ISSN 1995-0802, Lobachevskii Journal of Mathematics, 2020, Vol. 41, No. 12, pp. 2566–2574. © Pleiades Publishing, Ltd., 2020.

Process Mining: Realization and Optimization of Process Discovery Algorithm

G. I. Savin[1*], **A. D. Chopornyak**[1**], **A. A. Rybakov**[1***], **and S. S. Shumilin**[1****]

(Submitted by A. M. Elizarov)

[1]*Joint Supercomputer Center, Scientific Research Institute for System Analysis of the Russian Academy of Sciences, Moscow, 119334 Russia*

Received May 6, 2020; revised May 31, 2020; accepted June 6, 2020

Abstract—The article discusses the approach to the analysis of business processes, called process mining, and the directions of its application. In particular, a description of the process discovery algorithm is provided, with the help of which the process model is reconstructed from the event log in the form of a workflow graph. The implementation and improvement of the algorithm are proposed, which allow avoiding costly enumeration of all subsets of activities included in the analyzed process model. The proposed approach allows the use of process discovery algorithm for large event logs.

DOI: 10.1134/S199508022012032X

Keywords and phrases: *process mining, process discovery algorithm, Petri nets, process model, event logs.*

1. INTRODUCTION

Process mining [1] is a collection of approaches and methods for extracting process related information from event logs. Process mining is closely related to areas such as business process management (BPM) [2] and workflow management (WFM) [3]. WFM technology has historically been primarily aimed at automating business processes from a mechanical point of view, while using BPM approaches, special attention is paid to other aspects in which bottlenecks can be hidden (human factor, organization features, fuzzy rules, and others). Currently, many process-aware information systems (PAIS), in addition to the classic WFM tools, include BPM extensions and process mining approaches. However, often these systems remain analysis and notification systems, being insufficiently integrated into the digital space. Along with the term PAIS [4], the term BPMS (BPM system) or simply PMS (process management system) is also used for such systems.

Any process analysis system works with a process model that is written using one or more notations. These notations represent the essence of the process and are essentially macro-descriptions of directed graphs of special kind. The main objects of process models are activities, states, transitions and subprocesses. Transitions between different activities are also called dependencies. The set of dependencies between activities and also additional conditions imposed on the process model give rise to many possible sequences of state changes, or many scenarios of the process. Depending on the details of the processes, additional attributes of any objects of the process model can be used: time references, duration of activities, dependency logic, resources and strategies for their use, roles of activity performers, and many others. Such a variety that arises in the modeling of processes generates a huge number of methods for analyzing processes and scenarios, and can serve as a source of useful and not always obvious knowledge used to optimize processes.

[*]E-mail: savin@jscc.ru
[**]E-mail: chopor@jscc.ru
[***]E-mail: rybakov.aax@gmail.com
[****]E-mail: shumilin@jscc.com

Process models recorded using formal notation can be used for various purposes: a review from various points of view, discussion of a process model between all participants, preparation of documentation (including in automatic mode), verification (some obvious errors may be detected just by analyzing the topology of the model graph, for example, it can be dead sections of the process, deadlocks, infinite loops and others), performance analysis (simulations of the process model can be used for bottlenecks identification and elimination), visualization of the process to simplify the analysis, specification (from the formal model, descriptions of interfaces and interaction protocols can be generated even before integration into a single digital space), configuration (the description of the process model can be used to configure the system during its implementation). Typically, when developing a process, there is both an informal model (used for discussion and documentation) and a formal model suitable for simulations (used for analysis and improvement).

Informal models are often divorced from life, abstract, idealized, while formal models, on the contrary, are too detailed and often understandable only to the actual developers of the model. There is no middle ground between these two models, and this often leads to additional risks when developing a process model, since in fact the tested and analyzed executable model is very different from the model at the presentations. But even this is not the main problem.

The main problem of any process model is that it is not known in advance how much the model relates to reality. In reality, people are included in the processes, the logic of their interaction with the system is immeasurably complex, and accurate modeling of a person's behavior, his motivation system and other features is impossible in a fixed process model (the process model cannot be rebuild taking into account each individual person). As a result, it may turn out that a superbly worked out and debugged process model is not viable when implemented in a working system. The only way out of this situation can be the presence of feedback between the process taking place in the real world and the process model. Such feedback in terms of the process mining approach is expressed in the registration of events about individual activities and the recording of event data in special repositories called event logs.

Event logs are filled in during the operation and monitoring of a working system and should be used to optimize the process model when re-designing this model [5]. However, at present, the data accumulated during monitoring the functioning of the system are not fully used to analyze the performance of the process model. Very few organizations verify compliance of accumulated data and the process model, and often the reasons for re-designing the process model are not performance issues, but external factors, such as changes in legislation, policies, standards, or changes in external cooperation. Thus, there are risks that the process model may be repeatedly redesigned, but retain the weaknesses and bottlenecks that lead to constant low system performance. There is a kind of vicious circle, the only way out of which is the intellectual analysis of monitoring data (event logs), with subsequent feedback when improving the process model.

Process mining is a new discipline, with the help of which the analysis of the monitoring data of a working system is carried out, on the basis of which you can get knowledge about the actual course of a process, which allows you to work with the process itself, and not with its idealized model.

While applying process mining technology, the business process model interacts with a working information system, the outside world and accumulated data. The process model is transformed into a process implemented within the information system through which interaction with the outside world is carried out. At the same time, events coming from the outside world through an information system are recorded in event logs. The main application of process mining arises when analyzing the interaction of a process model and accumulated data.

There are three main areas of process mining. The first direction is the construction of a process model based on accumulated data (process discovery) [6]. It consists in the analysis of the actual accumulated data on the course of the real process (fixed chains of data flows, a log of the interaction of process participants with each other, message passing between process participants) and the construction of a process model based on it without using information about the designed idealized model. The second area is called conformance testing [7]. The purpose of this type of analysis is to verify the conformity of the developed model and the event logs related to this model. When conducting this check, it is possible to detect regions in the process model that are not sufficiently covered by data from event logs or regions with insufficiently clear logic, within which the course of the real process slows down due to the additional costs of overcoming these difficulties. For example, in the place of a fuzzy description of the logic of the process, additional coordination of some action that is not reflected in the process model may be required. In this case, the model can demonstrate high performance, however, with real flow, the process will slow down or even block. The third area or process mining is process improvement based on analysis (process enhancement) [8]. If checking the conformity of the process model and its data helps to find places where the process model diverges from reality, then in this direction it is possible to develop recommendations on how to change the process model in order to reduce this discrepancy. Improvements may relate to the modification of the process model to more accurately reflect reality, and may cause the model to expand if the logic of the existing model is too meager to reflect the complex interactions of the process participants.

It is worth noting that the logic of the process, data flows, interaction within the process, messaging are the most simple aspects that affect process mining. However, technology is not limited to these aspects. Resource allocation, planning issues, decision rules, complex interaction logic, probabilistic events all of these aspects also apply to process mining and can be analyzed using the tools of this technology [9].

This article discusses an algorithm for constructing a real process based on data from an event log and discusses ways to optimize this algorithm.

2. SIMPLE PROCESS DISCOVERY ALGORITHM

In this article we will consider the description of the process model using workflow nets (WFN) [10]. WFN reflects only the sequence of activities during the flow of the process scenario and does not affect other aspects of the analysis. WFN is a directed graph whose edges are capable of transmitting tokens that reflect the course of the scenario.

There are two types of WFN nodes. The nodes of the first type are tokens places (on the diagrams of process models are shown below with white rectangles, place labels are denoted $p0$, $p1$, etc.). The nodes of the second type represent the activities of the process (in the diagrams of the process models are shown below with black rectangles, the activity labels are named a, b, etc.). Two token places are specially emphasized in WFN: I is the global input from which the scenario starts, O is the global output where the scenario ends (these special places are shown by gray rectangles on the diagrams of process models).

WFN edges can connect only nodes of different types (that is, only activity can be a predecessor or successor of a token place, only a token place can be a predecessor or successor of an activity). During the course of the process, some activity can be performed only if all the predecessor token places of this node contain tokens. After performing some activity in WFN, tokens from the predecessor nodes are deleted, after which new tokens are created, which are distributed along all output edges and are placed in all successor nodes of this activity.

Consider the main local patterns of the process that can be displayed using WFN (Fig. 1).

Figure 1a shows the usual sequence of activities $a \rightarrow b$. This template prescribes that after the a activity is completed, the token will be placed in the place of $p0$. After that, it turns out that the condition b is fulfilled that all its predecessors contain tokens (since the only predecessor b is $p0$). Thus, the b activity will be performed, the token will be removed from the place $p0$, and new tokens will be sent along the output edges to the nodes-successors of the b activity.

Figure 1b shows an XOR-split template that implements the selection of one of the activities after the execution of the a activity. After the activity a has been completed, the place $p0$ is occupied by a token, which can be used when only one of the activities b or c is performed. Thus, it is possible to execute either a sequence of activities $a \rightarrow b$ or a sequence $a \rightarrow c$.

Figure 1c shows the XOR-join template, which by analogy implements either a sequence of activities $a \rightarrow c$ or a sequence of activities $b \rightarrow c$.

Figure 1d shows the AND-split pattern, which implements the execution of both b and c after the a activity. As soon as the a activity is completed, tokens fall into the places $p0$ and $p1$, which make it possible to execute the b and c activities. In this case, the order in which these activities will be performed is not important.

Figure 1e shows the AND-join pattern, demonstrating that the activity c can be performed only after the execution of both activities a and b. The c activity requires for its execution the presence of tokens immediately in $p0$ and $p1$, and this can happen only after the execution of both a and b activities. The execution order of the a and b activities is not important in this case.

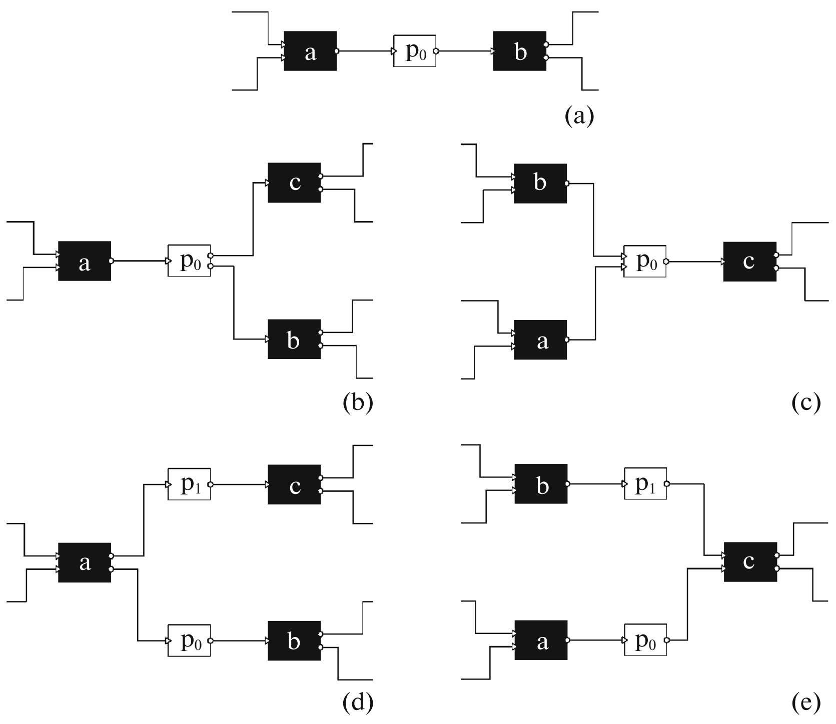

Fig. 1. Process patterns in WFN model. (a) Sequence, (b) XOR-split, (c) XOR-join, (d) AND-split, (e) AND-join.

After describing this simplified process model using WFN, we can proceed directly to the algorithm for constructing the model based on the event log. Process discovery algorithm description is based on fundamental work of W.M. van der Aalst [1] in scope of process mining. In this case, an event log is simply a list of scenarios, each of which is an ordered sequence of activities. According to the provided event log, it is required to build a process model described using WFN that implements all the scenarios from this log.

We will consider simple scenarios in which it will be seen how strongly the event log entries affect the real process model. For example, let us have an ideal model of a process, which consists of a fixed sequence of activities that must be performed in a strictly defined sequence. Here is the sequence of activities: $a \to b \to c \to d \to e \to f$. This sequence of activities corresponds to the following process model shown in the Fig. 2.

Even if all participants in the process agree with such a specific sequence of activities that is necessary for the scenario to proceed, then in reality situations usually happen when the process proceeds differently. Imagine a simple situation when, due to rush or force majeure, all intermediate activities in the scenario execution were omitted and the entry $a \to f$ appeared in the event log. Figure 3 shows a new process model that also reflects this simplified scenario.

In this figure, we already see the emergence of the XOR-split and XOR-join patterns, and in general the process model has become much more complicated.

And now let's look at an even more extended event log, in which there are scenarios with other deviations. In particular, we consider a scenario with a violation of the order of activities, a partial scenario, and a scenario with the repeated execution of certain sequences of activities. Let's collect the full event log, it will look like this:

$$a \to b \to c \to d \to e \to f,$$
$$a \to f,$$
$$a \to b \to d \to c \to e \to f,$$
$$a \to b \to c,$$
$$a \to b \to c \to d \to e \to b \to c \to d \to e \to f.$$

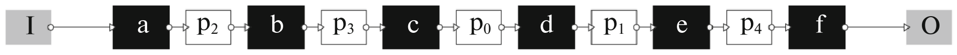

Fig. 2. Process model that build from one scenario $a \to b \to c \to d \to e \to f$.

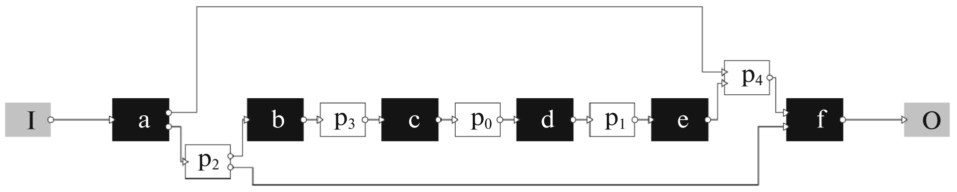

Fig. 3. Adjusted process model after processing the scenario $a \to f$.

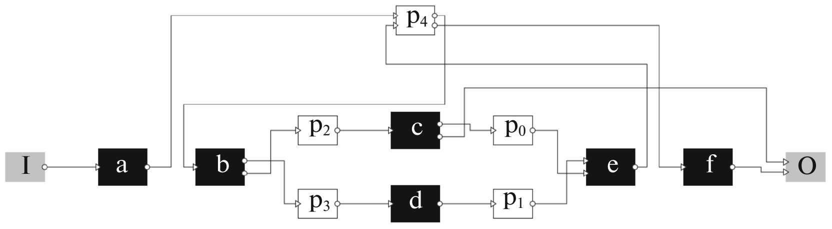

Fig. 4. A process model based on an event log containing distorted scenarios.

When building a process model from the presented event log, we get the circuit shown in the Fig. 4.

This model of the process, of course, does not look like the ideal model from the Fig. 2. It is also obvious that any analysis should be carried out using this real model, and not an ideal one. The construction of a real model from the event log can only be performed automatically, since any distortion of real scenarios significantly affects the topology of the model. We also note that the above examples show completely trivial scenarios that do not come close to the complexities of real-life business processes. Therefore, to build models of real processes for real event logs, efficient algorithms are required.

Let us describe the simplest algorithm for constructing a process model from an event log. In the process of describing the algorithm, we will immediately show its results on the already considered model event log (we denote it by L).

So we have an event log

$$L = [a \to b \to c \to d \to e \to f, \quad a \to f,$$
$$a \to b \to d \to c \to e \to f, \quad a \to b \to c,$$
$$a \to b \to c \to d \to e \to b \to c \to d \to e \to f].$$

First, we need to determine the set of all activities present in the event log (we denote it A). In addition, we need to define the set of activities with which the scenarios from the event log start ($A_I \subset A$), as well as the activities that end the scenarios from the event log ($A_O \subset A$):

$$A = \{a, b, c, d, e, f\}, \quad A_I = \{a\}, \quad A_O = \{c, f\}.$$

The next step requires the construction of a matrix of activity sequences (M_S). This matrix is square; it has size $|A| \times |A|$, and its element $M_S[x, y]$ contains 1 if there is a scenario in the event log that contains the sequence $x \to y$.

The M_S matrix for our test event log is as follows:

$$M_S = \left|\begin{array}{ccccccc} & [a] & [b] & [c] & [d] & [e] & [f] \\ [a] & 0 & 1 & 0 & 0 & 0 & 1 \\ [b] & 0 & 0 & 1 & 1 & 0 & 0 \\ [c] & 0 & 0 & 0 & 1 & 1 & 0 \\ [d] & 0 & 0 & 1 & 0 & 1 & 0 \\ [e] & 0 & 1 & 0 & 0 & 0 & 1 \\ [f] & 0 & 0 & 0 & 0 & 0 & 0 \end{array}\right|.$$

Next, the relationship matrix M_R is needed to be constructed. It has the same size as the M_S matrix, and its elements determine the relationships between activities that should be reflected in the constructed process model. The strict follow relationship between the x and y activities (we will denote it by $x \Rightarrow y$) means that in the event log there is a scenario in which $x \to y$, but there is no such scenario in which $y \to x$. In other words, the x and y activities are related by a strict follow relationship if $M_S[x, y](1 - M_S[y, x]) = 1$. Similarly, the activities of x and y can be called parallel, if in some scenarios x may follow y, and in other scenarios y may follow x. The condition for parallel activities is $M_S[x, y]M_S[y, x] = 1$ (designation $x \mid\mid y$). If, in no scenarios, the sequences $x \to y$ or $y \to x$ are detected, then the relation between such activities will be called uncertainty. The uncertainty condition is $(1 - M_S[x, y])(1 - M_S[y, x]) = 1$ (designation $x\ ?\ y$).

For the considered event log, the M_R matrix is as follows:

$$M_R = \left|\begin{array}{ccccccc} & [a] & [b] & [c] & [d] & [e] & [f] \\ [a] & ? & \Rightarrow & ? & ? & ? & \Rightarrow \\ [b] & \Leftarrow & ? & \Rightarrow & \Rightarrow & \Leftarrow & ? \\ [c] & ? & \Leftarrow & ? & \mid\mid & \Rightarrow & ? \\ [d] & ? & \Leftarrow & \mid\mid & ? & \Rightarrow & ? \\ [e] & ? & \Rightarrow & \Leftarrow & \Leftarrow & ? & \Rightarrow \\ [f] & \Leftarrow & ? & ? & ? & \Leftarrow & ? \end{array}\right|. \tag{1}$$

After constructing a matrix of relations between activities, we can proceed to constructing set of token places in the process model (P). A token place must be built for each such pair of disjoint subsets of activities $X \subset A$, $Y \subset A$, which satisfies two conditions. Firstly, any two activities from X are related by the uncertainty relation, also any two activities from Y are connected by the uncertainty relation. Secondly, any activity $x \in X$ is associated with any activity $y \in Y$ by the relation $x \Rightarrow y$. In this case, the token place $p(X, Y)$ is built, all the activities from X become its predecessor nodes, and all activities from Y become the successor nodes.

In our case under consideration, the set of token places are as follows:

$$P = \{p(\{d\},\{e\}), p(\{c\},\{e\}), p(\{e\},\{f\}), p(\{e\},\{b\}), p(\{e\},\{b,f\}), \\ p(\{a\},\{f\}), p(\{a\},\{b\}), p(\{a\},\{b,f\}), p(\{a,e\},\{f\}), p(\{a,e\},\{b\}), \\ p(\{a,e\},\{b,f\}), p(\{b\},\{d\}), p(\{b\},\{c\})\}. \tag{2}$$

The final step in the formation of the WFN, which is a model of the process, is removal of extra token places. If there are two places $p(X, Y) \in P$ and $p(X', Y') \in P$ in the set of token places such that $X' \subseteq X$ and $Y' \subseteq Y$, then the place $p(X', Y')$ is redundant and can be removed from the model. After removing excess places, the final set of token places will take its final form

$$P = \{p(\{d\},\{e\}), p(\{c\},\{e\}), p(\{a,e\},\{b,f\}), p(\{b\},\{d\}), p(\{b\},\{c\})\}. \tag{3}$$

The set of given token places and the construction of all edges between token places and activities leads us to the process model shown in the Fig. 4.

3. SIMPLE PROCESS DISCOVERY ALGORITHM OPTIMIZATION

In the algorithm presented in the previous section, the bottleneck is the construction of token places set P with the subsequent removal of redundancies. As the total number of activities increases, the number of possible combinations of subsets of activities for the formation of token places increases exponentially. Also, most of the built token places subsequently still have to be removed. For example, with the constructed token place $p(\{a,b\},\{c,d\})$, the following excess token places will be necessarily included into the set P: $p(\{a\},\{c\})$, $p(\{a\},\{d\})$, $p(\{b\},\{c\})$, $p(\{b\},\{d\})$. In the future, these excess token places will be deleted. But to search for redundant token places, it is also necessary to spend computational time.

We will propose an algorithm for constructing token places in which only the necessary subsets of activities are checked, and control over redundant token places is performed automatically.

First, we introduce the relations between the sets of activities as follows. Let X and Y be two sets of activities. We say that $X\ ?\ Y$ if for any two activities $x \in X$, $y \in Y$ the relation $x\ ?\ y$ is correct. We will also say that $X \Rightarrow Y$ if for any two activities $x \in X$, $y \in Y$ the relation $x \Rightarrow y$ holds.

We introduce the concept of a graph of relations between sets of activities. This graph will contain directed edges, as well as relations between these edges. Let A be the set of activities, between the elements of which the relations $x \Rightarrow y$ and $x\ ?\ y$ are introduced (the relation $x \,||\, y$ in this case does not interest us). A relationship graph is a graph $G = G(N, \overrightarrow{E}, \Xi)$. Elements of the set of nodes N of a given graph are subsets $X \subseteq A$ such that any two activities $a \in X$, $b \in X$ of this subset are related by the relation $a\ ?\ b$. Elements of the set of directed edges $\overrightarrow{E}$ are ordered pairs of (X,Y) elements from N such that $X \Rightarrow Y$ (in the pictures below, these edges will be represented by solid black arrows). Elements of the set Ξ are such ordered pairs $(\overrightarrow{e}', \overrightarrow{e})$, where $\overrightarrow{e}' = (X', Y') \in \overrightarrow{E}$, $\overrightarrow{e} = (X,Y) \in \overrightarrow{E}$ and $X' \subseteq X$, $Y' \subseteq Y$.

If for a given set of activities A such a graph G is constructed, then the token places that were required to be found within the framework of the process discovery algorithm are those token places $p(X,Y)$ for which $(X,Y) \in \overrightarrow{E}$, and also there is no element $\overrightarrow{e} \in \overrightarrow{E}$ such that $((X,Y), \overrightarrow{e}) \in \Xi$. Note that in the end we are interested in the edges of the graph G, so that at any stage of constructing this graph, isolated vertices can be removed from it without specifying it specially (isolated vertices of the graph G do not affect the operation of the algorithm).

Consider the construction of the required graph G in an iterative way. In the process of constructing the graph G, we allow it to contain undirected edges that are unordered pairs of $\{X,Y\}$ elements from N such that $X\ ?\ Y$. We denote the set of these edges by $\overline{E}$ (in the pictures below, these edges will be depicted by dashed lines). Thus, during the construction process, we will consider the graph $G(N, \overline{E}, \overrightarrow{E}, \Xi)$.

At the beginning, we create the first version of the graph G_0, which we will use for the subsequent transformations to translate into the required form G. So, we define the graph $G_0 = G_0(N_0, \overline{E}_0, \overrightarrow{E}_0, \Xi_0)$ as follows:

$$\begin{cases} N_0 = \{\{x\} \mid x \in A\}, \\ \overline{E}_0 = \{\{X,Y\} \mid X \in N_0, Y \in N_0, x\ ?\ y\}, \\ \overrightarrow{E}_0 = \{(X,Y) \mid X \in N_0, Y \in N_0, x \ \Rightarrow\ y\}, \\ \Xi_0 = \emptyset. \end{cases}$$

Figure 5 shows the graph G_0 constructed for the matrix M_R from (1).

After constructing the graph G_0, we will sequentially expand it using several separate steps. At each step, an arbitrary edge $\overline{e} = \{X,Y\} \in \overline{E}$ is taken and the following actions are performed for it. A new element $X \cup Y$ is added to the set N. For each element $Z \in N$ such that $\{Z,X\} \in \overline{E}$, $\{Z,Y\} \in \overline{E}$ new element $\{Z, X \cup Y\}$ is added into the set of non-oriented edges $\overline{E}$. For each element $Z \in N$ such that

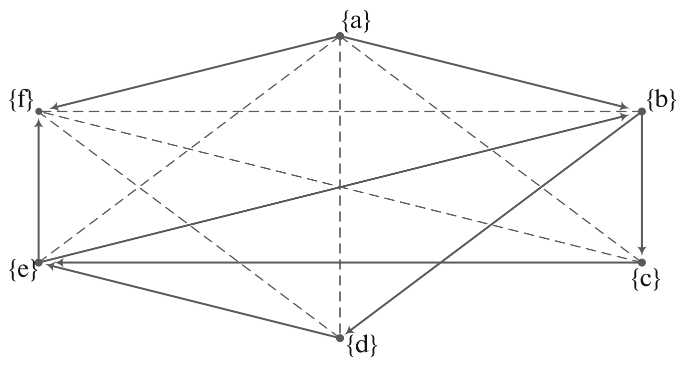

Fig. 5. Graph G_0 constructed for test event log.

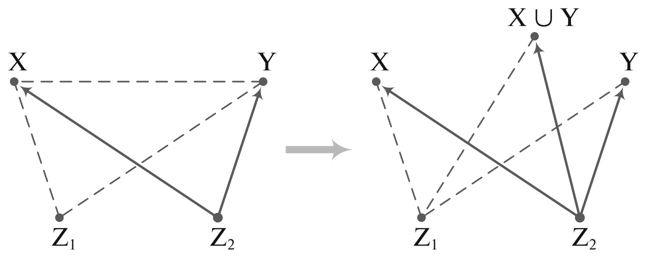

Fig. 6. One step of the graph G expansion.

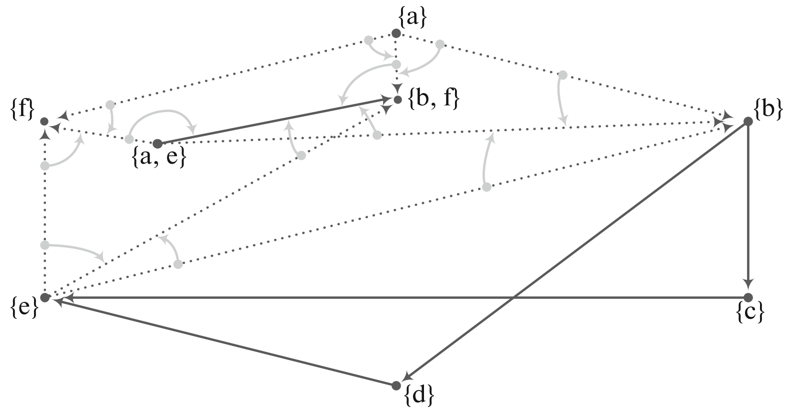

Fig. 7. Final form of graph G.

$(Z, X) \in \overrightarrow{E}$, $(Z, Y) \in \overrightarrow{E}$ new element $(Z, X \cup Y)$ is added into the set of oriented edges $\overrightarrow{E}$, and the elements $((Z, X), (Z, X \cup Y))$ and $((Z, Y), (Z, X \cup Y))$ are added to the set Ξ. The edge $\overline{e}$ is removed from the set $\overline{E}$. An illustration of the execution step of the graph G expansion is presented in Fig. 6.

The described steps for expanding the graph G are performed until the set $\overline{E}$ becomes empty. It is worth noting that the number of elements in this set does not have to decrease at each step of the graph extension (for example, in Fig. 6 the number of edges from $\overline{E}$ has not changed). However, if at particular step of graph G expansion the number of edges in $\overline{E}$ has not decreased, then some complete graph composed of edges from $\overline{E}$ necessarily must be destroyed (in Fig. 6 the triangle XYZ_1 was destroyed), instead of which a complete graph of only a smaller order can appear. Thus, after a finite number of expansion steps, the set $\overline{E}$ will become empty and the final graph G will be generated.

Figure 7 shows the final form of the graph G for considered example. Elements from the set Ξ in this figure are shown by gray arcs, and redundant edges from $\overrightarrow{E}$ are shown by dotted lines. With the construction of the graph of relations G described, it is not difficult to detect all redundant edges from $\overrightarrow{E}$. Thus, the desired set of edges from $\overrightarrow{E}$ is shown by solid black arrows in the figure, which coincides with the set from (3).

The disadvantage of the algorithm is that redundant edges cannot be removed during the expansion steps of the graph G, since any intermediate edge can participate when adding new edges. To eliminate this drawback, additional research and improvement of this algorithm is required, which is beyond the scope of this article.

4. CONCLUSION

The considered approach to the implementation of the process discovery algorithm allows it to be applied to large-size event logs. Building a relationship graph with subsequent expansion allows to get rid of enumeration of all subsets of activities involved in the analyzed process. In fact, when constructing a relationship graph, the branch and bound method is implicitly used, in which only those subsets of the vertices of the relationship graph are considered, each pair of which is connected by the relation $x\ ?\ y$. Of particular interest is the order of traversing the edges $\overline{E}$ of the relationship graph G during the application of the algorithm, since the construction speed of the graph G depends on it. This aspect deserves attention, since it would be useful to get rid of the need to save all relations between edges from $\overrightarrow{E}$ until the end of the algorithm work.

FUNDING

The work has been done at the JSCC RAS as part of the state assignment for the topic 0065-2019-0016 (reg. no. AAAA-A19-119011590098-8). The supercomputer MVS-10P, located at the JSCC RAS, was used for calculations during the research.

REFERENCES

1. W. M. van der Aalst, *Process Mining. Discovery, Conformance and Enhancement of Business Processes* (Springer, New York, 2011).
2. J. Jeston and J. Nelis, *Business Process Management. Practical Guidelines to Successful Implementations* (Routledge, London, New York, 2014).
3. K. van Hee, *Workflow Management: Models, Methods, and Systems (Information Systems)* (MIT Press, Boston, 2004).
4. M. Dumas, W. M. van der Aalst, and A. H. ter Hofstede, *Process Aware Information Systems: Bridging People and Software Through Process Technology* (Wiley-Interscience, New York, 2005).
5. S. L. Mansar and H. A. Reijers, "Best practices in business process redesign: Validation of a redesign framework," Comput. Ind. **56**, 457–471 (2005).
6. M. Dumas, M. La Rosa, J. Mendling, and H. A. Reijers, *Fundamentals of Business Process Management* (Springer, New York, 2018).
7. A. Rozinat and W. M. van der Aalst, "Conformance checking of process based on monitoring real behavior," Inform. Syst. **33**, 64–95 (2008).
8. A. Burattin, A. Sperduti, and M. Veluscek, "Business models enhancement through discovery of roles," in *Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining* (2013), pp. 103–110.
9. J. Mendling, *Business Process Management. Metrics for Process Models* (Springer, Cham, 2008), pp. 1–15.
10. R. Agrawal, D. Gunopulos, and F. Leymann, "Mining process models from workflow logs," in *Proceedings of the 6th International Conference on Extended Database Technology* (1998), pp. 469–483.