GA-ProM: A Genetic Algorithm for Discovery of Complete Process Models from Unbalanced Logs

Sonia Deshmukh[1], Shikha Gupta[2(✉)], and Naveen Kumar[1]

[1] Department of Computer Science, University of Delhi, Delhi 110007, India
[2] Shaheed Sukhdev College of Business Studies, University of Delhi, Delhi 110007, India
shikhagupta@sscbsdu.ac.in

Abstract. Process mining is the practise of distilling a structured process description from a series of real executions. In past decades, different process discovery algorithms have been used to generate process models. In this paper, we propose a genetic mining algorithm (GA-ProM) for process discovery and compare it with other state-of-the-art algorithms, namely, α^{++}, genetic process mining, heuristic miner, and inductive logic programming. To evaluate the effectiveness of the proposed algorithm the experimentation was done on 21 synthetic event logs. The results show that the proposed algorithm outperforms the compared algorithms in generating the quality model.

Keywords: Process models · Genetic mining · Quality dimensions · Process mining · Event logs · Completeness

1 Introduction

Information systems record abundance of data to support automation of business processes in different domains, e.g., supply management, banking, software development, maintenance, and healthcare systems. A process is a collection of tasks or activities having one or more links between the activities. Actors perform the tasks to achieve the business goals. For example, actors in a banking system include bank manager, cashier, other employees, and the customers. The activities are performed by a subset of these actors to complete any process. Opening an account in any bank is a process where a customer performs the following activities—selecting the bank account type, choosing the right bank for creating account, gathering the required documents, completing the application process, collecting the account number, and depositing the funds into the account. Each organisation has detailed description of their processes. For example, in a banking system the documents required for opening an account, transaction limits are clearly described. In any process flow there may be areas of improvement that allow reduction in the overall working time/cost of the system, improve productivity and quality, balance resource utilization [18]. Thus arises the need

S. Sachdeva et al. (Eds.): BDA 2021, LNCS 13167, pp. 209–218, 2022.
https://doi.org/10.1007/978-3-030-96600-3_15

of a technique which could understand the flow and deviations in the processes. Process mining techniques facilitate discovery of process models from the data stored in the form of an event log and use these for further analysis. In the real world, process mining has found applications in the domains of healthcare, information and communication technology, manufacturing, education, finance, and logistics [18].

In this paper, we present a genetic mining approach for discovering process models. The experimentation was done on 21 unbalanced event logs and the proposed algorithm was compared with state-of-the-art algorithms, namely, Genetic process mining, Heuristic miner, α^{++} and Inductive logic programming. The results show that the proposed algorithm generates competitive process models as compared to the compared algorithms and it produces "good" quality process models for all the datasets.

The rest of the paper is organized as follows: Sect. 2 gives a brief summary of process mining concepts and the related work. Section 3 describes the proposed approach. Section 4 outlines the experimentation and explains the results. Section 5 concludes the paper.

2 Process Mining and the Related Work

Process mining algorithms abstract the event log data into a well-structured form known as a process model. An event log is a collection of cases that represents a process. A case, also known as a trace, is a set of events [16]. Each row of the event log represents an occurrence that pertains to a single case. Events are ordered within a case and have attributes such as case ID, activity/task name, and timestamp, among other things. Table 1 depicts an example event log with three tasks and three scenarios. Each distinct task is assigned an integer symbol—'Search product' (1), 'Check availability' (2), and 'Order product' (3). The timestamp indicates the occurrence time of the corresponding event. A trace is formed by picking all the events with the same Case ID and arranging them in ascending order of their timestamp. Each trace (which may include numerous tasks) can be any length and is expressed as a series of integer symbols. In the example log, the traces corresponding to the Case ID 7011 (events 1, 2, 5), 7012 (events 3, 7), and 7013 (events 4, 6) are 123, 12, and 13 respectively. Process mining approaches can be divided into three categories:

- Process discovery: the process of creating a process model from an event log without any prior knowledge [2,17].
- Conformance checking: determining whether the model provided by an algorithm matches the given event log [2,17].
- Enhancement: using information from the original process captured in an event log, improve or extend an existing process model [2,17].

In this paper, we focus on process discovery algorithms. Different state-of-the-art process discovery techniques include α [1], α^{+} [15], Multi-phase miner [10,19], Heuristics miner [22], Genetic process mining (GPM) [16], α^{++} [23], $\alpha^{\#}$

Table 1. An example event log

Case ID	Activities/Tasks	Timestamp
7011	Search product	15-6-2020
7011	Check availability	16-6-2020
7012	Search product	15-6-2020
7013	Search product	15-6-2020
7011	Order product	17-6-2020
7013	Check availability	16-6-2020
7012	Order product	17-6-2020

[24], α^* [14], Fuzzy miner [3], Inductive logic programming (ILP) [12] etc. A few of these algorithms have explored the use of evolutionary methodology for discovering process models [11,16,21]. The recent algorithms in the domain of process discovery include Evolutionary tree miner (ETM) [6], Inductive miner [13], Multi-paradigm miner [8], a hybrid process mining approach that integrates the GPM, particle swarm optimization (PSO), and discrete differential evolution (DE) techniques to extract process models from event logs [7], Fodina algorithm which is an extension of Heuristic miner [4], Binary differential evolution [9]. Most of these algorithms either propose a hybrid technique using multiple evolutionary approach or optimize a weighted function of popular quality metrices of the process model.

3 Proposed Genetic Algorithm for Process Discovery (GA-ProM)

The process mining data is encapsulated in the form of an event log. Event log is the starting point for process discovery algorithms. An individual in the initial population is generated from the event log in the form of a causal relation matrix. Each bit of an individual is either 0 or 1. The steps for the proposed genetic mining algorithm for process discovery are outlined in the Algorithm 2. These steps are explained as follows:

The first task is to compute the dependency links between the tasks/activities in an event log V with n tasks/activities. The dependency relations that represent domain knowledge are stored in the form of the dependency measure matrix D [16]. Dependency measure matrix D which provides information about tasks in self loops, short loops (length-one and length-two loops) and tasks in parallel, is then used to generate causality relation matric M (Algorithm 1) [16]. Each individual in the initial population is represented by a causality relation matrix, which is computed as:

$$m_{t1,t2} = \begin{cases} 1 \text{ if } \mathrm{r} < \mathrm{D(t1,\ t2,\ V)}^P \\ 0, \text{ otherwise} \end{cases} \tag{1}$$

where $t1,\ t2\ \in [1, n]$, $r \in [0, 1)$ is an n × n matrix of random numbers, generated for each individual in the population of N individuals. P regulates the influence of the dependency measure while creating the causality relation. [16] suggested the value of P = 1. A process mining algorithm should, in an ideal situation, generate a model that is simple (simplicity), precise (precision), general (generalization) and fits the available logs (completeness). All the quality dimensions are measured on the scale [0, 1]. Completeness [16] is a measure of the ability of the mined process model to reproduce the traces in the event log. Preciseness [20,21] is a metric for determining how much extra behaviour a model generates that isn't visible in the event log. Simplicity indicates the number of nodes in the mined model [20,21]. Generalization [11] measures the degree of which the mined model can reproduce future unforeseen behaviour.

Algorithm 1. Computation of Initial population

for `i=1:N` **do**
  M^i=*zeros(n,n)*
  `r=rand(n,n),` $r \in [0, 1)$
  for every tuple (t1, t2) in n x n do **do**
    $m^i_{t1,t2} \leftarrow (r < D(t1,t2,V)^P)$
  end for
end for

Algorithm 2. Pseudocode of the Proposed Genetic Algorithm for Process Mining (GA-ProM)

```
Generate N individuals (n x n dimension each) from the given event log
Evaluate each individual and find best solution
while Stopping criteria not met do
   for 1 to number of generations do
      Apply Binary Tournament Selection
      Create child population after applying Crossover and Mutation
      Repair the population
      Combine initial population and repaired population
      Apply Elitism to find fittest individuals
   end for
end while
 Return the best solution
```

In tournament selection, given a population of N individuals, a binary tournament amongst its members is used. Two individuals are randomly selected from the population and every individual must play two times. Out of those 2N randomly selected individuals (N pairs), the best individual is selected on the basis of quality dimensions under considerations. The winner individual goes

into selection pool in the next iteration. Then, on the basis of a random crossover point, the causal relation of an individual is replaced with the casual relation of another randomly chosen individual at that crossover point. Input and output tasks are upgraded in causality matrix. The individuals are then mutated by adding a causality relation, deleting a causality relation or redistributing the relations in input or output sets of casual matrices. The mutations, that are inconsistent with the event log, are rejected. The proposed algorithm stops when there is no improvement in solutions for ten generations in a row, or it executes for the maximum number of specified generations.

4 Experimentation and Results

The population size is set to 100. The proposed algorithm (GA-ProM) is run for maximum of 100 iterations. The total number of runs is fixed at 30 and the results are based on the average performance of these runs. We have picked 21 complex case scenarios without noise [21]. The logs used in this experimentation are imbalanced, i.e., they contain traces with very different frequencies. The event logs were present in the .xes, .xml, and .hn formats. Using Prom 6 we first converted the .xes file into .csv format then traces are formed by picking all the events with the same Case ID and arranging them in ascending order of their timestamp to give the desired input file for experimentation. Summary of the experimental datasets is given in the Table 2.

The proposed algorithm (GA-ProM) is run on 21 unbalanced logs (Table 2), namely, ETM, g2–g10, g12–g15, g19–g25 [21]. These logs contain traces of varying frequencies. As a result, these logs are better for evaluating an algorithm's ability to overfit or underfit the data. The proposed algorithm is compared with α^{++} [23], Heuristic miner [22], Genetic miner [16], and ILP [12] algorithms. The values for completeness, preciseness, and simplicity for ETM, g2–g10, g12–g15, g19–g25 datasets are taken as reported by [21]. However, [21] do not report the value of generalization for these datasets. The value of generalization for ETM, g2–g10, g12–g15, g19–g25 datasets, for the models generated by α^{++}, Heuristic miner, Genetic miner, and ILP algorithms, is computed using the Cobefra tool [5,21]. The parameter settings for each of these algorithms is provided in Table 3.

4.1 Results and Analysis

In this paper, we compare the proposed algorithm (GA-ProM) with α^{++}, Heuristic miner, Genetic process mining (GPM), and ILP algorithms. α^{++} is an abstraction-based algorithm that deals with non-free choice constructs. It is an extension of one of the earliest process discovery algorithm, known as α-algorithm [23]. Heuristic Miner is heuristic-based algorithm and also an extension of α-algorithm. The frequency of activity in the traces is taken into account by heuristic miner to deal with noise and imperfection in the logs [22]. GPM, a main contender for the proposed algorithm, is a pure genetic approach proposed in process mining domain. GPM optimizes a function of completeness and

Table 2. Details of the datasets characteristics

Event-log name	Activities	Traces	Events	Source
ETM	7	100	790	[21]
g2	22	300	4501	
g3	29	300	14599	
g4	29	300	5975	
g5	20	300	6172	
g6	23	300	5419	
g7	29	300	14451	
g8	30	300	5133	
g9	26	300	5679	
g10	23	300	4117	
g12	26	300	4841	
g13	22	300	5007	
g14	24	300	11340	
g15	25	300	3978	
g19	23	300	4107	
g20	21	300	6193	
g21	22	300	3882	
g22	24	300	3095	
g23	25	300	9654	
g24	21	300	4130	
g25	20	300	6312	

Table 3. Parameter settings for the algorithms

Algorithm	Parameters
HM [21,22]	Relative-to-best–0.05, length-one-loops–0.9, length-two-loops–0.9, dependency–0.9, long distance–0.9
α^{++} [21,23]	–
ILP [12,21]	ILP Solver–(JavaILP, LPSolve 5.5), ILP Variant–Petri net, number of places–per causal dependency
GPM [16,21]	Selection–Binary tournament selection, Mutation Rate–0.2, Crossover Rate–0.8, Elitism Rate–0.2, extra behavior punishment–0.025
GA-ProM	Selection–Binary tournament selection, Mutation Rate–0.2, Crossover Rate–0.8

preciseness. ILP was created to add artificially generated negative events, such as the traces describing a path that is not permitted in a process [12]. A process model with "good" completeness value indicates that the process model is

Table 4. Quality dimensions for the process models obtained using α^{++}, HM, GPM, ILP, and the proposed algorithm (GA-ProM) for synthetic datasets (C: Completeness, P: Preciseness, S: Simplicity, G: Generalization)

Algorithm		ETM	g2	g3	g4	g5	g6	g7	g8	g9	g10	g12	g13	g14	g15	g19	g20	g21	g22	g23	g24	g25
Heuristic miner	C	0.37	1	1	0.78	1	0.66	1	0.52	0.74	0.78	1	1	0.91	0.87	0.85	1	1	0.9	0	0.93	0.23
	P	0.98	1	1	1	1	0.99	1	1	1	1	1	1	0.99	1	1	1	1	1	0.42	0.97	0.97
	S	1	1	1	1	1	0.99	0.98	0.93	0.96	1	1	1	1	1	1	1	1	1	0.93	1	0.94
	G	0.62	0.91	0.89	0.81	0.92	0.80	0.81	0.90	0.73	0.60	0.84	0.87	0.54	0.75	0.85	0.94	0.89	0.88	0.60	0.52	0.51
Alpha++	C	0.89	0.33	0	1	1	0.45	0	0.35	0.48	0.56	1	0.48	0	0.05	0.25	0.46	0.68	0.43	0	0	0.97
	P	1	0.96	0.18	0.97	1	1	0.12	1	1	1	0.97	1	0.82	0.14	0.98	0.86	0.09	1	0.42	0.11	0.26
	S	1	0.78	0.79	1	1	0.76	0.93	0.74	0.79	0.76	0.99	0.79	0.79	0.97	0.75	1	0.83	0.32	0.34	0.42	0.89
	G	0.56	0.62	0.74	0.91	0.92	0.84	0.81	0.91	0.59	0.43	0.94	0.95	0.61	0.72	0.72	0.97	0.89	0.65	0.36	0.33	0.45
ILP	C	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1	1
	P	1	0.97	0.97	1	1	0.99	1	0.98	0.98	0.95	0.97	0.07	0.95	0.96	0.98	0.96	0.96	0.93	0.83	0.89	0.8
	S	0.93	0.93	0.92	0.96	1	0.74	0.93	0.67	0.9	0.68	0.99	0.97	0.69	0.76	0.79	0.79	0.97	0.52	0.28	0.7	0.59
	G	0.69	0.99	0.93	0.91	0.92	0.79	0.91	0.92	0.76	0.61	0.94	0.92	0.60	0.78	0.85	0.94	0.89	0.87	0.58	0.66	0.60
Genetic miner	C	0.3	1	0.31	0.59	1	1	1	0.26	0.48	0.48	1	0.75	1	0.15	0.2	1	1	0.43	0.2	0.72	0.41
	P	0.94	1	0.6	1	1	1	1	0.15	1	1	1	1	0.81	0.14	0.08	1	1	0.96	0.42	0.99	0.75
	S	1	1	1	0.97	1	1	1	0.72	0.96	0.88	1	0.95	0.99	0.88	0.95	1	1	1	0.88	1	0.82
	G	0.56	0.91	0.88	0.90	0.92	0.80	0.91	0.88	0.75	0.61	0.91	0.97	0.56	0.50	0.82	0.71	0.89	0.90	0.42	0.47	0.49
GA-ProM	C	1	1	0.997	0.99	1	0.98	0.99	0.99	0.98	0.96	1	1	0.99	0.99	0.99	1	1	0.99	0.87	0.99	0.88
	P	1	1	0.93	0.96	1	0.86	0.93	0.92	0.83	0.75	1	1	0.85	0.99	0.97	1	1	0.98	0.64	0.85	0.69
	S	1	1	1	1	1	1	1	1	1	1	1	1	0.99	0.99	0.99	1	1	0.98	0.99	0.99	0.99
	G	0.88	0.92	0.94	0.91	0.94	0.92	0.95	0.89	0.92	0.90	0.91	0.92	0.95	0.88	0.87	0.93	0.89	0.82	0.94	0.91	0.93

Table 5. Weighted average of the quality dimensions for synthetic datasets

Algorithm	ETM	g2	g3	g4	g5	g6	g7	g8	g9	g10	g12	g13	g14	g15	g19	g20	g21	g22	g23	g24	g25
HM	0.485	0.993	**0.992**	0.816	0.994	0.722	0.984	0.618	0.776	0.8	0.987	0.99	0.89	0.88	0.87	**0.995**	**0.992**	0.914	0.15	0.907	0.36
α^{++}	0.882	0.435	0.132	**0.991**	0.994	0.546	0.143	0.473	0.552	0.602	0.992	0.58	0.17	0.179	0.38	0.57	0.66	0.48	0.086	0.066	0.869
ILP	0.972	0.992	0.986	0.99	0.994	0.963	0.988	0.967	**0.972**	**0.942**	0.992	0.92	0.942	0.962	0.97	0.976	0.986	0.948	**0.899**	0.94	0.92
GPM	0.423	0.993	0.429	0.675	0.994	**0.985**	**0.993**	0.335	0.578	0.561	0.99	0.80	0.951	0.23	0.296	0.978	0.99	0.55	0.286	0.74	0.474
GA-ProM	**0.991**	**0.994**	0.981	0.988	**0.995**	0.971	0.979	**0.976**	0.962	0.939	**0.993**	**0.994**	**0.976**	**0.982**	**0.979**	**0.995**	**0.992**	**0.975**	0.867	**0.973**	**0.871**

able to reproduce the behavior expressed in the event log [6]. Completeness may be considered an important metric to measure the quality of process models. It's important was highlighted by the authors of [6] when they assigned a 10 times higher weight to completeness than the weight assigned to other quality dimensions.

The proposed algorithm optimizes completeness to discover the process models. In order to assess the quality of the discovered process models, we also compute the remaining three quality dimensions—namely, preciseness, simplicity, and generalization. We have also computed a weighted average of the quality dimensions, so as to rank the algorithms based on the weighted average and to compare the quality of the model generated by different algorithms [6].

Table 4 shows the values of the quality dimensions for the process models discovered by different algorithms. The proposed algorithm is able to achieve completeness values for the discovered process models that are better or at least as good as those achieved by the other algorithms. The results show that the discovered process models exhibit "good" values for the other quality dimensions also. Table 5 and Fig. 1 show that in terms of the weighted average, the process models discovered by the proposed algorithm are better in 14 datasets out of 21 in contrast to the other algorithms.

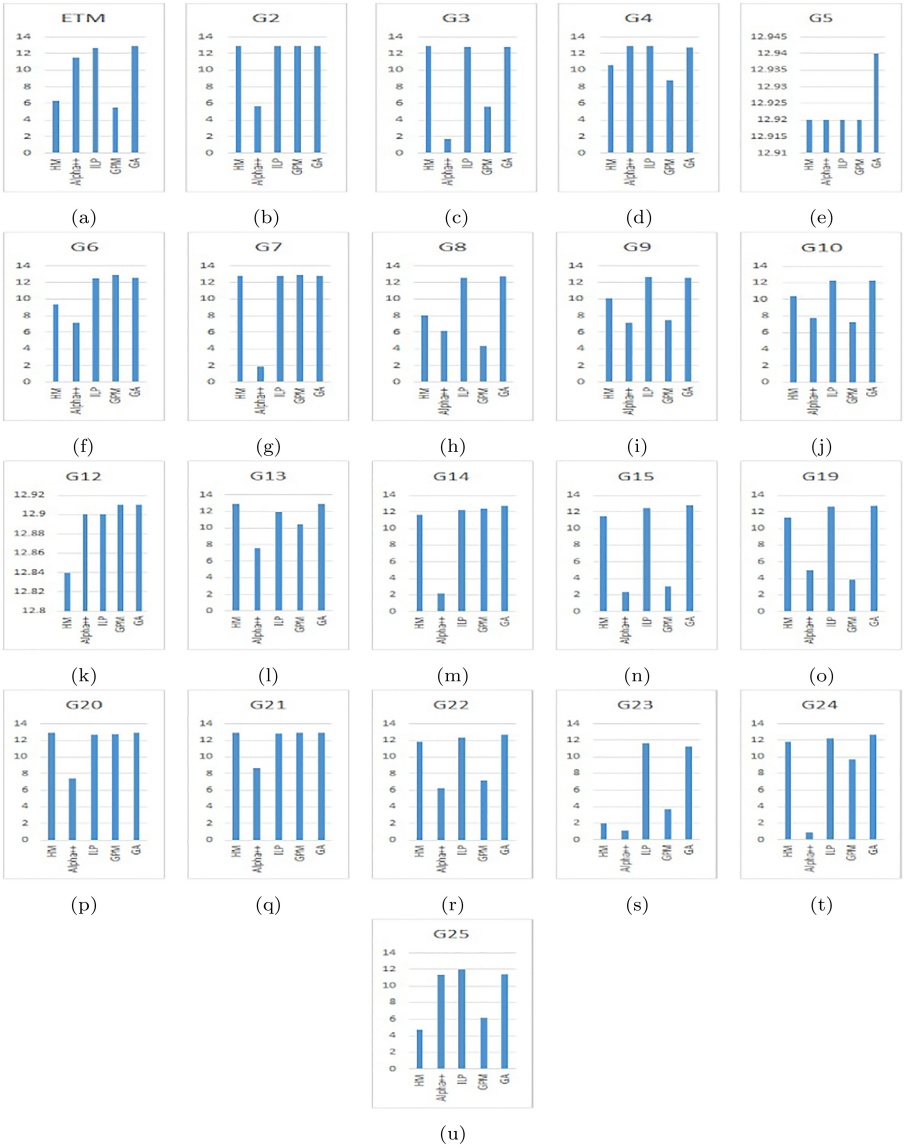

Fig. 1. Plots of weighted average of quality dimensions for the process models generated from Heuristic Miner, α^{++}, ILP, Genetic Process Miner, and the proposed algorithm (GA-ProM). In the figures GA-ProM has been abbreviated as GA.

5 Conclusion

In the past decade, the domain of process mining has developed into an active research area. Process mining techniques empower organizations to gain valuable insights into their business process from the digital data recorded in the information systems. The discovery of process models from the data stored in the form of an event log is usually the first step towards process improvements. The quality of the discovered process models is measured using four metrices, namely, simplicity, preciseness, generalization, and completeness. Completeness is considered more important than the others as a "good" completeness value indicates that the process model is able to reproduce the behavior expressed in the event log.

In this paper, we present a genetic mining algorithm (GA-ProM) for process model discovery from the given event data. We have experimented with completeness as the fitness function. The algorithm was tested on 21 unbalanced logs. We have also compared the proposed algorithm (GA-ProM) with the state-of-the-art algorithms, namely, Heuristic miner, α^{++}, Genetic process Mining (GPM), and Inductive logic programming (ILP). The results show the effectiveness of the proposed approach as it produces "good" quality process models for all the datasets. The proposed approach compares well with the other algorithms in terms of completeness value of the discovered process models and also in terms of the weighted average of the quality dimensions for the process models.

References

1. Van der Aalst, W., Weijters, T., Maruster, L.: Workflow mining: discovering process models from event logs. IEEE Trans. Knowl. Data Eng. **16**(9), 1128–1142 (2004)
2. van der Aalst, W.M.: Process Mining: Data Science in Action. Springer, Heidelberg (2016). https://doi.org/10.1007/978-3-662-49851-4
3. van der Aalst, W.M., Gunther, C.W.: Finding structure in unstructured processes: the case for process mining. In: Seventh International Conference on Application of Concurrency to System Design, ACSD 2007, pp. 3–12. IEEE (2007)
4. vanden Broucke, S.K., De Weerdt, J.: Fodina: a robust and flexible heuristic process discovery technique. Decis. Support Syst. **100**, 109–118 (2017)
5. vanden Broucke, S.K., De Weerdt, J., Vanthienen, J., Baesens, B.: A comprehensive benchmarking framework (CoBeFra) for conformance analysis between procedural process models and event logs in ProM. In: 2013 IEEE Symposium on Computational Intelligence and Data Mining (CIDM), pp. 254–261. IEEE (2013)
6. Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: On the role of fitness, precision, generalization and simplicity in process discovery. In: Meersman, R., Cruz, I.F. (eds.) OTM 2012. LNCS, vol. 7565, pp. 305–322. Springer, Heidelberg (2012). https://doi.org/10.1007/978-3-642-33606-5_19
7. Cheng, H.J., Ou-Yang, C., Juan, Y.C.: A hybrid approach to extract business process models with high fitness and precision. J. Ind. Prod. Eng. **32**(6), 351–359 (2015)
8. De Smedt, J., De Weerdt, J., Vanthienen, J.: Multi-paradigm process mining: retrieving better models by combining rules and sequences. In: Meersman, R., et al. (eds.) OTM 2014. LNCS, vol. 8841, pp. 446–453. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-662-45563-0_26

9. Deshmukh, S., Gupta, S., Varshney, S., Kumar, N.: A binary differential evolution approach to extract business process models. In: Tiwari, A., et al. (eds.) Soft Computing for Problem Solving. AISC, vol. 1392, pp. 279–290. Springer, Singapore (2021). https://doi.org/10.1007/978-981-16-2709-5_21
10. van Dongen, B.F., Van der Aalst, W.M.: Multi-phase process mining: aggregating instance graphs into EPCs and petri nets. In: PNCWB 2005 Workshop, pp. 35–58 (2005)
11. van Eck, M.: Alignment-based process model repair and its application to the Evolutionary Tree Miner. Ph.D. thesis, Master's thesis, Technische Universiteit Eindhoven (2013)
12. Goedertier, S., Martens, D., Vanthienen, J., Baesens, B.: Robust process discovery with artificial negative events. J. Mach. Learn. Res. **10**, 1305–1340 (2009)
13. Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Discovering block-structured process models from event logs - a constructive approach. In: Colom, J.-M., Desel, J. (eds.) PETRI NETS 2013. LNCS, vol. 7927, pp. 311–329. Springer, Heidelberg (2013). https://doi.org/10.1007/978-3-642-38697-8_17
14. Li, J., Liu, D., Yang, B.: Process mining: extending α-algorithm to mine duplicate tasks in process logs. In: Advances in Web and Network Technologies, and Information Management, pp. 396–407 (2007)
15. Alves de Medeiros, A., Van Dongen, B., Van Der Aalst, W., Weijters, A.: Process mining: extending the α-algorithm to mine short loops. Technical report, BETA Working Paper Series (2004)
16. Alves de Medeiros, A.K.: Genetic process mining. CIP-Data Library Technische Universiteit Eindhoven (2006, printed in)
17. van der Aalst, W.M.P.: Process Mining: Discovery, Conformance and Enhancement of Business Processes, vol. 8, p. 18. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-19345-3
18. dos Santos Garcia, C., et al.: Process mining techniques and applications-a systematic mapping study. Expert Syst. Appl. **133**, 260–295 (2019)
19. van Dongen, B.F., van der Aalst, W.M.P.: Multi-phase process mining: building instance graphs. In: Atzeni, P., Chu, W., Lu, H., Zhou, S., Ling, T.-W. (eds.) ER 2004. LNCS, vol. 3288, pp. 362–376. Springer, Heidelberg (2004). https://doi.org/10.1007/978-3-540-30464-7_29
20. Vázquez-Barreiros, B., Mucientes, M., Lama, M.: A genetic algorithm for process discovery guided by completeness, precision and simplicity. In: Sadiq, S., Soffer, P., Völzer, H. (eds.) BPM 2014. LNCS, vol. 8659, pp. 118–133. Springer, Cham (2014). https://doi.org/10.1007/978-3-319-10172-9_8
21. Vázquez-Barreiros, B., Mucientes, M., Lama, M.: ProDiGen: mining complete, precise and minimal structure process models with a genetic algorithm. Inf. Sci. **294**, 315–333 (2015)
22. Weijters, A., van Der Aalst, W.M., De Medeiros, A.A.: Process mining with the heuristics miner-algorithm. Technische Universiteit Eindhoven, Technical report. WP 166, pp. 1–34 (2006)
23. Wen, L., van der Aalst, W.M., Wang, J., Sun, J.: Mining process models with non-free-choice constructs. Data Min. Knowl. Disc. **15**(2), 145–180 (2007)
24. Wen, L., Wang, J., Sun, J.: Mining invisible tasks from event logs. In: Dong, G., Lin, X., Wang, W., Yang, Y., Yu, J.X. (eds.) APWeb/WAIM -2007. LNCS, vol. 4505, pp. 358–365. Springer, Heidelberg (2007). https://doi.org/10.1007/978-3-540-72524-4_38