- Aggregated features (regarding a given time window):
  - Process perspective: the number of waiting cases, process workload.
  - Trace perspective: average service time, average waiting time.
  - Event perspective: number of active events with a specific activity name, number of waiting events with a specific activity name.

[7] Please note that even though according to this definition the causal structure may just have the causal relationships compatible with the discovered PAG, in the implemented plugin the user can freely modify causal relationships by adding or removing edges in the discovered PAG or provide the causal structure starting from an empty graph.

– Resource perspective: average service time, average waiting time

Let $L \in \mathcal{L}$ be an event log, $k \in \mathbb{N}$ (a non-zero natural number) the number of time windows, t_{min} the minimal timestamp, and t_{max} the maximum timestamp in L, we divide the time span of L into k consecutive time windows with equal length (the length of each time window is $(t_{max} - t_{min})/k$ and compute the value of aggregated attributes for each of these k time windows. We define $\xi : \mathcal{L} \times \mathcal{U}_{att} \times \mathbb{N} \times values(timestamp) \rightarrow \mathbb{R}$ as a function that given an event log, an aggregated attribute name, the number of time windows, and a timestamp returns the value of the given aggregated attribute in the time window that includes the timestamp. We can use ξ for aggregated attributes at both the event and the trace-level. More precisely, given $L \in \mathcal{L}$, $(\sigma, m) \in L$, $e \in \sigma$, $k \in \mathbb{N}$, and $at \in \mathcal{U}_{att}$ where at is an aggregated attribute, we define $e(at) = \xi(L, at, k, e(timestamp))$ and $m(at) = \xi(L, at, k, t')$ where $t' = max\{e(timestamp) \mid e \in \sigma\}$.

In other words, we can use any of the (process, trace event, and resource perspective) aggregated features in both event and trace levels. At the event-level, we compute the value of the aggregated feature in the time window including the timestamp of the event. While in the trace-level, we compute its value for the time window that includes the timestamp of the last event of the trace. It is worth noting that there are different possible design choices on how to compute and enrich the event log with aggregated features.

As the second step, the user needs to specify *csf* and $\boldsymbol{SF}$. The user can specify $\boldsymbol{SF}$ by manually selecting the proper set of situation features or use the implemented situation feature and value recommendation method on an initial situation feature table (for example an initial situation feature table in which the descriptive situation features includes all the implemented situation features) to identify the relevant set of situation features to *csf*.

According to the selected $\boldsymbol{SF}$ and *csf* the proper situation subset of the event log is generated and the situation feature table is extracted. Then we infer the causal structure of the situation feature table. For this goal, we use the Greedy Fast Causal Inference (GFCI) algorithm [25] which is a hybrid search algorithm. The inputs of GFCI algorithm are the situation feature table and possibly background knowledge. The output of GFCI algorithm is a PAG with the highest score on the input situation feature table. In [25], it has been shown that under the large-sample limit, each edge in the PAG computed by GFCI is correct if some assumptions hold. Also, the authors of [25], using empirical results on simulated data, have shown that GFCI has the highest accuracy among several other search algorithms. Some of the assumptions that need to hold to ensure the correctness of the discovered causal structure of the situation features by GFCI considering the large sample limits are:

- Independence and identically distribution of the instances in the situation feature table.
- *Causal Markov condition* which is a form of local causality [22]. This condition states that a situation feature is independent of all other situation features except its decedents, given its direct causes (parents).
- *Causal faithfulness condition* [22]. This condition states that all the independence relationships among the measured situation features are implied by the causal Markov condition.
- No selection bias which implies that the presence of each instance in the situation feature table is independent of the values of its measured situation features.
- The existence of no feedback cycle among the measured situation features.

Assessing the satisfaction of these conditions by the situation feature table is non-trivial task. For example, for the first condition, we need the instances in the situation feature table to be independent and identically distributed. This assumption is probably violated in many cases when the class situation feature is in the form of (G, at) where $G \in \mathcal{G}$. In this case, each trace in the event log may map to multiple situations (and consequently to several rows of the table) which are not independent. It is worth noting that even if one or more of these assumptions are violated, the PAG generated by GFCI algorithm may still include correct edges (but there are no theoretical guarantees for that).

In the implemented plugin, we have used the Tetrad [28] implementation of the GFCI algorithm. To use the GFCI algorithm, we need to set several parameters. We have used the following settings for the parameters of the GFCI algorithm in the experiments: cutoff for p-values = 0.05, maximum path length = -1, maximum degree = -1, and penalty discount = 2.

In the implemented plugin, we have assumed linear dependencies among the situation features and additive noise when dealing with continuous data. In this case, given a SEM $\mathcal{EQ}$ over $\boldsymbol{SF}$, we can encode $\mathcal{EQ}$ as a weighted graph. This weighted graph is generated from the corresponding causal structure of $\mathcal{EQ}$ by considering the coefficient of sf_2 in $\mathcal{EQ}(sf_1)$ as the weight of the edge from sf_2 to sf_1. Using this graphical representation of the SEM, to estimate the magnitude of the effect of *sf* on the *csf*, we can simply sum the weights of all directed paths from *sf* to *csf*, where the weight of a path is equal to the multiplication of the weights of its edges.

7.2 Synthetic event log

For the synthetic data, we use the IT company example in Sect. 2. The situation feature extraction plan is:

$$\{(Team\ size, G_3), (Duration, G_2), (Priority, G_1), (Implementation\ phase\ duration, \perp)\}$$

where the class situation feature is (*Implementation phase duration*, $\perp$). We assume that the true causal structure of the data is as depicted in Fig. 5.

To generate an event log, we first created the Petri-net model of the process as shown in 2 using CPN Tools [29]. Then, using the created model, we generated an event log with 1000 traces. We have enriched the event log by adding the duration of each event to each event and also *Implementation phase duration* attribute to the traces. The later attribute indicates the duration of the sub-model including "development" and "test" transitions. When generating the log, we have assumed that the true SEM of the process, which we call it $\mathcal{EQ}_1$, is as follows:

$$\begin{aligned}
&(Complexity, \perp) = N_{(Complexity,\perp)}\\
&(Priority, G_1) = N_{(Priority,G_1)}\\
&(Duration, G_2) = 10(Complexity, \perp) + N_{(Duration,G_2)}\\
&(Team\ size, G_3) = 5(Complexity, \perp) + 3(Priority, G_1) + N_{(Team\ size,G_3)}\\
&(Implementation\ phase\ duration, \perp) = 50(Complexity, \perp) +\\
&5(Team\ size, G_3) + N_{(Implementation\ phase\ duration,\perp)}
\end{aligned}$$

$$\begin{aligned}
&N_{(Complexity,\perp)} \sim Uniform(1, 10)\\
&N_{(Priority,G_1)} \sim Uniform(1, 3)\\
&N_{(Duration,G_2)} \sim Uniform(-2, 4)\\
&N_{(Team\ size,G_3)} \sim Uniform(-1, 2)\\
&N_{(Implementation\ phase\ duration,\perp)} \sim Uniform(10, 20)
\end{aligned}$$

The summary of the generated event log and its trace variants (generated by ProM) are shown in Fig. 8.

Generating situation feature table.

We generate a situation feature table using the above situation feature extraction plan. A snapshot of the generated situation feature table using the implemented plugin is shown in Fig. 9. In this figure, the class situation feature is colored in pink and the descriptive situation features are colored gray. Please note that in the pictures of this section, $(Duration, G_2)$ is denoted as "Duration, Product backlog", $(Implementation\ phase\ duration, \perp)$ is denoted as "Implementation phase duration, Trace", $(Team\ size, G_3)$ is denoted as "Team size, Team charter", $(Priority, G_1)$ is denoted as "Priority, Business case study", and $(Complexity, \perp)$ is denoted as "Complexity, Trace".

SEM inference.

Applying the implemented plugin on the situation feature extracted from the event log, the PAG depicted in Fig. 10a was discovered. In this PAG, $(Complexity, \perp)$ has not been considered as a descriptive situation feature. As a consequence, two potential causal relationships between $(Duration, G_2)$ and $(Implementation\ phase\ duration, \perp)$ and also between $(Duration, G_2)$ and $(Team\ size, G_3)$ have been discovered which do not exist in data-generating model $\mathcal{EQ}_1$. The customer may guess that another influential attribute might exist that acts as a confounder. Considering $(Complexity, \perp)$ as another descriptive situation feature, then the discovered PAG by the implemented plugin would be as the one in Fig. 10b. This PAG is more accurate and includes the true causal structure of the situation feature table. We have assumed that the complexity of a project is a feature that is not recorded in the event log. The customer, may assume (based on domain knowledge) that the duration of "product backlog" is longer in more complex projects and assign to the complexity of a project the floor of the value of $(Duration, G_2)$ divided by 10[8]. Now, using domain knowledge and the chronological order of transitions, we can turn the discovered PAG into the causal structure depicted in Fig. 10c. After estimating the strength of the causal relationships, we obtain the SEM shown in Fig. 10d.

Moreover, we can have the inferred SEM in text format. In this case, the output would be as shown in Fig. 11.

By comparing the estimated coefficients of situation features names in the output of the plugin (and equivalently the weights of the edges in Fig. 10d), and those in the equations of the true SEM of the data, we can see that the estimated and real strengths of causal relationships are quite close.

Using the discovered SEM, we can answer the question posed by the manager of the IT company in Sect. 2. For example, we can see that $(Team\ size, G_3)$, $(Priority, G_1)$, and $(Complexity, \perp)$ have a causal effect on (*Implementation phase duration*, $\perp$), but $(Duration, G_2)$ does not. To investigate the effect of an intervention on any of the situation features on the class situation feature, we can find the equation capturing the effect of intervention by simply clicking on its corresponding vertex in the causal structure. For example, if we click on the corresponding vertex of $(Team\ size, G_3)$, we have

$$\begin{aligned}
&(Implementation\ phase\ duration, \perp)\\
&= 75.0004 \times (Complexity, \perp) + noise.
\end{aligned}$$

[8] Please note that the choice of the denominator has a high effect on the discovered potential causal relationships by this method (on the discovered PAG).

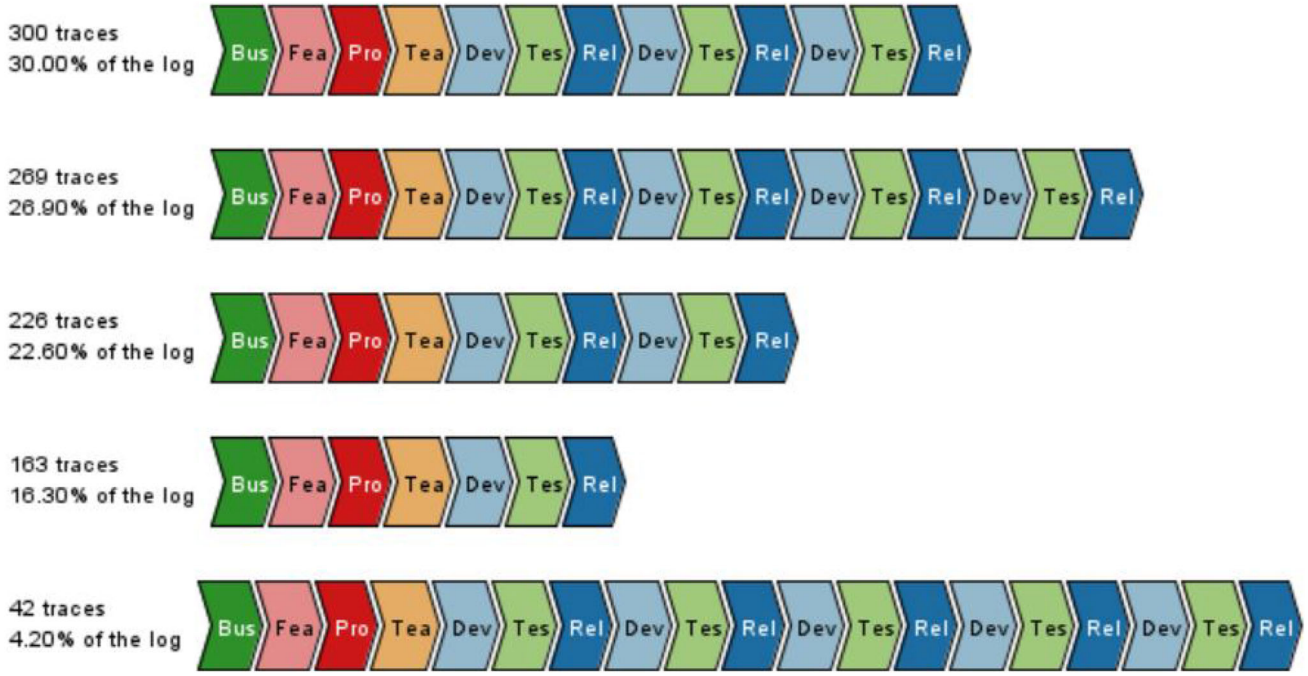


Fig. 8 The trace variants in the synthetic event log

Duration, Product backlog	Implementation phase duratio...	Team size, Team charter	Priority, Business case study
32	266	20	1
49	432	34	3
12	138	15	3
71	577	42	2
30	265	20	2
12	100	8	1
72	590	46	3
82	641	45	2
52	413	30	2
91	740	54	3
41	357	28	3
20	200	18	3
28	283	23	2

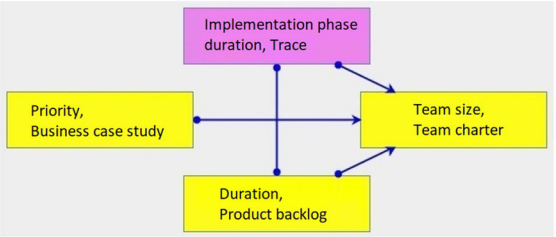
Fig. 9 A snapshot of the situation feature table generated for the synthetic event log

This equation means that by enforcing the complexity of a project to be one unit more complex, then we expect that its implementation phase takes approximately 75 more days (assuming that the complexity of a project is actionable). As another example, equation ($Implementation\ phase\ duration, \perp$)$= 0.0 \times (Duration, G_2)$ shows the estimated effect of intervention on $(Duration, G_2)$. We can interpret this equation as "intervention on $(Duration, G_2)$ has no effect on $(Implementation\ phase\ duration, \perp)$".

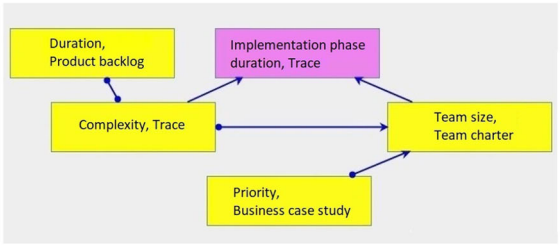
The heat map of the correlation among different situation features in this experiment is shown in Fig. 12. As it is observed in this figure, there is a high correlation between $(Implementation\ phase\ duration, \perp)$ and $(Duration, G_2)$ $(Team\ size, G_3)$, and $(Complexity, \perp)$. Thus, if we consider situation features with high correlation with the class situation feature as its causes, then we would consider $(Duration, G_2)$ as one of the causes of $(Implementation\ phase\ duration, \perp)$ which is in contradiction with the data-generating model. On the other hand, we could find the correct causal relationships using the proposed method.

To investigate the effect of the amount of the noise on the discovered SEM, we have generated three more synthetic event logs with the true SEM of them are similar to $\mathcal{EQ}_1$ except for the noise functions $N_{(Duration, G_2)}$, $N_{(Team\ size, G_3)}$, and $N_{(Implementation\ phase\ duration, \perp)}$[9]. We

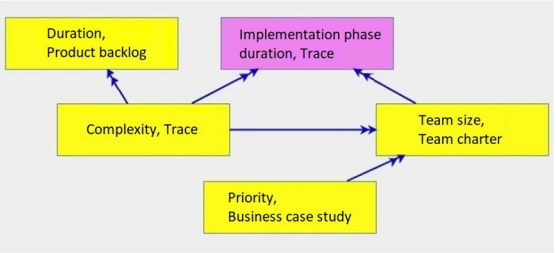
[9] To generate event logs such that their true SEM of the process is similar to $\mathcal{EQ}_1$ with customized uniform noise intervals, you can use Footnote 9 continued



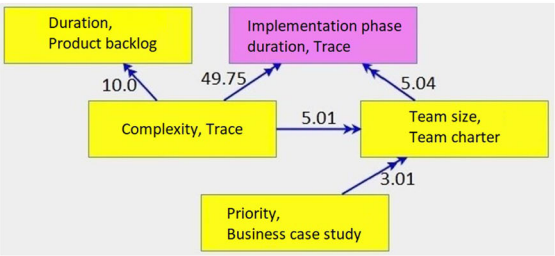
(a) The discovered PAG by applying the implemented plugin on the situation feature table extracted using the situation feature extraction plane in this section.



(b) The discovered PAG after adding $(Complexity, \perp)$ to the situation feature extraction plan (as one of the descriptive situation features).



(c) The causal structure which is obtained by modifying the PAG in 10b based on common sense and domain knowledge.



(d) The inferred SEM by estimating the strength of the discovered causal relationships.

Fig. 10 The PAG, causal structure and the SEM discovered using implemented plugin for the synthetic event log

```
Duration, Product backlog = Complexity, Trace * 10.0 + noise
Implementation phase duration, Trace  = Team size, Team charter * 5.04 + Complexity, Trace * 49.75 + noise
Team size, Team charter = Priority, Business case study * 3.01 + Complexity, Trace * 5.01 + noise
Priority, Business case study = noise
Complexity, Trace = noise
```

Fig. 11 The discovered SEM from the situation feature table extracted from the synthetic event log after adding $(Complexity, \perp)$ to the situation feature extraction plan

call these SEMs $\mathcal{EQ}_2$, $\mathcal{EQ}_3$, and $\mathcal{EQ}_4$ where their noise functions are as follows (we add the noise functions of $\mathcal{EQ}_1$ to this table for the completeness):

where the true SEM of the process are respectively $\mathcal{EQ}_2$, $\mathcal{EQ}_3$, and $\mathcal{EQ}_4$. The situation feature extraction plan in all of these experiments is:

	$N_{(Complexity,\perp)}$	$N_{(Priority,G_1)}$	$N_{(Duration,G_2)}$	$N_{(Team\ size,G_3)}$	$N_{(Implementation\ phase\ duration,\perp)}$
$\mathcal{EQ}_1$	$Uniform(1, 10)$	$Uniform(1, 3)$	$Uniform(-2, 4)$	$Uniform(-1, 2)$	$Uniform(10, 20)$
$\mathcal{EQ}_2$	$Uniform(1, 10)$	$Uniform(1, 3)$	$Uniform(-2, 58)$	$Uniform(-1, 29)$	$Uniform(10, 210)$
$\mathcal{EQ}_3$	$Uniform(1, 10)$	$Uniform(1, 3)$	$Uniform(-2, 118)$	$Uniform(-1, 59)$	$Uniform(10, 310)$
$\mathcal{EQ}_4$	$Uniform(1, 10)$	$Uniform(1, 3)$	$Uniform(-2, 178)$	$Uniform(-1, 89)$	$Uniform(10, 410)$

Figure 13, on the left side, shows the heat map of the correlation between the situation features of the generated event logs and, on the right side, the PAG discovered by the implemented plugin. Moreover, in this figure, the top, middle, and button parts demonstrate the result of the experiment

$$\{(Team\ size, G_3), (Duration, G_2), (Priority, G_1), (Implementation\ phase\ duration, \perp), (Complexity, \perp)\}.$$

As it is shown in Fig. 13, top and middle part, with a relatively high noise this method id capable of finding possibility of potential causal relationships among the situation features, even though it fails to discover the direction of dis-

the ProM plugin *generate event log for IT company with selected noise intervals*.

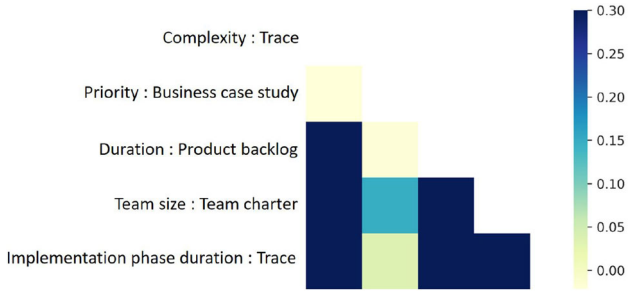


Fig. 12 The heat map of the correlation between the situation features of the IT company

covered possible relationships. However, when the amount of the noise is too high, this method fails to discover all the possibility of potential causal relationships. This result was expected as GFCI algorithm utilize partial correlation tests to discover possible causal relationships in the data.

7.3 Time and quality evaluation

Here, we evaluate the time efficiency and quality of the selected situation feature set by the proposed situation feature and value recommendation method. By the quality of a selected situation features set, we mean the portion of the causal relationships with the class situation feature that has been preserved in the trimmed situation feature table.

To evaluate the time efficiency of the selected set of situation features by the proposed method (which we call it *Situation Feature Value Pair Recommendation (SFVPR)*), we compare its performance with two situation feature selection methods;

- Situation Feature Selection using Random Forest (SFSRF) and
- Situation Feature Selection Based on Correlation (SFSBC).

For these two methods, we have used the implementation provided by WEKA [30] with their default setting. More precisely, for SFSRF, a random forest with 100 trees, unlimited maximum depth of the tree, minimum of one instance per leaf, and 10 folds cross validation have been used and. For SFSBC, the backwards search method is greedy and the merit of the found subset is 0.95.

The two main types of feature selection techniques in machine learning are supervised and unsupervised, where the supervised methods are further divided into wrapper, filter and intrinsic [31]. Filter-based methods are based on statistical measures and do not incorporate a specific machine learning algorithm. Wrapper methods, a specific machine learning technique is used to evaluate the best subset of features and the selected features are optimized for that particular machine learning technique. Finally, intrinsic methods utilize those machine learning techniques, such as decision tree and random forest, that perform feature selection automatically as part of learning the model. The proposed situation feature and value recommendation method is a supervised method. We have selected SFSRF as an instance of intrinsic and SFSBC as an instance of a filter method to evaluate our method.

Moreover, we have used the flowing event logs:

- Receipt phase of an environmental permit application process (WABO) CoSeLoG project (receipt log for short) that has 1434 traces [32].
- A subset of business process intelligence (BPI) challenge 2017 event log that includes traces of length at least 20 but at most 30. This event log has 11044 traces [33].
- A subset of BPI challenge 2019 event log that includes traces of length at least 8 but at most 10. This event log has 12574 traces [34].

We have used trace-level class situation features in these experiments. So, the number of instances in the situation feature table is equal to the number of traces in each event log. We have used the receipt event log for the first and second experiments, BPI challenge 2017 event log for the third and fourth experiments, and BPI challenge 2019 event log for the fifth experiment. Figure 14 shows the results of this experiment. We can see that considering time efficiency, SFVPR is comparable with SFSRF and SFSBC. Figure 15 illustrates the number of vertices and edges in the discovered PAGs. Based on Fig. 15, we can see that in all cases the complexity of the discovered PAG in terms of the number of vertices and edges have been reduced.

To evaluate the quality of the recommended situation features, we have generated 10 synthetic event logs with 1000 traces such that each one of them includes 20 situation features[10]. These features are generated based on a randomly generated SEM (that is the true SEM of the data). The causal structures of the randomly generated SEMs include two connected components with equal number of vertices. We have applied feature selection using SFVRP as well as SFSBC and SFSRF. We have created a trimmed situation feature table using the selected set of situation features. Then we have compared the discovered causal structure with the true causal structure of the data. Discovering the SEM of the observed data, having its causal structure, is a simple estimation. So we focus on the differences in the discovered causal structure. In this experiment, we set the number of bins to 20 and $\alpha = 0.01$ (in SFVPR). Moreover, we have considered

[10] The data-generating model and the generated event logs that have been used for this experiment are available in https://github.com/mahnaz-qafari/Experimental-data-generator.

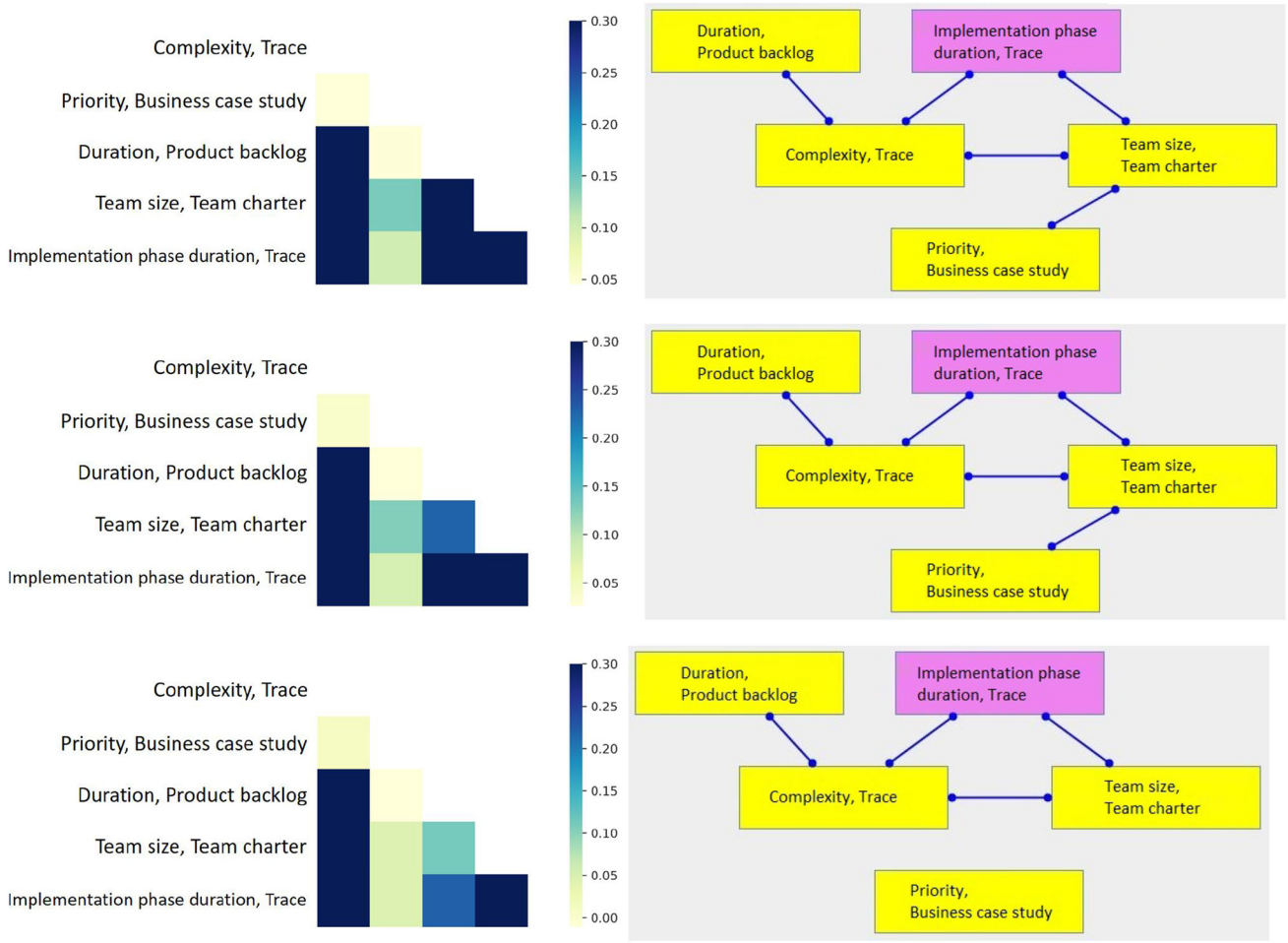


Fig. 13 The heat map of the correlations between the situation features and the discovered PAG using implemented plugin for the synthetic event logs generated such that the true SEMs are $\mathcal{EQ}_2$ (the top figures), $\mathcal{EQ}_3$ (the middle figures), and $\mathcal{EQ}_4$ (the button figures)

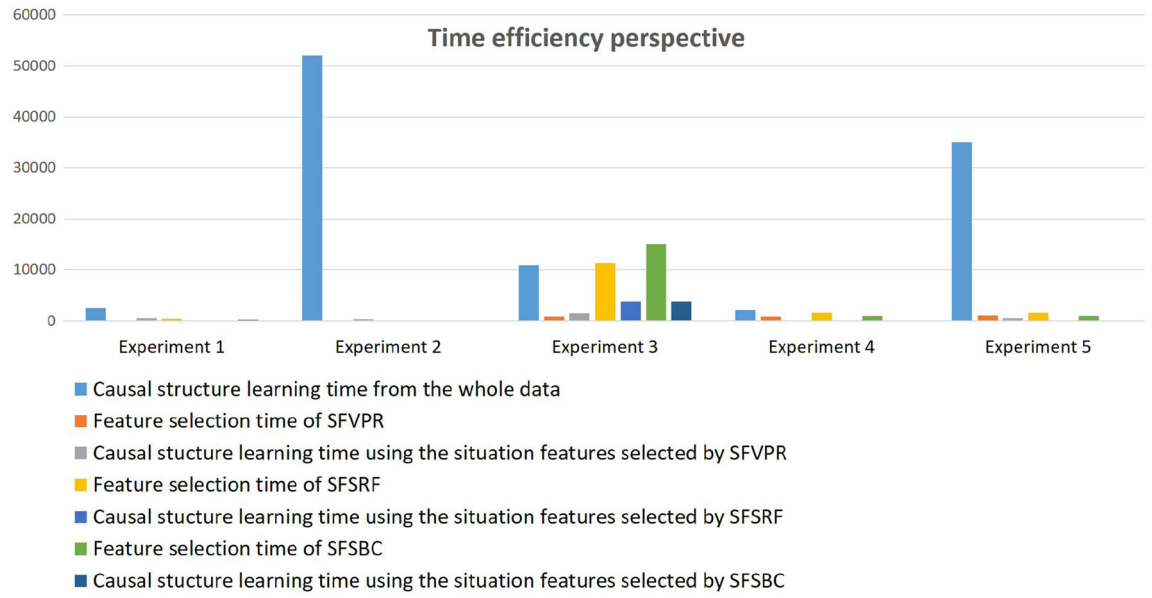


Fig. 14 The time needed for causal structure learning in milliseconds for the whole situation feature table, and the trimmed situation feature table using the situation features selected by SFVPR, SFSRF, and SFSBC

Table 2 The quality evaluation of the selected situation features by FARF (and FSBC) and SFVPR in terms of parent recall, parent precision, causal relationship recall, and causal relationship precision

	True causal structure					Causal structure of recommended situation features by SFSRF					
	Structural properties					Structural properties		Quality metrics			
Event log	Number of descriptive situation features	Number of edges	Number of edges of effective causal structure	Number of parents	Number of ancestors	Number of recommended situation features	Number of edges	Parent recall	Parent precision	Causal relationship recall	Causal relationship precision
1	19	41	7	2	6	2	3	0.5	0.33	0.28	0.67
2	19	40	11	5	6	5	6	0.6	1	0.27	0.5
3	19	32	12	6	8	6	6	0.33	0.67	0.41	0.83
4	19	34	15	4	8	3	4	0.2	0.5	0.13	0.5
5	19	30	11	4	5	3	2	0.25	0.5	0.09	0.5
6	19	34	6	2	4	3	3	1	1	0.5	1
7	19	37	5	3	4	4	6	1	0.5	0.6	0.5
8	19	38	11	5	6	4	6	0.5	0.5	0.36	0.67
9	19	39	9	3	6	4	3	0.33	0.33	0.11	0.33
10	19	35	6	2	5	3	6	0.33	0.33	0.17	0.17

	Causal structure of recommended situation features by SFSBC						Causal structure of recommended situation features by SFVPR					
	Structural properties		Quality metrics				Structural properties		Quality metrics			
Event log	Number of recommended situation features	Number of edges	Parent recall	Parent precision	Causal relationship recall	Causal relationship precision	STAB Number of recommended situation features	Number of edges	Parent recall	Parent precision	Causal relationship recall	Causal relationship precision
1	5	6	1	0.67	0.28	0.33	4	5	1	1	0.71	1
2	5	6	0.6	1	0.27	0.5	7	8	0.4	0.67	0.18	0.25
3	6	6	0.33	0.67	0.41	0.83	8	10	0.33	0.67	0.58	0.7
4	3	4	0.2	0.5	0.5	0.5	8	14	0.4	0.5	0.47	0.5
5	5	4	0.5	0.5	0.5	0.75	4	4	0.25	0.5	0.18	0.5
6	3	3	1	1	1	1	7	10	1	0.67	0.83	0.5
7	5	7	1	1	0.5	0.57	7	10	1	1	0.6.	0.3
8	4	6	0.2	0.5	0.36	0.67	10	12	0.8	1	0.72	0.67
9	4	3	0.33	0.33	0.11	0.33	6	10	0.33	0.5	0.56	0.5
10	4	9	0.5	0.33	0.33	0.22	6	12	1	0.67	0.67	0.36

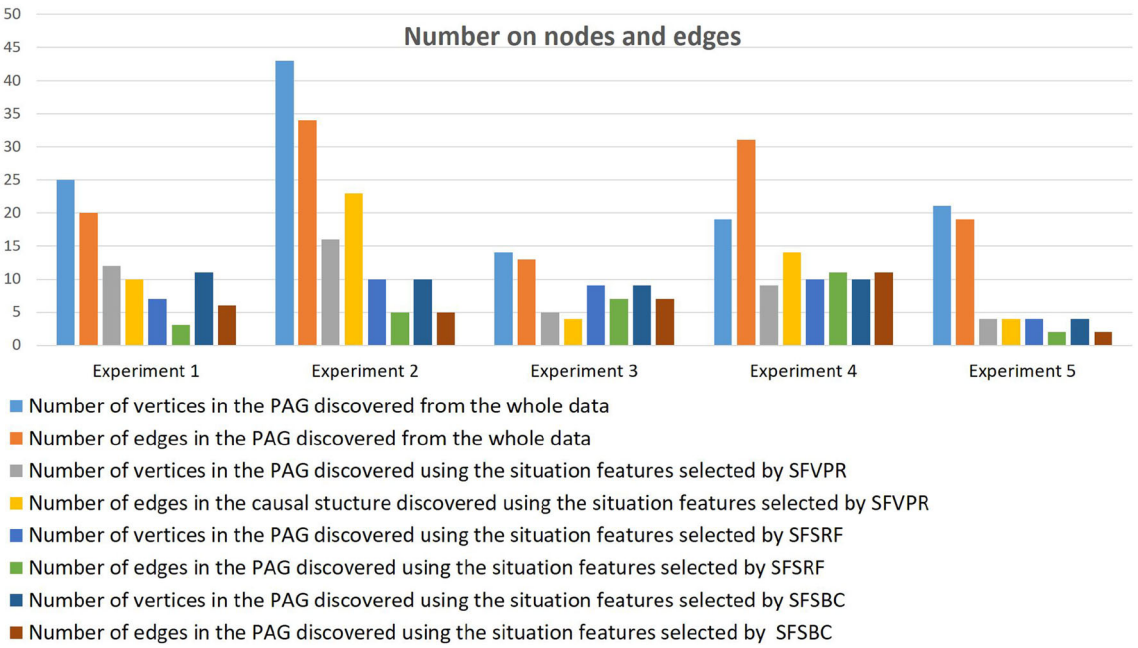


Fig. 15 The number of vertices and edges in the PAG discovered from the whole situation feature table, and in the PAG discovered using the trimmed situation feature table using selected situation features via applying SFVPR, SFSRF, and SFSBC

the mean value of the class situation feature as the threshold where the values lower than this threshold are undesirable class situation feature values.

We call the causal structure obtained by projecting the true causal structure on the set of vertices including the class situation feature and those situation features that have a causal effect on it *effective causal structure*. Also, as in the PAG discovered by the implemented plugin the direction of the potential causal relationships (edges) are not determined (and the discovered PAG has to be modified by the user by modifying the direction of the discovered potential causal relationship), we consider all the situation features whose corresponding vertices are connected to the class situation feature by an edge (regardless of the edge type), the set of *potential parents* of the class situation feature in that PAG. Moreover, consider:

- rsf as the set of selected situation features.
- $ptcs$ as the set of parents of the class situation feature in the true causal structure.
- $pfcs$ as the set of potential parents of class situation feature in the PAG discovered using the trimmed situation feature table.
- $etcs$ as the set of causal relationships (edges) in the effective causal structure.
- $efcs$ as the set of potential causal relationships (edges regardless of their type) in the causal structure discovered using the trimmed situation feature table.

We use the following metrics to quantify the difference in the two causal structures.

$$\text{parent recall} = \frac{|\ ptcs \cap prsfcs\ |}{|\ ptcs\ |},$$

$$\text{parent precision} = \frac{|\ atcs \cap arsfcs\ |}{|\ atcs\ |},$$

$$\text{causal relationship recall} = \frac{|\ (ptcs \cup atcs) \cap rsf\ |}{|\ rsf\ |},$$

$$\text{causal relationship precision} = \frac{|\ (ptcs \cup atcs) \cap rsf\ |}{|\ rsf\ |}.$$

We can interpret the above metrics as follows:

- *parent recall*: the portion of parents of the class situation features in the true causal structure that have been also a potential parent in the PAG discovered using the trimmed situation feature table.
- *parent precision*: the portion of potential parents of the class situation feature in the PAG discovered using the trimmed situation feature table which are also a parent of the class situation features in the true causal structure of the data.
- *causal relationship recall*: the portion of causal relationships in the effective causal structure that have been detected by the PAG discovered using the trimmed situation feature table. structure.
- *causal relationship precision*: the portion of potential causal relationships in the PAG discovered using the trimmed situation feature table which are also a causal relationship in the effective causal structure.

The results of this comparison is presented at Table 2. The above experiment shows that:

- In general, except for causal relationship precision, SFVPR achieved better results than SFSBC and SFSRF. Please note that none of the methods achieved the best results in all the experiments.
- Considering causal relationship precision, SFVPR achieved weaker results in comparison with SFSBC and SFSRF. It can be explained by considering that this method recommends more situation features than the other two methods which usually results in discovering more potential causal relationships in the discovered PAG. In addition, to compute the causal relationship precision, we compare the portion of the potential causal relationships in the discovered PAG on the trimmed situation feature which are also causal relationships in the effective causal structure. The effective causal structure includes a subset of the causal relationships of the connected component of the true causal relationship that includes the class situation feature. Many of the potential causal relationships present in the discovered PAG on the trimmed situation feature by SFVPR method are corresponding to the causal relationships in the connected including the class situation feature but not in the effective causal structure.
- In none of the experiments the recommended set of situation features by SFVPR includes a situation feature that does not belong to that connected component of true causal structure which includes class situation feature. However, in three experiments both SFSBC and SFSRF recommend situation features that do not belong to the same connected component of the true causal structure that includes class situation feature.

8 Conclusion

Distinguishing causal from mere correlational relationships among the process features is a vital task when investigating the root causes of performance and/or conformance problems in a company. The best way to identify the causal relationships is by using randomized experiments. However, this requires implementing process changes to see their effect. As applying randomized experiments is usually quite expensive (if not impossible) in the processes, we propose a method for causal analysis based on the theory of causality which uses a mixture of data analysis and domain knowledge. The stakeholders can use this method to incorporate both domain knowledge and potential statistically supported causal effects to find the SEM of the features and indicators of the process. Moreover, this method helps stakeholders to investigate the effect of an intervention on the process. This information can be used to design and order the re-engineering steps.

The validity of a discovered structural equation model (and any other machine learning technique) is highly influenced by the set of features that have been used for data extraction and consequently for structural equation model discovery. However, the complex and dynamic interdependencies in processes make the task of selecting the set of features with a potential causal effect on the observed problem in the process a challenging task. So, we have proposed a simple yet intuitive and effective feature recommendation method in this paper. The proposed method provides the user not just the set of features with the possible causal effect on the class situation feature but also those values of the features that increase the possibility of the observed problem in the process more than a given threshold. Moreover, we have shown the effectiveness of the proposed method in terms of time efficiency and quality of the selected set of situation features.

As future work, we would like to learn more process-related features that go beyond individual cases. For example, bottlenecks are caused by competing cases or a shortage of resources. Also, notions such as blocking, batching, and overtaking are not captured well. We would also like to make the diagnostics more understandable. This requires mapping diagnoses related to features back onto the process model and event log. Finally, we would like to enhance simulation models with SEM-based rules.

Acknowledgements We thank the Alexander von Humboldt (AvH) Stiftung for supporting our research.

Funding Open Access funding enabled and organized by Projekt DEAL.

References

1. de Leoni, M., van der Aalst, W.M.P., Dees, M.: A general process mining framework for correlating, predicting and clustering dynamic behavior based on event logs. Inf. Syst. **56**(C), 235–257 (2016)
2. Qafari, M.S., van der Aalst, W.M.P.: Case level counterfactual reasoning in process mining. (2021) arXiv preprint arXiv:2102.13490

3. Pearl, J.: Causality, 2nd edn. Cambridge University Press, Cambridge (2009)
4. Peters, J., Janzing, D., Schölkopf, B.: Elements of Causal Inference: Foundations and Learning Algorithms. MIT press, Cambridge (2017)
5. Qafari, M.S., van der Aalst, W.: Root cause analysis in process mining using structural equation models. In: Business Process Management Workshops, pp. 155–167. Springer, Cham (2020)
6. Gupta, N., Anand, K., Sureka, A.: Pariket: Mining business process logs for root cause analysis of anomalous incidents. In: Chu, W., Kikuchi, S., Bhalla, S. (eds.) Databases in Networked Information Systems, pp. 244–263. Springer, Cham (2015)
7. Fani Sani, M., van der Aalst, W., Bolt, A., García-Algarra, J.: Subgroup discovery in process mining. In: Abramowicz, W. (ed.) Business Information Systems, pp. 237–252. Springer, Cham (2017)
8. Mothilal, R.K., Sharma, A., Tan, C.: Explaining machine learning classifiers through diverse counterfactual explanations. In: Proceedings of the 2020 Conference on Fairness, Accountability, and Transparency, pp. 607–617 (2020)
9. Wang, Y., Liang, D., Charlin, L., Blei, D.M.: The deconfounded recommender: A causal inference approach to recommendation. arXiv preprint arXiv:1808.06581 (2018)
10. Hompes, B.F.A., Maaradji, A., La Rosa, M., Dumas, M., Buijs, J.C.A.M., van der Aalst, W.M.P.: Discovering causal factors explaining business process performance variation. In: Dubois, E., Pohl, K. (eds.) Advanced Information Systems Engineering, pp. 177–192. Springer, Cham (2017)
11. Narendra, T., Agarwal, P., Gupta, M., Dechu, S.: Counterfactual reasoning for process optimization using structural causal models. In: Hildebrandt, T., van Dongen, B.F., Röglinger, M., Mendling, J. (eds.) Business Process Management Forum, pp. 91–106. Springer, Cham (2019)
12. Bozorgi, Z.D., Teinemaa, I., Dumas, M., La Rosa, M., Polyvyanyy, A.: Process mining meets causal machine learning: Discovering causal rules from event logs. In: 2020 2nd International Conference on Process Mining (ICPM), pp. 129–136 (2020). IEEE
13. Lehto, T., Hinkka, M.: Discovering business area effects to process mining analysis using clustering and influence analysis. In: International Conference on Business Information Systems, pp. 236–248 (2020). Springer
14. Lehto, T., Hinkka, M., Hollmén, J.: Focusing business improvements using process mining based influence analysis. In: International Conference on Business Process Management, pp. 177–192 (2016). Springer
15. Lehto, T., Hinkka, M., Hollmén, J., *et al.*: Focusing business process lead time improvements using influence analysis. In: SIMPDA, pp. 54–67 (2017)
16. Finch, S.R.: Mathematical Constants. Cambridge University Press, New York (2003)
17. Margaritis, D.: Learning bayesian network model structure from data. Technical report, Carnegie-Mellon Univ Pittsburgh Pa School of Computer Science (2003)
18. Heckerman, D., Geiger, D., Chickering, D.M.: Learning bayesian networks: The combination of knowledge and statistical data. Mach. Learn. **20**(3), 197–243 (1995)
19. Russell, S., Norvig, P.: Artificial Intelligence: A Modern Approach. Springer, Cham (2002)
20. Meek, C.: Graphical models: Selecting causal and statistical models. PhD thesis, Carnegie Mellon University (1997)
21. Cheng, J., Bell, D.A., Liu, W.: An algorithm for bayesian network construction from data. In: Sixth International Workshop on Artificial Intelligence and Statistics, pp. 83–90 (1997). PMLR
22. Spirtes, P., Glymour, C.N., Scheines, R., Heckerman, D.: Causation, Prediction, and Search. MIT press, Cambridge (2000)
23. Verma, T., Pearl, J., et al.: Equivalence and Synthesis of Causal Models. Springer, Cham (1991)
24. Chickering, D.M.: Optimal structure identification with greedy search. J. Mach. Learn. Res. **3**, 507–554 (2002)
25. Ogarrio, J.M., Spirtes, P., Ramsey, J.: A hybrid causal search algorithm for latent variable models. In: Proceedings of Probabilistic Graphical Models-Eighth International Conference, pp. 368–379 (2016)
26. Zhang, J.: On the completeness of orientation rules for causal discovery in the presence of latent confounders and selection bias. Artif. Intell. **172**(16–17), 1873–1896 (2008)
27. Verbeek, H., Buijs, J., Van Dongen, B., van der Aalst, W.M.P.: Prom 6: the process mining toolkit. Proc. BPM Demonstr. Track **615**, 34–39 (2010)
28. Scheines, R., Spirtes, P., Glymour, C., Meek, C., Richardson, T.: The tetrad project: constraint based aids to causal model specification. Multivar. Behav. Res. **33**(1), 65–117 (1998)
29. Ratzer, A.V., Wells, L., Lassen, H.M., Laursen, M., Qvortrup, J.F., Stissing, M.S., Westergaard, M., Christensen, S., Jensen, K.: Cpn tools for editing, simulating, and analysing coloured petri nets. In: van der Aalst, W.M.P., Best, E. (eds.) Applications and Theory of Petri Nets 2003, pp. 450–462. Springer, Berlin, Heidelberg (2003)
30. Frank, E., Hall, M.A., Holmes, G., Kirkby, R., Pfahringer, B., Witten, I.H.: In: Maimon, O., Rokach, L. (eds.) Weka: A Machine Learning Workbench for Data Mining., pp. 1305–1314. Springer, Berlin (2005)
31. Kuhn, M., Johnson, K., et al.: Applied Predictive Modeling, vol. 26. Springer, New York (2013)
32. Buijs, J.: Receipt phase of an environmental permit application process (‘wabo’), coselog project. Eindhoven University of Technology (2014)
33. van Dongen, B.F.: BPI challenge 2017. 4TU.ResearchData. Dataset (2017)
34. van Dongen, B.: BPI challenge 2019. 4TU.ResearchData. Dataset (2019)