DOI: 10.15514/ISPRAS-2026-38(4)-27

Synthesis of Acyclic Models for Processes Without Repeating Events

A.K. Joulitov, ORCID: 0009-0001-0507-7408 <akzhulitov@edu.hse.ru>
I.A. Lomazova, ORCID: 0000-0002-9420-3751 <ilomazova@hse.ru>

HSE University,
11, Pokrovsky Bulvar, Moscow, 101000, Russia.

Abstract. In process mining, DFG (Directly-Follows Graph) models are popular due to their simplicity and clarity. However, if a process is acyclic but contains concurrent events, standard algorithms for discovering DFG models can generate "fake" cycles that do not actually exist in the event log. These cycles hinder the analysis of information processes, significantly reducing the quality and precision of the model. This problem was studied by N. Shaimov et al., where it was proposed to discover DFG models without false cycles by duplicating graph vertices. This problem does not have a single or best solution, and the solution proposed there is heuristic. The aim of this paper is to propose an alternative algorithm for discovering acyclic DFG models for processes without repeating events and to compare it with the existing one on real-life and artificial data. The method presented in this paper produces a smaller model than existing solution and is stable when constructing models of highly concurrent processes. It is also proved that eliminating false cycles for highly concurrent processes leads to an exponential increase in the model size.

Keywords: process mining; automatic model synthesis; directly follows graphs; acyclic models.

For citation: Joulitov A.K., Lomazova I.A. Synthesis of Acyclic Models for Processes Without Repeating Events. Trudy ISP RAN/Proc. ISP RAS, vol. 38, issue 4, part 2, 2026, pp. 215-224. DOI: 10.15514/ISPRAS-2026-38(4)-27.

Acknowledgements. This article is an output of a research project (HSE-BR-2025-024) implemented as a part of the Basic Research Program at HSE University.

Синтез ациклических моделей для процессов без повторяющихся событий

А.К. Жулитов, ORCID: 0009-0001-0507-7408 <akzhulitov@edu.hse.ru>
И.А. Ломазова, ORCID: 0000-0002-9420-3751 <ilomazova@hse.ru>

Национальный исследовательский университет «Высшая школа экономики»,
Россия, 101000, г. Москва, Покровский бульвар, д. 11.

Аннотация. В майнинге процессов (process mining) графы непосредственного следования (Directly-Follows Graph, DFG) популярны благодаря своей простоте и наглядности. Однако, если процесс является ациклическим, но содержит параллельные события, стандартные алгоритмы построения DFG-моделей могут генерировать «ложные» циклы, которые не представлены в журнале событий. Такие циклы мешают анализу информационных процессов, значительно снижая интерпретируемость и точность (precision) модели. Эта проблема рассматривалась в работе Н. Шаимова и др., где было предложено синтезировать DFG-модели без ложных циклов с помощью дублирования вершин графа. Задача эта не имеет единственного или лучшего решения, и предложенное ранее решение является эвристическим. Цель данной статьи – предложить альтернативный алгоритм синтеза ациклических DFG-моделей для процессов без повторяющихся событий и сравнить его с существующим на реальных и искусственных данных. Представленный в этой статье метод позволяет получить модель меньшего размера по сравнению с существующим решением, а также стабильно ведёт себя при построении моделей процессов с высокой степенью параллелизма. Также в работе доказано, что устранение ложных циклов ведет к экспоненциальному увеличению размера моделей для процессов с высокой степенью параллелизма.

Ключевые слова: майнинг процессов; автоматический синтез моделей; графы непосредственного следования; ациклические модели.

Для цитирования: Жулитов А.К., Ломазова И.А. Синтез ациклических моделей для процессов без повторяющихся событий. Труды ИСП РАН, том 38, вып. 4, часть 2, 2026 г., стр. 215–224 (на а нглийском языке). DOI: 10.15514/ISPRAS-2026-38(4)-27.

Благодарности. Исследование осуществлено в рамках Программы фундаментальных исследований НИУ ВШЭ (HSE-BR-2025-024).

1. Introduction

Process mining comprises methods and approaches for analyzing and improving processes in information systems based on event log data. The emergence and development of process mining as a discipline was driven by the availability of detailed business process execution data. Almost all actions performed on digital systems are recorded in event logs, which form a history of the process. The main goal of process mining is to extract process knowledge from massive event logs, overcoming the limitations of traditional analysis methods based on subjective descriptions or idealized models.

DFG (Directly-Follows Graph) models are the most popular formalism used in many industrial tools such as PM4Py [1], Celonis [2], and Disco [3], among others. Nevertheless, known discovery algorithms produce models that exhibit extraneous behavior, which hinders process analysis.

Acyclic processes, whose logs represent traces of non-recurring events, can be distinguished as a specific subclass. The appearance of cycles in a DFG model of an acyclic process is due to the use of sequential (interleaving) semantics for concurrency. In DFG models an edge exists between event nodes a and b if and only if there is a trace in the log where these events occur consecutively. But what if events a and b are independent and can occur in the log in any order? In this case, standard algorithms map them to unique nodes a and b and create bidirectional edges, forming "false" cycles [4]. These false cycles severely hinder process analysis by suggesting looping behaviors that do not actually exist in the log. More complex formalisms, such as BPMN and Petri nets, support

concurrency, meaning that such process models do not generate extraneous cycles absent from the log. However, these models may be more intricate and difficult for users to analyze.

In process discovery, there is a fundamental and often mutually exclusive trade-off between two quality dimensions: *precision* and *simplicity*. Precision measures how much extraneous behavior (not seen in the log) the model restricts. Simplicity reflects the model's readability, usually measured by its size (the number of nodes and edges). If we try to create a highly detailed model that captures exact behavior (increasing precision), we usually have to duplicate nodes extensively, causing the model size to grow and simplicity to drop. Conversely, if we heavily generalize the model by merging nodes (improving simplicity), false cycles appear, which generates non-existent behavior and severely worsens precision.

To ensure that a DFG model does not contain cycles and that precision is improved, we use node duplication. This task was recently researched in [5]. They generalized DFG models by allowing duplicated nodes and developed an algorithm based on two stages: log partitioning and merging of log parts. Later we will see that the first stage works well on regular logs and use it as the base for our algorithm. We will also develop a more precise merging approach that allows us to maintain more specific invariants in the model.

Consider the example in Fig.1, where the log L consists of two traces $\langle start, a, b, end\rangle$ and $\langle start, b, a, end\rangle$, i.e., events a and b are concurrent and independent. In this log, each event is executed once, yet the DFG model generated for L by the standard algorithm will contain a cycle with nodes a and b. This is a DFG artifact; it does not occur in BPMN model synthesis, due to the presence of the AND-gate. Note also that although the third model in Fig.1 does not contain cycles, it does contain a path where event b occurs twice, while all events in the traces in L were unique. In this case, an absolutely accurate model can only be built by reproducing the entire log, and this is not what we want. Therefore, it is necessary to balance model size and precision.

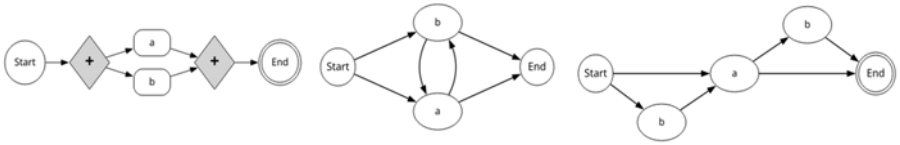

Fig. 1: BPMN model, standard DFG model, and cycle-free DFG model synthesized for log L.

The main objective of this paper is to propose a novel alternative algorithm for synthesizing acyclic DFG models and to compare it with the existing heuristic from [5]. To improve accuracy, we additionally require that the synthesized model does not generate not only cycles, but also paths with repeating events.

In this paper, we provide a theoretical exponential bound (relative to the number of events) on the size of a node-minimal DFG model without repeats for a highly concurrent process, whereas a cycle-free model that allows repeating events in its runs can be of polynomial size. Based on this, we developed a new iterative synthesis algorithm. In an experimental evaluation, we show that for concurrent logs, our algorithm produces a smaller model compared to [5]. This is a significant advantage: an exponential increase in model size is only a theoretical worst-case scenario, while in practice we get a model of comparable size but better quality.

2. Literature Review

Representing concurrency in DFG models using standard algorithms is a well-known problem [4], as it creates cycles even if the process is acyclic. Despite this, DFG models are widely used due to their simplicity. Other formalisms (Petri nets, BPMN) explicitly represent parallelism, but have a more complex structure. Some synthesis algorithms, for example, Split Miner [6], use DFGs in the initial stages to ultimately produce a model that supports concurrency.

Discovery algorithms typically construct models in which each event is represented by at most one node, but models with duplicate events have also been considered in the literature. For example, in [7], BPMN models with repeating event nodes are used for exploratory data analysis.

Repeating nodes simplify the model and are one way to reflect parallelism. In [5], repeated event nodes are used to construct cycle-free DFG models by log splitting and iteratively merging acyclic DFG models. Process mining is highly relevant in theoretical computer science, sitting at the intersection of business process research and data analysis. It is applied in business, healthcare, manufacturing, and education.

Representing concurrency in DFG models using standard algorithms is a well-known problem [4], as it creates cycles even if the process is acyclic. Despite this, DFG models are widely used due to their simplicity. Other formalisms (Petri nets, BPMN) explicitly represent parallelism, but they have a more complex structure. Some synthesis algorithms, for example, Split Miner [6], use DFGs in the initial stages to ultimately produce a model that supports parallelism.

Synthesis algorithms typically construct models in which each event is represented by at most one node, but models with duplicate events have also been considered in the literature. For example, in [7], BPMN models with repeating event nodes are used for exploratory data analysis. Repeated nodes simplify the model and are a way to reflect concurrency. In [5], repeated event nodes are used to construct cycle-free DFG models via log splitting and iterative merging of acyclic DFG models.

3. Basic Definitions and Problem Statement

The definitions hear will be similar to those used in [5] For a set X, let X^* denote the set of all finite sequences (words) over X.

Definition 1. Let E be a finite set of event names. A *trace* over E is a finite sequence $\sigma = \langle e_1, \dots, e_n\rangle$ of events from E, i.e., $\sigma \in E^*$. An *event log* L over E is a finite multiset of traces. A *sublog* of L is a log $L' \subseteq L$. An event e repeats in a trace $\sigma = \langle \sigma_1, \dots, \sigma_k\rangle$ if there exist indices $i \neq j$ such that $\sigma_i = \sigma_j$. A log L *contains no repeating events* if every trace $\sigma \in L$ has no repeating events.

Definition 2. Event b *directly follows* event a in L (denoted $a < b$) if there exists a trace $\sigma = \langle e_1, \dots, e_n\rangle \in L$ such that $e_i = a$ and $e_{i+1} = b$ for some $i \in \{1, \dots, n-1\}$.

Definition 3. A *Directly-Follows Graph (DFG)* over E is a node-labeled graph $G = (V, A, \lambda, v_{start}, v_{end})$, where:

- V is a finite set of nodes;
- $A \subseteq V \times V$ is a finite set of edges;
- $\lambda: V \setminus \{v_{start}, v_{end}\} \to E$ is a labeling function assigning an event name to each node;
- $v_{start} \in V$ is the unique start node with no incoming edges;
- $v_{end} \in V$ is the unique end node with no outgoing edges.

The *size* of a DFG is defined as the number of nodes $|V|$.

Definition 4. Let $G = (V, A, \lambda, v_{start}, v_{end})$ be a DFG over E. A sequence $r = \langle e_1, \dots, e_n\rangle \in E^*$ is a *run* in G if there is a path $\langle v_{start}, v_1, \dots, v_n, v_{end}\rangle$ in G such that $\lambda(v_1) = e_1, \dots, \lambda(v_n) = e_n$.

Definition 5. A DFG model G *perfectly fits* an event log L if every trace $\sigma \in L$ is a run in G.

4. Problem Statement

Let E be a set of events, and L be an event log over E without repeating events. We aim to construct a minimal by nodes DFG G such that:

- G perfectly fits the event log L.
- Every run in G contains no repeating events (which implies G contains no cycles).

In process discovery, a fundamental trade-off exists between the precision of a model and its simplicity, which is frequently quantified by the size of the graph [6-7]. Striving to capture system behavior as accurately as possible–without allowing spurious execution sequences–often leads to substantial model expansion (e.g., through node duplication), thereby complicating its visual analysis. Conversely, overly compact graphs are prone to overgeneralization (the underfitting effect): they allow for event sequences that were absent from the original event log, thus sacrificing precision for the sake of interpretability [7]. Consequently, striking an optimal balance between model size and precision remains one of the key challenges in the analysis of real-world business processes.

Therefore, we maintain strict requirements for the perfect fitting and acyclicity of a model, and we try to synthesize a model with the minimal possible size to achieve maximal simplicity.

5. Algorithm Description

Our approach consists of two main steps:

- Log partitioning (taken from [5]).
- Iterative merging.

5.1 Step 1: Log Partitioning

In the first phase of the algorithm, we partition the initial event log into several sub-logs such that applying a standard DFG discovery algorithm to each sub-log produces an acyclic model.

To achieve this, we rely on the fundamental property that a cycle emerges in a DFG if and only if there exist at least two traces whose combination in DFG produce a cycle. Therefore, we can construct an undirected conflict graph where each vertex represents a single trace from the log, and an edge connects two traces if their combination produces a cycle.

Therefore, dividing the event log into valid cycle-free sub-logs is equivalent to partitioning the vertices of this conflict graph into the minimum number of independent sets (anti-cliques). By taking the complement of this conflict graph, the task translates to the well-known Minimum Clique Cover problem. Since finding an optimal clique cover is an NP-hard problem, we employ a heuristic approach to compute a near-optimal log partitioning.

5.2 Step 2: Iterative Merging

In the second phase, the algorithm iteratively merges the acyclic DFGs generated from the sub-logs by pairs. Let DFG A (the accumulated sub-logs graph) and DFG B (the current sub-log graph) be two acyclic DFGs. Our objective is to synthesize a unified, cycle-free model that contains both Model A and Model B, as valid subgraphs. This merging process is executed through the following procedural steps:

1. **Candidate pool initialization**

   Initially, for every event label present in B, we identify a pool of "candidate" vertices in A that share the same label. The candidate must ensure that the route of model B can be continued through model A without violating the restrictions.

2. **Topological sorting**

   Next, we compute a topological sort of the vertices in B. The basic idea is to iteratively, layer by layer, build traces of B into A. With a topological ordering, we guaranty that before constructing any new edge corresponding to a target vertex, the entire trace prefix has already been successfully constructed in the merged model.

3. **Mapping Construction**

   We construct a mapping function $F: V_B \to V_A$, ensuring that every trace originating in B is structurally mapped to a valid path in Model A. The *start* vertex of B is mapped to the *start* vertex of A. Iterating through the vertices of B in topological order, let $v \in V_B$ be the current vertex being processed. We attempt to map v to an optimal candidate in A, such that drawing directed edges from the already-mapped predecessors of v to this candidate keep all constraints. The selection of the optimal candidate among multiple valid options is guided by specific heuristics (detailed below).

   If no existing candidate satisfies the constraints, the algorithm spawns a new duplicate vertex with the required label in A. After completing the topological traversal of V_B, the algorithm returns a structurally correct, perfectly fitting, and strictly acyclic union of A and B.

We can also skip the log splitting step, and connect traces one by one, and then we will get an algorithm that will handle complex logs well, where splitting divides the log into many parts.

6. Heuristics

To optimize the mapping process during Step 3 and keep the resulting model as compact as possible, the algorithm uses two types of heuristic: hard restrictions and scoring, the algorithm takes the vertex, which satisfies the restrictions and has the maximal score:

- **Hard Constraints (Cycle and Dead-end Prevention)**: Before scoring, a candidate is immediately rejected if connecting to it creates a topological cycle in Model A, or if choosing it leads to a "dead end" (a state where the descendants of v will have no valid candidates to map to in the future).
- **Soft Scoring (Edge Reuse Maximization)**: To prevent unnecessary graph inflation (i.e., spawning excessive duplicate nodes), the algorithm penalizes the creation of new edges. Instead, it prioritizes candidates that already share directed edges with the mapped parents of v. By maximizing edge reuse, the algorithm compresses identical process behaviors into shared structural paths while isolating diverging behaviors.

7. Theoretical Evaluation of Model Size

Consider a process of n parallel events $e_1, \dots, e_n$. A log L of all possible executions will contain all $n!$ permutations. A trivial perfectly fitting DFG of polynomial size exists (allowing all n^n combinations with repeats), but it is analytically useless as it over-generalizes.

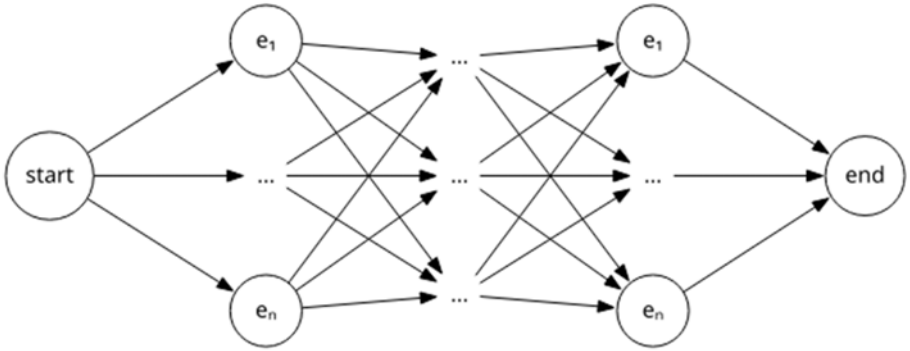

Fig. 2: A polynomial-size DFG model without cycles but with repeating events in its runs.

Definition 6. The *left context* of a node v ($\lambda\backslash(v\backslash) = e$) in DFG G is the set of event sequences $\langle e_1, \dots, e_k, e\rangle$ corresponding to paths from v_{start} to v. The *right context* is symmetrically defined for paths from v to v_{end}.

Definition 7. The *full log* over E (denoted $E!$) is the set of all permutations of its elements.

Proposition 1. There exists a DFG model G such that:

1. G perfectly fits the full log $E!$.
2. G is minimal in terms of nodes (for any perfectly fitting G', $|V| \le |V'|$).
3. For every node $v \in V$, sequences in the left context of v have the exact same length and contain the exact same set of events (differing only by permutation).

Proof. Take any minimal by size DFG G. We can group nodes into layers by distance from the start. G will have exactly $|E| + 2$ layers (fewer prevents fitting, more creates repeats). Edges skipping backward or skipping multiple layers forward can be removed without affecting the fit of $E!$. The resulting model satisfies properties 1-3.

To prove property 3 (same event sets), assume by contradiction that the left context of v contains C and C' such that $c \in C \setminus C'$. Since C leads to v, the log $E!$ dictates that v can be followed by any permutation of $E \setminus C$. But $c \in E \setminus C$, so following C' through v and then executing c results in c repeating, which is forbidden. Contradiction. □

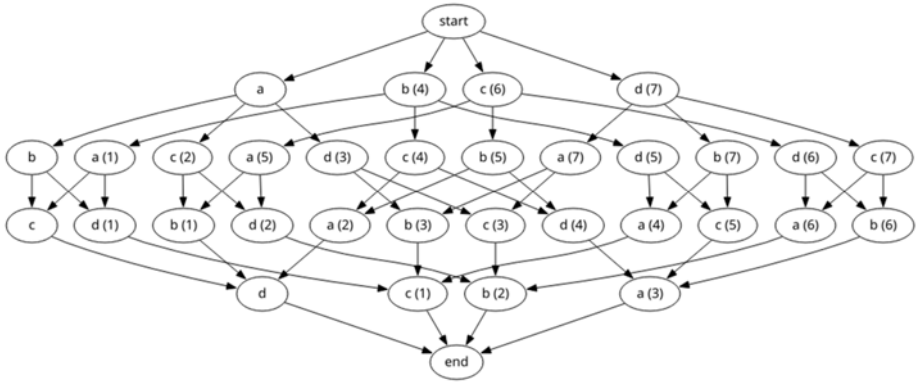

Fig. 3: DFG model for a system of 4 parallel events.

Following this proposition, at the i-th layer ($1 \le i \le n$), there must be a distinct node for every possible subset of i events. Thus, the number of nodes on layer i is $\binom{n}{i}$. Summing over all layers gives $\sum_{i=1}^{n} i\binom{n}{i} = n\,2^{n-1}$. Adding v_{start} and v_{end} yields $n2^{n-1} + 2$. Therefore, for highly concurrent systems, the minimal node-size DFG preventing repeating events grows exponentially.

It is worth noting that this exponential blow-up aligns with established lower bounds in automata theory. An acyclic DFG with duplicated nodes can essentially be viewed as an acyclic Nondeterministic Finite Automaton (NFA). According to the Glaister-Shallit argument [8], any NFA tracking the exact occurrence of elements from a set of size n inherently requires an exponential number of states ($\Omega(2^n)$) to "remember" the executed subsets. However, while general automata theory establishes this asymptotic limit, our derivation provides the exact lower bound on the model size specifically for DFGs (i.e., $n2^{n-1} + 2$ nodes). This confirms that the exponential size of a node-minimal DFG for highly concurrent processes is a fundamental theoretical limit rather than an artifact of a specific synthesis algorithm.

8. Experimental Evaluation

To evaluate the effectiveness of the proposed approach, we conduct a comparative analysis of three distinct DFG discovery algorithms. Algorithm A represents the standard DFG discovery approach (e.g., as implemented in PM4Py [1]), which maps each unique event to a single node, inherently generating spurious cycles in the presence of concurrent behavior. Algorithm B refers to the heuristic approach proposed by Shaimov et al. [5], which mitigates false cycles by duplicating nodes through log partitioning and iterative merging. Finally, Algorithm C denotes our proposed iterative synthesis algorithm, which leverages topological sorting and edge-reuse heuristics to construct compact, strictly acyclic models without repeating events.

Real Logs: We used an educational process log from a Learning Management System (LMS) representing three different student groups (Table 1). The process is globally acyclic, but schedule variations cause event orderings to differ, inducing false cycles for algorithm A.

Results for real logs (Table 2) show that B and C produce larger models than A, which is expected as they duplicate nodes. Consequently, B and C achieve significantly higher precision. Algorithm C produces smaller model than B, but have lower precision, this is consistent with the tradeoff principle.

Artificial Logs: Log 1 simulates two independent company departments executing disjoint tasks in arbitrary order. Log 2 simulates a highly parallel system of 6 completely independent events.

As shown in Table 3, on artificial Log 1, algorithm B takes significantly longer and produces a massively inflated model, whereas C handles it efficiently. This happens because merging algorithm B is more complicated and works poorly with a large number of iterations, while our algorithm uses a simpler heuristic approach that prevents blow of model size. Note that all algorithms evaluated achieve perfect fitness (Fitness = 1.0) by design, hence this metric is excluded from the tables.

9. Conclusion

This paper presents an algorithm for synthesizing DFG models for acyclic processes under the strict constraint that model execution runs contain no repeating events.

Experiments on real and artificial event logs validate the algorithm, demonstrating an optimal precision and compact model size. The method performs comparably to recently proposed alternatives but produces models with lower model size and therefore better simplicity. We also formulated a theoretical bound showing that the minimal node-size DFG avoiding repeating events for a fully parallel system scales exponentially, which is matched optimally by our algorithm.

Table 1: Fragment of an Event Log from an LMS System.

Case_ID	Timestamp	Activity	Grade
...			
-6531...114	2023-09-04	calc class_1 present	1
-6531...114	2023-09-06	bmgc lecture_1 none	0
-6531...114	2023-09-08	bmgc seminar_1 absent	0
-6531...114	2023-09-11	calc class_2 present	1
...			
1259...744	2023-12-18	calc class_14 present	1
1259...744	2023-12-19	calc homework med	7.125
1259...744	2023-12-20	calc quiz med	3.5
...			

Table 2: Comparison of Algorithms on Real Logs.

Log Groups	Alg.	Precision	Time	Size	Cycles
1, 2, 3	A	0.604	0.009 s	127	> 2000000
	B	0.802	0.172 s	224	0
	C	0.702	0.824 s	202	0
1, 2	A	0.580	0.007 s	118	104
	B	0.755	0.089 s	165	0
	C	0.636	0.155 s	156	0
2, 3	A	0.699	0.007 s	124	38014
	B	0.795	0.073 s	201	0
	C	0.772	0.130 s	159	0
1, 3	A	0.670	0.007 s	124	150730
	B	0.827	0.069 s	201	0
	C	0.774	0.146 s	170	0

Table 3: Comparison of Algorithms on Artificial Logs.

Log ID	Alg.	Precision	Time	Size	Cycles
1	A	0.836	0.009 s	8	409
	B	0.987	178.121 s	2133	0
	C	1.000	7.956 s	194	0
2	A	0.580	0.008 s	41	21
	B	0.980	0.089 s	69	0
	C	0.981	0.078 s	79	0

References

[1]. Berti A., van Zelst S., Schuster D. Pm4py: A process mining library for python. Software Impacts, 2023, vol. 17, p. 100556.

[2]. Celonis, 2024. Available at: https://www.celonis.com/, accessed 26.07.2026.

[3]. Günther C., Rozinat A. Disco: discover your processes. In Proc. of the demonstration track of the 10th international conference on business process management (BPM 2012), CEUR Workshop Proceedings, 2012, pp. 40-44.

[4]. van der Aalst W. Academic View: Development of the Process Mining Discipline. Springer International Publishing, 2020. pp. 181-196.

[5]. Shaimov N. D., Lomazova I. A., Nesterov R. A. How to Prevent Fake Cycles in DFG Models Discovered from Event Logs? Programming and Computer Software 51 (6), 395-408.

[6]. Buijs J. C. A. M., van Dongen B. F., van der Aalst W. M. P. Quality dimensions in process discovery: The importance of fitness, precision, generalization and simplicity. International Journal of Cooperative Information Systems, 2014, vol. 23, no. 1, 1440001.

[7]. van der Aalst W. M. P. Process Mining - Data Science in Action. 2nd ed. Springer, 2016.

[8]. Glaister I., Shallit J. A lower bound technique for the size of nondeterministic finite automata. Information Processing Letters, 1996, vol. 59, no. 2, pp. 75-77.

[9]. GitHub acyclic-dfg-synthesis. Available at: https://github.com/ajoulitov/acyclic-dfg-synthesis, accessed 26.07.2026.

Информация об авторах / Information about authors

Артём Константинович ЖУЛИТОВ – студент Национального исследовательского университета «Высшая школа экономики». Сфера научных интересов: майнинг процессов, формальные методы, автоматический синтез моделей процессов.

Artyom Konstantinovich JOULITOV – student at HSE University. Research interests: process mining, formal methods, automated process model synthesis.

Ирина Александровна ЛОМАЗОВА – доктор физико-математических наук, профессор Национального исследовательского университета «Высшая школа экономики». Сфера научных интересов: интеллектуальный анализ процессов, сети Петри, формальные модели параллельных вычислений.

Irina Aleksandrovna LOMAZOVA – Dr. Sci. (Phys.-Math.), Prof. Research interests: include process mining, Petri nets, formal models of concurrency.