



# Efficient Discrete Particle Swarm Optimization Algorithm for Process Mining from Event Logs

Gong-Liang Li<sup>1,2</sup> · Si-Yuan Jing<sup>3</sup> · Yan Shen<sup>4</sup> · Bing Guo<sup>1</sup>

Received: 17 August 2021 / Accepted: 1 March 2022  
© The Author(s) 2022

## Abstract

Process mining, which aims to mine a high-quality process model from event log, provides a powerful tool to support the design, enactment, management, and analysis of operational business processes. However, the task is not easy because the algorithm needs to discover various complex process structures, handle noisy and incomplete event logs and balance multiple performance indicators. In this paper, a novel algorithm (called PSOMiner) for process mining is proposed, which consists of a discrete particle swarm optimization algorithm and guided local mutation. The former is in charge of searching the solution space of causal matrix and the latter is used to help the algorithm skip out the local optimum when it suffers from premature. A fine-grained scoring strategy which used to assign a score to each position of a particle (i.e. causal matrix) is presented to direct the mutation. The experiments were performed on 28 synthetic event logs with/without noise and 4 real-life event logs, and three classical algorithms of process mining (ETM, Hybrid ILP Miner, HM) were chosen for comparison. The results show that (1) PSOMiner achieved the best f-score on 25 synthetic event logs; (2) The average f-score of PSOMiner is 0.825 on 4 real-life event logs, which is superior to ETM whose average f-score is 0.703.

**Keywords** Process mining · Discrete particle swarm optimization · Event log · Causal matrix · Guided local mutation

## Abbreviations

ETM	Evolutionary Tree Miner
Hybrid ILP Miner	Hybrid Integer Linear Programming Miner
HM	Heuristic Miner
BPM	Business process management
PSO	Particle swarm optimization
SWF-net	Structural workflow net
CI	Computational intelligence
GM	Guided local mutation
GA	Genetic algorithm

BP	Behavior precision
BR	Behavior recall
SP	Structural precision
SR	Structural recall

## 1 Introduction

In the last decade, process mining, also known as workflow mining, has emerged as a hot topic in the field of business process management (BPM) and it provides a bridge between data mining and BPM. The starting point of process mining is a collection of historical information which is known as event log recorded by information systems. Based on event logs, the goal of process mining is to output a process model observed by the algorithm which reflects the real execution path of business process in an information system. The mined process model can be applied to many fields in BPM, including process model design, analysis of bottleneck of business process, etc. [1, 2].

Recently, the explosive growth of block-chain technology brings new challenges to BPM. Block-chain can make the traditional inter-organizational business process more secure and trustable. BPM can be promoted by the block-chain

✉ Gong-Liang Li  
ligongliang1982@126.com

✉ Bing Guo  
guobin@scu.edu.cn

<sup>1</sup> College of Computer Science, Sichuan University, Chengdu 610000, China

<sup>2</sup> Institute of Computing Applications, China Academy of Engineering Physics, Mianyang 621000, China

<sup>3</sup> School of Artificial Intelligence, Leshan Normal University, Leshan 614000, China

<sup>4</sup> School of Computer Science, Chengdu University of Information Technology, Chengdu 610000, China

technology to applied in many fields, such as international finance, security data exchange (e.g. bank account information, personal health information), and so on. Weber et al. [3] firstly described a proposal to support inter-organizational process through block-chain. Mendling et al. [4] pointed out that how to collect event logs on trusted nodes in the chain and how to discover the real business process among organizations are big challenges in the field of block-chain-based BPM. In this work, we do not consider the collection of event logs and assume that (1) all the required process information have been collected and recorded in an event log; (2) The process information from different organizations has been converted to a unified format. Thus, we only focus on how to mining a high-quality process model from an event log.

Mining a high-quality process model from event logs is not an easy task. On the one hand, besides common process structures (e.g. sequence structure, parallel structure, choice structure, etc.), the process mining algorithm needs to handle lots of special structures, such as duplicated activity, invisible activity, non-free-choice structure, long-distance dependence structure, and so on [5]. On the other hand, event logs are always noisy or incomplete in the real life that may greatly affect the mining results. There are many reasons for the appearance of noisy data in an event log, such as server outage, network congestion, etc. In addition, for a complicated process model, it is possible to get an event log that cannot contain all possible execution paths, thus the event log is often incomplete (or unbalanced).

To evaluate the quality of a process model, researchers have proposed many effective indicators. The most commonly used indicators include replay fitness, precision and simplicity [6]. Replay fitness quantifies the extent to which the discovered model can accurately reproduce the traces recorded in the event log (a trace is described as a sequence of events and it represents a complete path of activity execution). However, only the replay fitness cannot prove the quality of the process model. For instance, the flower model is an extreme case which is able to replay arbitrary sequence of events. To handle this problem, researchers usually employ precision to quantify the fraction of the behavior allowed by the model but not observed in the event log. Simplicity is used to capture the complexity of the process model. A complex model is not required because it is hard to be understood. How to balance these competing indicators to discover an optimal process model is an important task in the design of the algorithm.

Computational intelligence (CI) is an efficient technique to handle such complex task. However, current CI-based process mining algorithms, including GeneticMiner [7], evolutionary tree miner [8, 9], ProDiGen et al. [10], suffer from slow convergence and premature. The reason behind the low efficiency of these algorithms is that all of them are based on genetic

algorithm and their genetic operators are very simple. This paper proposes a new CI-based process mining algorithm, named PSOMiner, which is based on the discrete particle swarm optimization technique. There are three major innovations in the work. Firstly, a series of arithmetic operators are proposed for velocity updating and position updating on causal matrix. Secondly, a local mutation operator is designed to help PSOMiner skip out the local optimum if the premature is detected. Thirdly, each position in a particle (i.e. a candidate process model) is assigned a score during the calculation of fitness value. Then, the score is employed to guide the local mutation to improve the efficiency of the algorithm.

The remainder of this paper is structured as follows. Section 2 reviews some works related to this paper. Section 3 gives some background knowledge of process mining and particle swarm optimization (PSO). Then, the details of PSOMiner are given in Sect. 4. Section 5 explains the experiments and analyzes the experimental results in depth. The last section give conclusions.

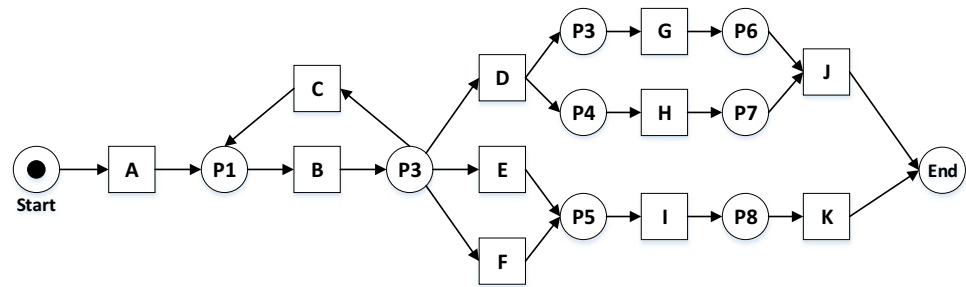
## 2 Related Work

The basic idea of process mining is proposed by Cook and Wolf in 1995 [11]. Their aim is to automatically discover process models from event logs of the software process. Later, Agrawal et al. [12] applied the technique to workflow management and officially named it as process mining. So far, a large amount of achievements have emerged in this field. Generally, these works can be categorized into four groups.

### (1) The $\alpha$ algorithm and its extensions

A milestone in the field of process mining is the  $\alpha$ -algorithm proposed by van der Aalst et al. [13]. The algorithm can discover a structural workflow net (SWF-net) from event logs under the assumption of that the event log is complete. It models the business process by petri net and focuses on discovery of four classical relations (i.e. sequence relation, causal relation, parallel relation and choice relation) among activities from event logs. Based on the  $\alpha$  algorithm, de Medeiros et al. proposed  $\alpha+$  algorithm [14] which can discover length-one loops and length-two loops from event logs. Wen et al. proposed  $\alpha\#$  algorithm [15] and  $\alpha++$  [16] which are able to handle invisible activity and non-free-choice structure respectively. The common characteristic of  $\alpha$  series algorithms are that they extract the basic relationship among activities by scanning traces in event logs to construct the process model. The process model mined by  $\alpha$  series algorithms has low structural complexity thus is easy to be understood. However, these algorithms do not consider replay fitness and precision, thus the quality of the obtained process model is low.

**Fig. 1** An illustrating example of process model



## (2) Heuristics algorithm.

Weijters et al. proposed a heuristic algorithm for process mining, named as Heuristic Miner (HM) [17]. The HM algorithm quantifies the dependency among activities by counting the occurrence frequency of some patterns in an event log. Since the observed patterns with low frequency in an event log are considered to be noises, the HM algorithm has strong ability of anti-noise. Unfortunately, some correct but low frequency patterns are easy to be wrongly handled, thus the HM algorithm is unable to get a perfect process model. Later, they proposed an improved algorithm, called Flexible Heuristic Miner [18], which significantly improves the performance of the HM algorithm. Besides that, Greco et al. proposed CNMiner algorithm [19] which considers precedence constraint in the heuristic method. The disadvantage of the heuristic algorithm is that the performance of the algorithm mainly depends on the number of structure patterns. The more the patterns are, the higher the time complexity of the algorithm is.

## (3) Region-based algorithm.

Bergenthum et al. firstly proposed a process mining algorithm based on language-based region [20]. However, the algorithm does not consider precision of the process model, moreover its time complexity increases exponentially with the size of event log. Werf et al. proposed ILP Miner which is also based on language-based region [21]. The ILP Miner employs integer linear programming technique to construct the process model. van Zelst et al. proposed Hybrid ILP Miner [22] which changes the objective function of ILP Miner and considers a method to handle the behavior with low frequency in an event log. Leemans et al. proposed Inductive Miner [23] which employs a strategy of divide-and-conquer. The Inductive Miner algorithm starts from a single node and it uses the recursive method to combine the adjacent local process models into a larger process model.

## (4) CI-based algorithm.

CI is a powerful tool to solve problems in data mining, nonlinear optimization and other fields. This kind of methods simulate the biological evolution process, and has very strong search ability and robustness. Medeiros et al. firstly

applied genetic algorithm in process mining (named as Genetic Miner) [7]. Through well-defined fitness function and crossover/mutation operators, the Genetic Miner is able to obtain a process model which is very consistent with the event log. Moreover, many problems, including non-free-choice structure, invisible activity, duplicated activity etc., are solved in a unified framework. Vázquez-Barreiros et al. proposed another algorithm ProDiGen [10] which is also based on genetic algorithm. However, the hierarchical objective function used in ProDiGen often obtains an unsatisfactory process model which has perfect replay fitness but very poor precision. Buijs et al. proposed Evolutionary Tree Miner (ETM) [8] which employs process tree to represent a process model. Moreover, the objective function of ETM considers replay fitness, precision and simplicity, therefore it can find high-quality process model.

## 3 Preliminaries

### 3.1 Process Mining

Process mining aims to extract knowledge from an event log and produce a process model which is required to be consistent with real business process. A process model is usually represented by petri net.

**Definition 1** A petri net over a set of activities  $A$  is denoted by a tuple  $N = (P, T, F, \pi)$ , where:

1.  $P$  is a finite set of places;
2.  $T$  is a finite set of transitions, and  $P \cap T = \emptyset$ ;
3.  $F = (P \times T) \cup (T \times P)$  is a finite set of directed arcs connecting places and transitions;
4.  $\pi : T \rightarrow A \cup \tau$  is a function mapping transitions to either activities or  $\tau$ , such that  $\forall a \in A \exists t \in T \pi(t) = a$ .

The symbol  $\tau$  denotes the invisible activity in a process model. Let  $N = (P, T, F, \pi)$  be a petri net,  $a, b \in P \cup T$  are two nodes in  $N$ , if  $\langle a, b \rangle \in F$ , we say that  $a$  is an input node of  $b$ , meanwhile  $b$  is the output node of  $a$ . We use the symbol  $\bullet a = \{b | \langle b, a \rangle \in F \wedge b \in P \cup T\}$  to denote the

input nodes of  $a$ . Similarly, the symbol  $a\bullet$  represents the output nodes of  $a$ .

Figure 1 gives an illustrating example of process model represented by petri net, in which the circles denote places and the squares denote transitions. For a business process, the activities are modeled by transitions and the causal dependency among activities are modeled by places and arcs. It is easy to find that there are eleven transitions and ten places in Fig. 1. Two nodes that named “start” and “end” denote the beginning place and the ending place respectively.

Petri net is a token-based system. The token is represented by a block dot (see the “start” place). If all the input nodes (a.k.a. places) of a transition have one token, the transition is enabled to be *fired*. A firing transition will remove a token from each input place, and then put a token to each output place. For example, in Fig. 1, it can be seen that the input place of transition “A” contains a token, thus the transition “A” is enabled to be fired. It would remove a token from the “start” place and put a token to place “P1”. Then, transition “B” is enabled to be fired. It should be noted that “P3” is an OR-Split structure, that is to say, one of the four following transitions (i.e. “C”, “D”, “E” and “F”) would be able to be fired if “P3” has a token. Besides, “P1” and “P5” are OR-join structure, transitions “D” and “J” represent AND-split and AND-join structure respectively. More details can be found in [13].

Although petri net provides a powerful tool for process modeling, it is hard for us to directly apply a CI-based method on a petri net. To solve this problem, Medeiros et al. [7] proposed causal net (also known as causal matrix) to represent the process model. The definition is given below.

**Definition 2** [7]: A causal matrix is a tuple  $\Pi = (A, C, I, O)$ , where.

- (1)  $A$  is a finite set of activities;
- (2)  $C \subseteq A \times A$  is the causality relation;
- (3)  $I : A \rightarrow P(P(A))$  is an input condition function<sup>1</sup>;
- (4)  $O : A \rightarrow P(P(A))$  is an output condition function.

Furthermore, since a process model represented by petri net is usually compared with a model represented by causal matrix, a method for converting a petri net to a causal matrix is given below.

**Definition 3** Let  $N = (P, T, F, \pi)$  be a petri net. The rules of mapping  $N$  to a causal matrix  $\Pi = (A, C, I, O)$  include:

- (1)  $A = \pi(T)$ .
- (2)  $C = \{(t_1, t_2) \in T \times T \mid t_1 \cdot \cap \cdot t_2 \neq \emptyset\}$ .
- (3)  $I : T \rightarrow P(P(T))$  such that  $\forall_{t \in T} I(t) = \{p \mid p \in \cdot t\}$ .
- (4)  $O : T \rightarrow P(P(T))$  such that  $\forall_{t \in T} O(t) = \{p \mid p \in t \cdot\}$ .

<sup>1</sup>  $P(A)$  is the power set of  $A$

**Table 1** The process model in Fig. 1 as a causal matrix

Activity	I	O
A	{}	{{B}}
B	{{A, C}}	{{C, D, E, F}}
C	{{B}}	{{B}}
D	{{B}}	{{G}, {H}}
E	{{B}}	{{I}}
F	{{B}}	{{I}}
G	{{D}}	{{J}}
H	{{D}}	{{J}}
I	{{E, F}}	{{K}}
J	{{G}, {H}}	{}
K	{{I}}	{}

For explanation, the process model in Fig. 1 is converted to a causal matrix shown in Table 1. It can be seen that each row in the table corresponds to an activity. Each row has two columns, i.e. the input of the activity  $I(a)$  and the output of the activity  $O(a)$ . The input and output are denoted by a set whose elements are also sets. For distinction, the element is called *subset* in the following paper. It should be noted that activities in a same subset of  $I(a)$  have an OR-join relation and those in different subsets have an AND-join relation. On the other hand, activities in a same subset of  $O(a)$  have an OR-split relation and those in different subsets have an AND-split relation.

Take transition “B” as an instance, it has two input places “P1” and “P2”. Moreover, the input transition of “P1” is transition “A” and the input transition of “P2” is transition “C”, thus  $I(B) = \{\{A, C\}\}$ . The output place of “B” is “P3” whose following transitions include “C”, “D”, “E” and “F”, thus  $O(B) = \{\{C, D, E, F\}\}$ .

Then, the definitions of event, trace and event log are given below.

**Definition 4 (Event, trace, and event log)** Let  $A$  be a set of activities, event is denoted as  $e = \omega(a)$ , where  $a \in A$  and  $\omega$  is a function relating each activity to an event. Let  $\Gamma$  be a universe of events, a trace  $t \in \Gamma^*$  is a finite sequence of events. Let  $\Sigma = \Gamma^*$  be a universe of traces, an event log  $L$  is a multi-set of traces, i.e.  $L = \mathcal{B}(\Sigma)$ .

It is not difficult to understand that the process model in Fig. 1 may produce many different execution paths represented by a sequence of events (i.e. a trace) and these traces consist of an event log, just like  $\{ABEIK^{50}, ACFIK^{20}, ACEBF^{100}, ABCBEIK^{80}, ABDGHJ^{60}, ABDHGJ^{45} \dots\}$ . The superscript represents the number of times that the trace appears in the event log.

## 3.2 Particle Swarm Optimization

PSO, which imitates the social behavior of bird flocking, is one of the most popular CI algorithm [24]. The PSO algorithm is initialized with a population of particles randomly distributed in the search place. Each particle imitates a bird which has two properties, i.e. its position  $x_i = (x_{i,1}, x_{i,2}, \dots, x_{i,n})$  and flying velocity  $v_i = (v_{i,1}, v_{i,2}, \dots, v_{i,n})$ , where  $n$  is the dimension of the problem. In addition, each particle is assigned a fitness value calculated by an optimization function. During each generation, each particle updates its velocity and position using the following rules [25]:

$$v_{i,j} = \omega v_{i,j} + c_1 r_{1,j} (pbest_{i,j} - x_{i,j}) + c_2 r_{2,j} (gbest_j - x_{i,j}) \quad (1)$$

$$x_{i,j} = x_{i,j} + v_{i,j} \quad (2)$$

where  $pbest_{i,j}$  and  $gbest_j$  are the value of  $j$ -th dimension of the best-so-far position yielded by the  $i$ -th particle and the whole population respectively. In addition,  $\omega$  is the inertia weight,  $r_{1,j}$  and  $r_{2,j}$  are two random numbers,  $c_1, c_2$  are two acceleration parameters which are used to control the speed flying to the best-so-far position found by the particle itself and the population. In this way, particles learn from their self-cognition and social interaction to approximate the optimal solution of the problem.

## 4 Discrete Particle Swarm Optimization Algorithm for Process Mining

### 4.1 Framework of PSOMiner

Most of the CI algorithms for process mining follow the approach of genetic evolution, such as Genetic Miner, ETM, ProDiGen, etc. All of them suffer from premature and slow convergence speed because of the poor search ability of the genetic operators.

In this work, we propose a novel CI algorithm for process mining which is based on PSO technique, called PSOMiner. The flowchart of the PSOMiner is shown in Fig. 2. Firstly, the algorithm initializes a population of particles by the heuristic method proposed in [7]. Then, the particles are evaluated by the fitness function. If the stopping criteria is achieved, the algorithm will output the best solution. Otherwise, it will check whether or not the algorithm falls into local optimum (i.e. premature). If so, the algorithm will select the guided local mutation algorithm to help itself skip out the local optimum; otherwise it will select the PSO algorithm to fast approximate the optimal solution. There are two termination conditions, which are (1) whether or not the

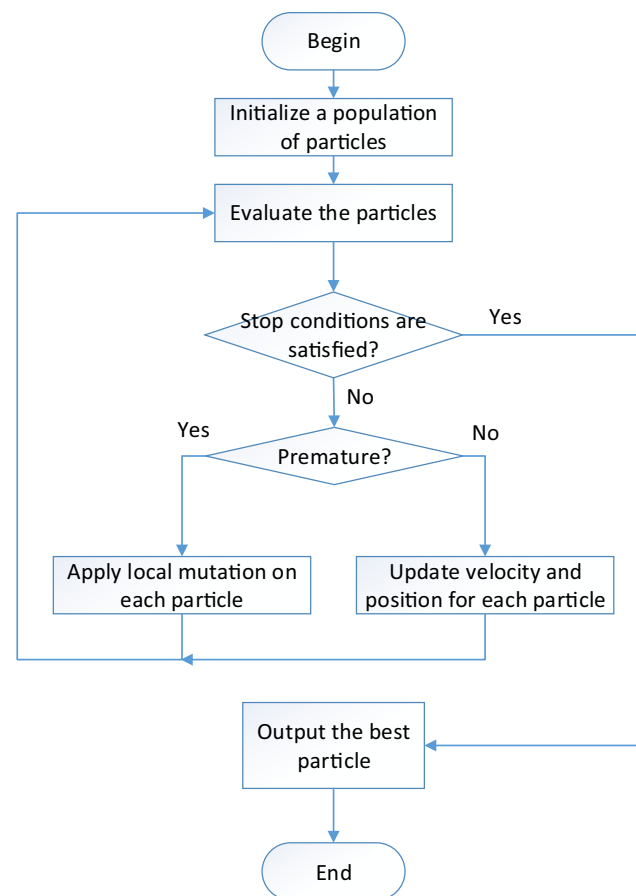


Fig. 2 The flowchart of PSOMiner

number of times that the best fitness value is not changed is higher than a predefined threshold, or (2) the number of generation reaches upper bound. The premature is judged by the variance of the fitness value of all particles. The details of the PSO algorithm as well as the guided local mutation technique will be introduced below.

### 4.2 Fitness Function

In this work, particles are evaluated by a fitness function which is composed of three indicators, including completeness  $F_c$ , precision  $F_p$  and simplicity  $F_s$ . The formula is given below, where  $\lambda_c, \lambda_p, \lambda_s$  are three weight coefficients.

$$fitness = \lambda_c \times F_c + \lambda_p \times F_p + \lambda_s \times F_s \quad (3)$$

The completeness  $F_c$  is in charge of quantifying the ability of a process model that it can properly replay the traces in an event log. For an event log without noise, the completeness of a perfect process model should be 1.0, i.e. all the traces can be properly replayed on the process model. A simple way to define the completeness metric is

the number of correctly replayed traces divided by the total number of traces. However, this definition is too coarse. Consider two process models, one is a totally incorrect process model and the other just misses an arc, the above method cannot distinguish the two models because both of them can not properly replay the event log. This work follows the definition of completeness in [7]. The definition of completeness  $F_c$  is shown in formula (4). The function  $allParsedActivities(\Pi, L)$  returns the total number of activities in the event log  $L$  that could be parsed without problems by the causal matrix  $\Pi$ , and  $numOfActivitiesInLog(L)$  gives the total number of activities in  $L$ .

$$F_c = \frac{allParsedActivities(\Pi, L)}{numOfActivitiesInLog(L)} \quad (4)$$

A process model maybe not a good solution even if it has completeness 1.0. An extreme instance is the flower model, i.e. for any activity, its inputs and outputs are the rest of the activities. The flower model can replay arbitrary traces, so its completeness value is the best. However, it is meaningless. Precision is an indicator for quantifying the fraction of the behavior allowed by the model which is not seen in the event log. However, it is difficult to give a proper definition of precision because it has to detect all the extra behavior in the process model. Therefore, our definition of precision takes into account all the activities that are enabled when the model replays an event log (see formula 5). The function  $allEnabledActivities(\Pi, L)$  returns the number of enabled activities the log  $L$  is replayed on the model  $\Pi$ .

$$F_p = 1 - \frac{allEnabledActivities(\Pi, L)}{\max_{\Pi' \in \Omega} (allEnabledActivities(\Pi', L))} \quad (5)$$

The third dimension of the fitness is simplicity, which is used to measure complexity of a process model [10]. The definition of the indicator is shown in formula (6), where the symbol  $\phi$  and  $\psi$  denote the AND-join structure and the AND-split structure respectively. It has been introduced that AND-split (or AND-join) structure is modeled by the activities in different subset of  $O(a)$ (or  $I(a)$ ). Therefore, what we need to do is to count the number of subset in  $I(a)$  and  $O(a)$ .

$$F_s = 1 - \frac{\sum_{a \in A} (\sum_{\phi \in I(a)} |\phi| + \sum_{\psi \in O(a)} |\psi|)}{\max_{\Pi \in \Omega} (\sum_{a \in A} (\sum_{\phi \in I(a)} |\phi| + \sum_{\psi \in O(a)} |\psi|))} \quad (6)$$

### 4.3 PSO Algorithm

Different with the S-PSO [24, 25], the particle in PSOMiner is naturally represented by causal matrix which is more complex than the representation of particles in

most of the S-PSO works. All the concepts and operators, such as position, velocity, position updating and velocity updating, are required to be redefined. Next, these new concepts and operators will be introduced.

#### 4.3.1 Position

In PSO, the position of a particle usually represents a solution to the problem. In this work, the position of a particle is naturally represented by the causal matrix. Formally, the position is denoted by a tuple  $Pos = (Pos^I, Pos^O)$ , where  $Pos^I = \{I(a) | a \in A\}$  is a set that represents the input structure of each node and  $Pos^O = \{O(a) | a \in A\}$  is a set that represents the output structure of each node. Take the process model in Fig. 1 for an example,  $Pos^I = \{\{\}, \{A, C\}, \{B\}, \{B\}, \{B\}, \{B\}, \{D\}, \{D\}, \{E, F\}, \{G\}, \{H\}, \{I\}\}$ . It can be seen that the representation of particle position in PSOMiner is much more complex than that in the traditional S-PSO. The former contains subsets of arbitrary length, but the latter contains subsets with fix length (just as  $\{(a, b), (b, c), \dots\}$ ).

#### 4.3.2 Velocity

In PSO, velocity determines the speed and direction of a particle to update its position. Similar to the S-PSO, velocity is defined as a set of elements with possibility in this work. The definition of the velocity  $V$  is shown in formula (7).

$$V = \{e/p(e) | e \subseteq P(A)\} \quad (7)$$

where  $e$  is the element which is a subset of the power set of activities  $A$ ,  $p(e) \in [0, 1]$  is the possibility of the subset. Note that an element with possibility zero will be ignored for the sake of convenience.

#### 4.3.3 Velocity Update

The velocity updating rule in PSOMiner is different from the traditional method (see formula 8) because we do not consider the velocity of last generation. The reason behind the consideration is that the algorithm only selects one node to update in each generation, thus the historical velocity may provide trivial information for the search of optimal solution.

$$V_{i,j} = c_1 r_{1,j} (pbest_{i,j} - Pos_{i,j}) + c_2 r_{2,j} (gbest_j - Pos_{i,j}) \quad (8)$$

The symbols  $V_{i,j}$  and  $Pos_{i,j}$  mean the velocity and the position of the  $j$ -th node (including the input set and the output set) of  $i$ -th particle respectively,  $pbest_{i,j}$  is the  $j$ -th node of the historically best position found by  $i$ -th particle, and  $gbest_j$  is the  $j$ -th node of the historically best position found by all particles. The meaning of other symbols are same with the

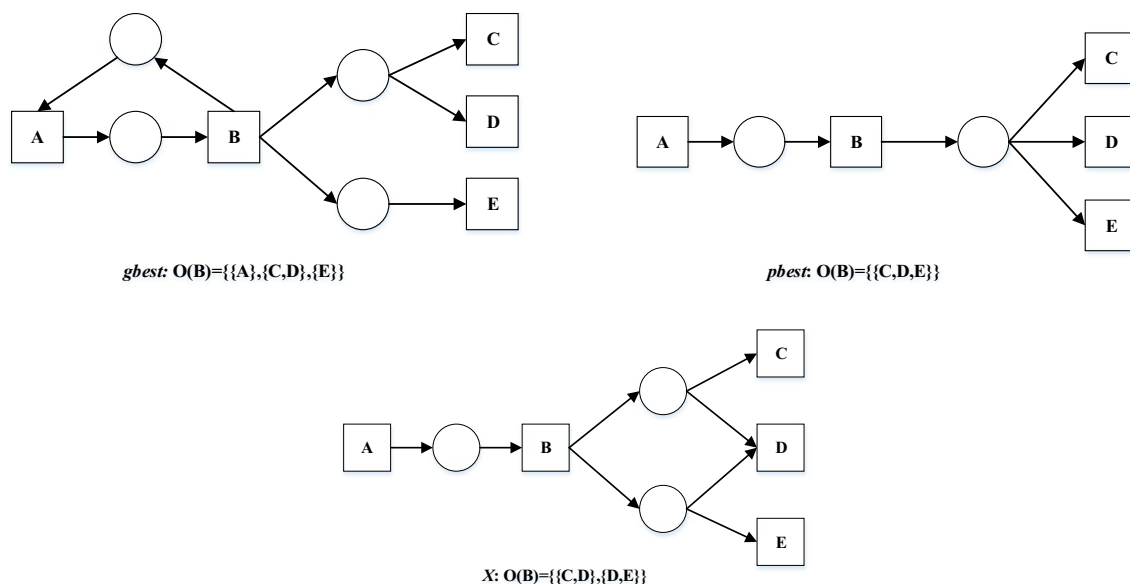


Fig. 3 An example for explanation of operators in PSOMiner

traditional PSO. Then, all related operators will be introduced in detail.

(1) Position—Position (e.g.  $pbest_{i,j} - Pos_{i,j}$ )

It has been explained that position of each particle contains two factors, i.e.  $Pos_i^l$  and  $Pos_i^o$ . Both of them are sets composed of the subsets of the  $P(A)$ . This section makes no distinction between them, i.e., all of the definitions proposed in this section can be applied to both factors. A minus operator between two positions are given below.

$$Pos_{i,j} - Pos_{k,j} = \{e \mid e \in Pos_{i,j} \wedge e \notin Pos_{k,j}\} \tag{9}$$

For explanation, an example is shown in Fig. 3, in which three different structures of the output of node “B” are given. Let  $gbest_j^o = \{\{A\}, \{C, D\}, \{E\}\}$ ,  $pbest_{i,j}^o = \{\{C, D, E\}\}$ ,  $Pos_{i,j}^o = \{\{C, D\}, \{D, E\}\}$ , then  $gbest_j^o - Pos_{i,j}^o = \{\{A\}, \{E\}\}$ ,  $pbest_{i,j}^o - Pos_{i,j}^o = \{\{C, D, E\}\}$ . This operator is able to find out the promising structures that appear in good solution but not appear in current solution.

(2) Real number  $\times$  Position (e.g.  $c_1 r_{1,j} (pbest_{i,j} - Pos_{i,j})$ )

The multiplication operator between a positive coefficient  $c$  and a position  $Pos$  yielded in the last step is defined by formula (10).

$$c \times Pos = \{e/p(e) \mid e \in Pos\}, p(e) = \begin{cases} c & \text{if } c < 1 \\ 1 & \text{if } c \geq 1 \end{cases} \tag{10}$$

By multiplying a positive coefficient  $c$ , the result of minus operation between two positions becomes a set of possibilities, i.e. it has been transformed to a velocity. Continue the

example above, assume  $c_1 r_{1,j} = 0.92$ ,  $c_2 r_{2,j} = 0.84$ , then  $c_1 r_{1,j} (pbest_{i,j}^o - Pos_{i,j}^o) = \{\{D, E, F\}/0.92\}$ ,  $c_2 r_{2,j} (gbest_j^o - Pos_{i,j}^o) = \{\{A\}/0.84, \{E\}/0.84\}$ .

(3) Velocity + Velocity (e.g.  $c_1 r_{1,j} (pbest_{i,j} - Pos_{i,j}) + c_2 r_{2,j} (gbest_j - Pos_{i,j})$ )

The addition operator between two velocities is given by formula (11).

$$V_1 + V_2 = \begin{cases} \{e_1/p(e_1), e_2/p(e_2) \mid e_1 \cap e_2 = \emptyset\} \\ \{e_1/p(e_1) \mid e_1 \cap e_2 \neq \emptyset \wedge p(e_1) \geq p(e_2)\} \\ \{e_2/p(e_2) \mid e_1 \cap e_2 \neq \emptyset \wedge p(e_1) < p(e_2)\} \end{cases} \tag{11}$$

where  $e_1/p(e_1) \in V_1, e_2/p(e_2) \in V_2$ . The principle behind this operator is simple that the structure with higher possibility will be kept for the position updating. Since the possibilities are generated randomly, this method can enhance the diversity of the particles. Continue the above example,  $c_1 r_{1,j} (pbest_{i,j} - Pos_{i,j}) + c_2 r_{2,j} (gbest_j - Pos_{i,j}) = \{\{A\}/0.84, \{C, D, E\}/0.92\}$ . Based on three operators, the velocity updating rule given by formula (8) can be performed.

### 4.3.4 Position Update

So far, a velocity is ready for updating the current position of the particle. However, it is impossible to simply add a velocity to a position because the former contains possibilities. We employ the method proposed in [25] to remove possibilities from the velocity meanwhile filter out the elements

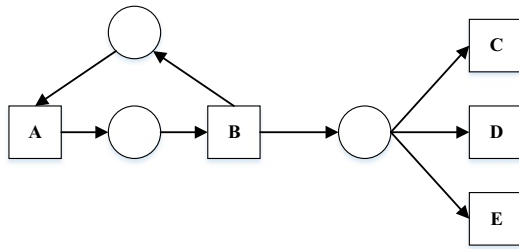


Fig. 4 The result of position update

whose possibilities are lower than a threshold  $\alpha$  (see formula 12).

$$V^\alpha = \{e | p(e) \in V \wedge p(e) \geq \alpha\} \tag{12}$$

Then, the obtained  $V^\alpha$  can be used to update the position of the particle. The operator is given in formula (13).

$$Pos_i + V_i^\alpha = \left\{ e \mid e \in \{e' / \cup V_i^\alpha \mid e' \in Pos_i\} \vee e \in V_i^\alpha \right\} \tag{13}$$

where  $\cup$  denotes a generalized union operation and  $e' / \cup V_i^\alpha$  means that it removes the elements in  $\cup V_i^\alpha$  from  $e'$ . The purpose is that it can keep the promising structures in  $V^\alpha$ . Continue the above example, let  $V_{ij}^{0.8} = \{\{A\}, \{C, D, E\}\}$ , thus  $\cup V_{ij}^{0.8} = \{A, C, D, E\}$  and  $Pos_{ij} + V_{ij}^{0.8} = \{\{C, D\}, \{D, E\}\} + \{\{A\}, \{C, D, E\}\} = \{\{A\}, \{C, D, E\}\}$ . The result is shown in Fig. 4.

#### 4.4 Model Repair

Usually, the new particle is not consistent. For example, let the  $j$ -th activity and  $k$ -th activity of  $i$ -th particle be “B” and “E” respectively, it is possible to obtain  $Pos_{ij}^O = \{\{C, E\}, \{F\}\}$ , but  $Pos_{ik}^I$  does not contain activity “B”. In addition, the input of the “start” activity as well as the output of the “end” activity may be not empty. Therefore, a repair operation is required after the position update. Firstly, we give a definition of consistence to determine whether a causal matrix is consistent or not.

**Definition 5:** let  $\Pi = (A, C, I, O)$  be a causal matrix, it is consistent if it satisfies following conditions:

- $O(start) \neq \emptyset \wedge I(start) = \emptyset$ , where  $start$  is the initial activity;
- $I(end) \neq \emptyset \wedge O(end) = \emptyset$ , where  $end$  is the last activity;
- for  $\forall a' \in \cup I(a), a \in \cup O(a')$ , where  $a, a' \in A \wedge a \neq start \wedge a' \neq end$ ;

for  $\forall a' \in \cup O(a), a \in \cup I(a')$ , where  $a, a' \in A \wedge a \neq end \wedge a' \neq start$ .

Next, the pseudo code of particle repair is shown in algorithm `particle_repair` ( $\Pi, a$ ). The algorithm firstly sets the  $I(start)$  and  $O(end)$  to empty. Then, it traverses all nodes  $a' \in A$  in the particle and repair the  $I(a')$  and  $O(a')$  iteratively.

---

#### Algorithm 1: `particle_repair` ( $\Pi, a$ )

//  $\Pi = (A, C, I, O)$  is a causal matrix;

//  $a$  is the updated activity

---

- (1) **if**  $a = start$
  - (2)  $I(a) \leftarrow \emptyset$ ;
  - (3) **if**  $a = end$
  - (4)  $O(a) \leftarrow \emptyset$ ;
  - (5) **for**  $\forall a' \in A \wedge a' \neq a$  **do**
  - (6) **if**  $a' = start \wedge a' \in O(a)$
  - (7) remove  $a'$  from  $O(a)$ ;
  - (8) repair ( $a, I(a), a', O(a')$ );
  - (9) **else if**  $a' = end \wedge a' \in I(a)$
  - (10) remove  $a'$  from  $I(a)$ ;
  - (11) repair ( $a, O(a), a', I(a')$ );
  - (12) **else**
  - (13) repair ( $a, O(a), a', I(a')$ );
  - (14) repair ( $a, I(a), a', O(a')$ );
- 

---

#### Subroutine: `repair` ( $a, S, a', S'$ )

- (1) **if**  $a' \in S \wedge a \notin S'$
  - (2) randomly add  $a$  into  $S'$ ;
  - (3) **else if**  $a' \notin S \wedge a \in S'$
  - (4) remove  $a$  from  $S'$ ;
- 

#### 4.5 Local Mutation

Although PSO is a stochastic algorithm that has strong search ability, it often suffers from the problem of premature.



In this work, a guided local mutation method is proposed to help the algorithm skip out the local optimum if the premature is detected during the generation.

To guide the local mutation, it requires to locate the incorrect node in a particle. Conformance checking is a hot issue in process mining. The most popular technique for conformance checking is based on alignment [26], in which the A\* algorithm is used to find the optimal alignment between a process model and an event trace. However, this method is not appropriate in this work, because finding the optimal alignment is too time-consuming. Besides, it can only locate the incorrect area but not a node. It is inaccurate for the guidance of the local mutation.

In this paper, we prepare to assign each node a score by a fine-grained method (see formula 14). Besides that it can accurately local the incorrect node, another advantage of this method is that the score can be obtained along with the calculation of fitness value of particles [27], thus it has

better performance than the alignment-based method. It can be seen that there are three factors in the formula, which are *allTokenParsedActivity*, *allMissingTokens* and *allTokensLeftOnActivity*. The three factors represent the total number of tokens parsed on the activity, the number of missing tokens and the number of tokens that are not consumed on the activity during the log replay, respectively.

$$Score_a = \frac{\#allMissingTokenOnActivity + \#allTokenLeftOnActivity}{\#allTokenParsedOnActivity} \quad (14)$$

The pseudo code of local mutation is given in algorithm 2. For each particle, it firstly selects an activity by the roulette wheel strategy. The higher the score of a node, the easier it is to be selected. Then, the algorithm randomly selects a mutation strategy, including add an arc, delete an arc and redistribute the structure, and applies it on the selected node.

**Algorithm 2:** local\_mutation ( $\Omega$ )

//  $\Omega$  is a set of causal matrices of particles

- 
- (1) **for**  $\forall \Pi \in \Omega$  **do**
  - (2)   select an activity  $a$  from  $\Pi$  based on roulette wheel strategy;
  - (3)    $r \leftarrow rand() \% 3$ ;
  - (4)   **if**  $r = 0$
  - (5)     **if**  $rand() \% 2 = 0$
  - (6)       randomly select an activity  $a' \notin I(a)$ , add an arc  $\langle a', a \rangle$  ;
  - (7)     **else**
  - (8)       randomly select an activity  $a' \notin O(a)$ , add an arc  $\langle a, a' \rangle$  ;
  - (9)   **else if**  $r = 1$
  - (10)   **if**  $rand() \% 2 = 0$
  - (11)     randomly select an activity  $a' \in I(a)$ , delete the arc  $\langle a', a \rangle$  ;
  - (12)   **else**
  - (13)     randomly select an activity  $a' \in O(a)$ , delete the arc  $\langle a, a' \rangle$  ;
  - (14)   **else**
  - (15)     **if**  $rand() \% 2 = 0$
  - (16)       randomly redistribute  $I(a)$  ;
  - (17)     **else**
  - (18)       randomly redistribute  $O(a)$  ;
-

**Table 2** Event logs

Model name	#Activities	#Events	Sequence	Choice	Parallelism	Length one loop	Length two loop	Structured loop	Arbitrary loop	Invisible activity
<i>ma6</i>	8	2040	✓	✓	✓					✓
<i>ma7</i>	9	2032	✓	✓	✓					
<i>ma10</i>	12	2665	✓	✓	✓					✓
<i>mal1</i>	9	3976	✓	✓	✓				✓	
<i>mal2</i>	13	5800	✓	✓	✓				✓	
<i>mbn2</i>	42	24,540	✓	✓				✓		✓
<i>mbn3</i>	42	35,527	✓	✓				✓		✓
<i>Herbst3p4</i>	12	5637	✓	✓	✓			✓		
<i>Herbst6p18</i>	7	9844	✓	✓		✓	✓	✓		✓
<i>Herbst6p30</i>	19	14,851	✓	✓						✓
<i>Herbst6p41</i>	16	3600	✓	✓	✓					
<i>M111</i>	6	1987	✓	✓		✓				
<i>M12lo</i>	6	2622	✓	✓			✓			
<i>M12ls</i>	6	4554	✓	✓			✓			✓

## 5 Experiments

This section gives the experimental results and the analysis. The experiments are composed of three parts. Firstly, we compared the search ability and the convergence speed among different combination of strategies to prove the effectiveness of PSOMiner. Secondly, we compared PSOMiner with three state-of-the-art algorithms of process mining on a group of noise-free event logs and a group of noisy event logs to evaluate the performance and the anti-noise ability of the proposed algorithm respectively. Finally, PSOMiner was run on event logs from real world to further validate its performance.

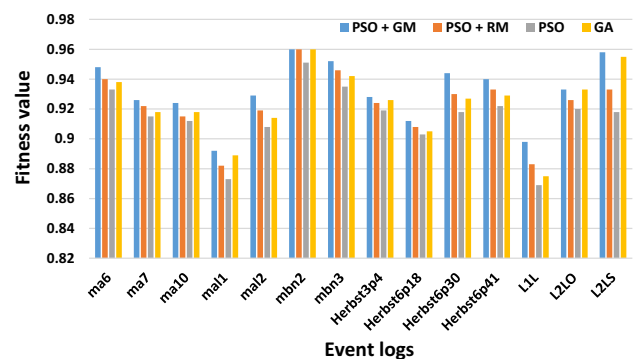
### 5.1 Event Logs and Experimental Settings

There are 32 different event logs used in the experiments. These event logs were divided into three groups. The first group contains 14 noise-free event logs generated by different process models [28]<sup>2</sup>. The process models are produced by the CPN Tool and have different structural characteristics, such as choice structure, parallel structure and loop structure with different length. The details of the event logs are shown in Table 2, in which the activities column is the number of activity contained in the process models and the event column is the number of events simulated by the CPN tool. Besides, each event log have 300 traces. The event logs of the second group were produced by randomly adding noise in event logs of the first group. We randomly choose one

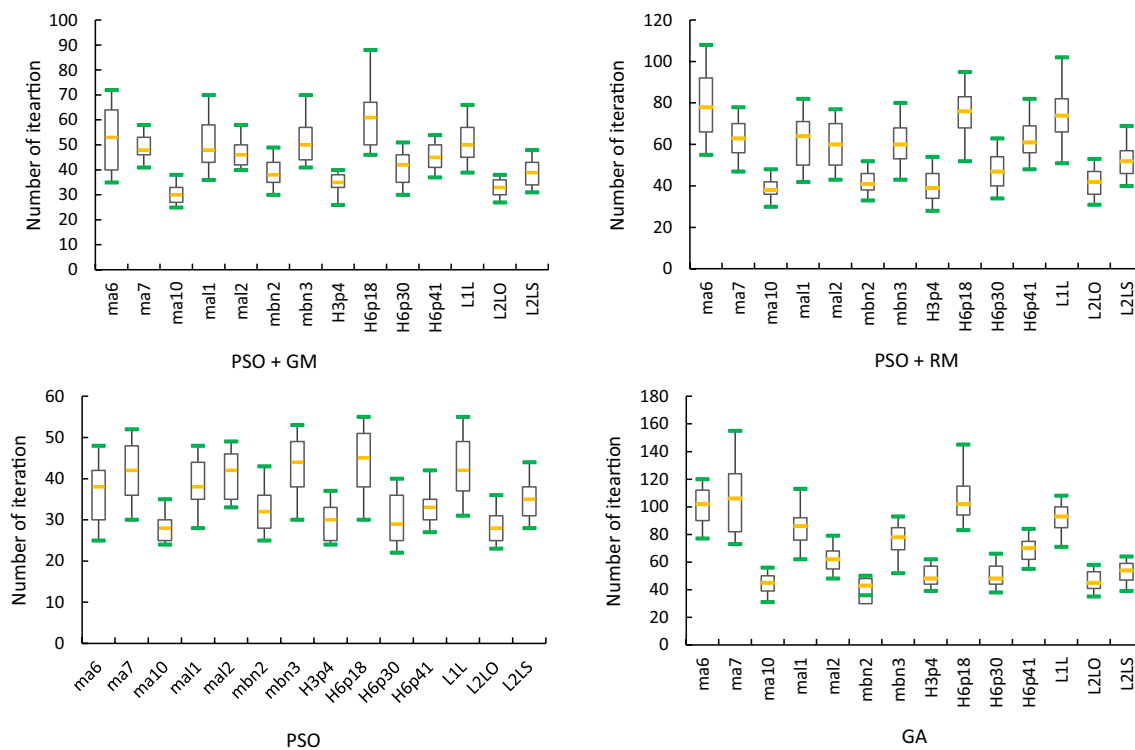
**Table 3** Setting of parameters

Parameters	values
ParticleSize	1 ~ 2
$\lambda_c$	0.7
$\lambda_p$	0.2
$\lambda_s$	0.1
$\mu$	0.1
MaxGeneration	400
MaxNotChanged	20
$c_1, c_2$	2
$\alpha$	0.8

of the following three operations with a probability of 10% to add noise to the event logs, including swap two adjacent events in a trace, add an event to a trace and remove an event from a trace. The third group contains four event logs from real production environment, named Sepsis, BPIC2012,

**Fig. 5** Fitness value achieved by four different methods

<sup>2</sup> <https://svn.win.tue.nl/repos/prom/DataSets/GeneticMinerLogs/>



**Fig. 6** Number of generation of four different algorithms

BPIC2013cp and BPIC2013inc respectively.<sup>3</sup> The former three event logs are from BPI Challenge and all of them are from real world. For example, The BPI2013cp event log is from the IT problem management system of VOLVO and it contains 6660 events that form 1487 traces. The Sepsis event log is from an information system of a hospital which records the process of diagnosis and treatment of sepsis. It contains 15,214 events that form 1050 traces.

The parameter settings are shown in Table 3. The *ParticleSize* is set to 1~2 times of the number of activity in the event log. The reason behind this is the heuristic algorithm of population initialization which would filter out those causal matrices that do not satisfy the constraints. The *MaxGeneration* means the upper bound of the number of generation and the *MaxNotChanged* means that the algorithm will stop if the number of times that the best fitness value does not change exceeds the threshold. Parameters  $\lambda_c$ ,  $\lambda_p$ ,  $\lambda_s$  are set to 0.7, 0.2 and 0.1. The reason is that, for a process model, whether it could reproduce the event log is the most important characteristic that users care about. Parameter  $\mu$  is for premature detection. That is to say, if the variance of the fitness value of particles is lower than 0.1, we think that the algorithm suffers from premature. Then, the guided local mutation strategy is used to help the algorithm skip out the local optimum. The parameter  $\alpha$  is set to 0.8. Parameters

$c_1, c_2$  are set to 2. The reason is that the efficiency of the algorithm would be greatly decreased if the value of parameter  $c$  is too small (e.g. below 1.0) because there is a high probability that  $c \times r$  is less than  $\alpha$ . Conversely, if the value of parameter  $c$  is large, the randomness of the algorithm will be affected.

## 5.2 Effectiveness

In this section, we focus on the validation of the effectiveness of PSOMiner. Four different combinatorial strategies were evaluated, which are PSO+GM (Guided local Mutation, a.k.a. PSOMiner), PSO+RM (Random local Mutation), PSO and GA (genetic algorithm). GA is the Genetic Miner proposed in [7]. Specifically, for comparison, we modified the fitness function and the stopping conditions in GA but the rest part was not changed. The metrics are the best fitness value achieved by these algorithms and the number of generation when they achieve the best results. To ensure the reliability of the results, all the algorithms were run for 20 times and the arithmetic average of the results were calculated.

Experimental results are shown in Figs. 5, 6. It is easy to find that the search ability of PSO is worse than GA. The reason behind it is that the GA algorithm employs some optimization techniques which enhance its search ability. Moreover, both of the number of generation and the achieved

<sup>3</sup> [https://data.4tu.nl/repository/collection:event\\_logs](https://data.4tu.nl/repository/collection:event_logs).

**Table 4** Comparing PSOMiner with three state-of-art algorithms on noise-free event logs

Event Log	ETM				Hybrid ILP Miner				HM				PSOMiner			
	Br	Bp	Sr	Sp	Br	Bp	Sr	Sp	Br	Bp	Sr	Sp	Br	Bp	Sr	Sp
<i>ma6</i>	0.98	0.91	0.93	0.93	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	0.87	0.68	0.93	1.00	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>
<i>ma7</i>	0.95	0.93	0.95	0.97	<b>1.00</b>	0.99	0.95	<b>1.00</b>	0.86	0.78	0.95	0.92	<b>1.00</b>	0.98	<b>1.00</b>	0.97
<i>ma10</i>	0.96	0.86	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	0.97	<b>1.00</b>	0.95	<b>1.00</b>	0.91	0.90	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>
<i>mal1</i>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>
<i>mal2</i>	0.95	0.88	0.94	0.95	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>
<i>mbn2</i>	0.98	0.96	0.98	0.94	<b>1.00</b>	0.99	<b>1.00</b>	0.98	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>
<i>mbn3</i>	–	–	–	–	<b>1.00</b>	0.98	<b>1.00</b>	0.98	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>
<i>Herbst3p4</i>	0.93	0.82	0.94	0.82	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>
<i>Herbst6p18</i>	0.95	0.82	0.92	0.92	<b>1.00</b>	0.80	<b>1.00</b>	0.80	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	0.92	0.98	<b>1.00</b>
<i>Herbst6p30</i>	0.92	0.98	0.90	0.91	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	0.98	<b>1.00</b>	0.95	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>
<i>Herbst6p41</i>	0.93	0.99	0.95	0.95	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>
<i>L1L</i>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	0.95	0.97	0.86	1.00	0.95	0.97	0.87	1.00	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>
<i>L2LO</i>	0.91	0.83	0.90	0.90	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>
<i>L2LS</i>	<b>1.00</b>	0.95	<b>1.00</b>	0.90	<b>1.00</b>	0.95	<b>1.00</b>	0.90	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>	<b>1.00</b>

fitness value of PSO are the least among four algorithms. It demonstrates that PSO is easy to fall into local optimum. The quality of the results achieved by PSO + RM is slightly worse than that achieved by GA. It is inferior to GA in 9 of 14 event logs, but the convergence speed of PSO + RM is obviously faster than that of GA. Finally, compared with PSO + RM and GA, PSO + GM achieves much better results in both the fitness value and the convergence speed. It won gold medals in 12 of 14 event logs and shared the best result in the rest 2 event logs. Moreover, its convergence speed is faster than PSO + RM and GA. The experimental results prove the effectiveness of PSOMiner.

### 5.3 Comparison with State-of-the-Art Algorithms

This section compares PSOMiner with three state-of-the-art algorithms of process mining, which are ETM, Hybrid ILP Miner and HM, to evaluate the performance of the proposed algorithm. We use ProM 6.9 [29] which is the most popular platform for process mining. The parameters of three algorithms for comparison are set to default. It should be pointed out that the output process model of three algorithms are different. The output of ETM is a process tree and the output of Hybrid ILP Miner is a petri net and the output of HM is a heuristic net. Fortunately, ProM 6.9 provides function to convert process tree and heuristic net to petri net. For comparison, we convert the obtained petri nets to causal matrices based on definition 3.

To evaluate process models represented by causal matrix, we employ four metrics proposed in [7], which are behavior precision (BP), behavior recall (BR), structural precision (SP) and structural recall (SR). The former two metrics

require to replay an event log on the obtained process model and the original process model. BP evaluates how much behavior is allowed by the obtained process model but not by the original process model. BR is for the opposite. If the obtained model perfectly fit to the original model, both of the BP and BR would be 1.0. The latter two metrics are based on the causal relations of the obtained process model and the original process model. SP evaluates how many causal relations the obtained process model has but not in the original process model, and SR is for the opposite.

In the experiments, both of PSOMiner and ETM were run for 20 times because they are stochastic algorithms. We calculate the arithmetic mean value of the four metrics of the output model. Firstly, the experiments were performed on event logs of the first group (a.k.a. the noise-free event logs). The results are shown in Table 4. To facilitate the analysis, the optimal value of metrics obtained by these algorithms are labeled by bold. From the table, it is easy to find that ETM, Hybrid ILP Miner, HM and PSOMiner mine the best process model (i.e. totally same with the original model) on 2 event logs, 8 event logs, 10 event logs and 12 event logs respectively. It can conclude that the performance of PSOMiner is better than other three state-of-art process mining algorithms on noise-free event logs.

The proposed algorithm only lost two event log which is *ma7* and *Herbst6p18*. To find out the reason why PSOMiner did not achieve the best results on the two event logs, the most frequent obtained results on *ma7* was analyzed in depth (see Fig. 7). In the figures, OR-S, OR-J, AND-S and AND-J denote structures of OR-Split, OR-Join, AND-Split and AND-Join respectively. We find that the incorrect part is the input of activity “D” labeled by red color. The original input of activity “D” is  $I(D) = \{\{C, E\}, \{E, F, B\}, \{G\}\}$

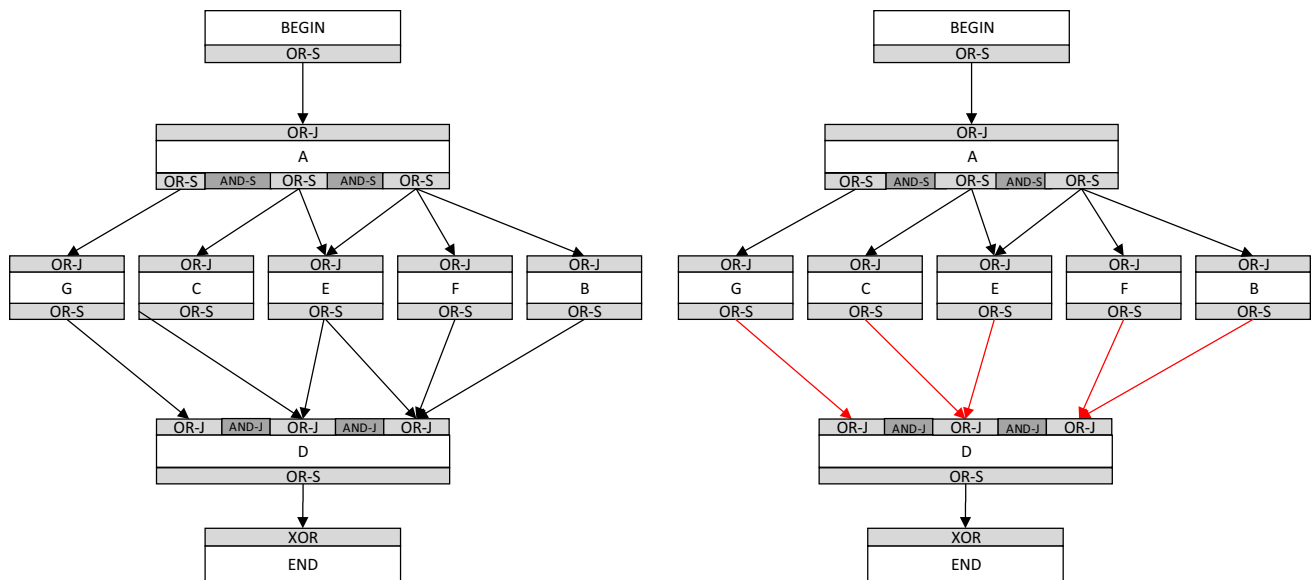


Fig. 7 The original process model of a7 (left) and the obtained process model of a7 (right)

Table 5 Comparing PSOMiner with three state-of-the-art algorithms on event logs with 10% noise

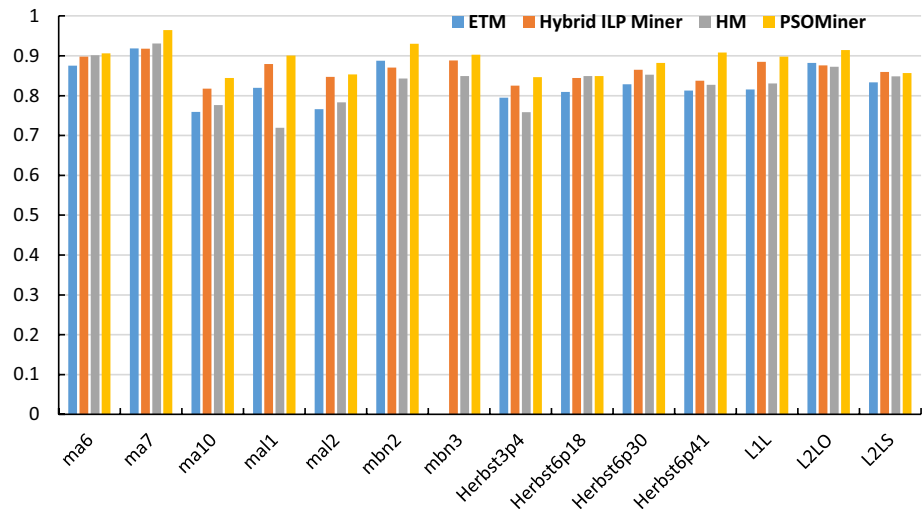
Event Log	ETM				Hybrid ILP Miner				HM				PSOMiner			
	Br	Bp	Sr	Sp	Br	Bp	Sr	Sp	Br	Bp	Sr	Sp	Br	Bp	Sr	Sp
ma6	0.98	0.91	0.86	0.93	<b>1.00</b>	0.89	<b>0.92</b>	0.87	<b>1.00</b>	0.89	0.91	0.93	<b>1.00</b>	<b>0.98</b>	<b>0.92</b>	<b>0.96</b>
ma7	0.98	0.91	0.86	0.93	<b>1.00</b>	0.95	<b>1.00</b>	0.98	<b>1.00</b>	0.88	0.94	0.86	<b>1.00</b>	<b>0.98</b>	0.92	0.96
ma10	0.90	0.74	0.82	0.62	0.91	0.79	0.86	<b>0.73</b>	0.84	0.73	0.86	0.70	<b>0.96</b>	<b>0.82</b>	<b>0.91</b>	0.72
mal1	0.92	0.73	0.84	0.81	<b>0.95</b>	0.81	0.85	<b>0.92</b>	0.91	0.67	0.75	0.60	0.94	<b>0.83</b>	<b>0.92</b>	<b>0.92</b>
mal2	0.88	0.62	0.81	0.80	0.94	0.79	0.83	<b>0.84</b>	0.89	0.73	0.74	0.79	<b>0.95</b>	<b>0.81</b>	<b>0.88</b>	0.79
mbn2	<b>1.00</b>	0.74	0.95	<b>0.90</b>	<b>1.00</b>	0.73	0.99	0.82	0.98	0.74	0.93	0.77	<b>1.00</b>	<b>0.84</b>	<b>1.00</b>	<b>0.90</b>
mbn3	-	-	-	-	0.96	0.84	0.95	<b>0.82</b>	<b>1.00</b>	0.79	0.91	0.74	<b>1.00</b>	<b>0.86</b>	<b>0.98</b>	0.80
Herbst3p4	<b>0.96</b>	0.70	0.85	<b>0.72</b>	0.93	0.77	0.92	<b>0.72</b>	0.95	0.66	0.89	0.63	<b>0.96</b>	<b>0.82</b>	<b>0.92</b>	<b>0.72</b>
Herbst6p18	0.93	0.74	0.92	0.70	0.92	<b>0.76</b>	0.92	0.80	<b>0.96</b>	0.75	<b>0.95</b>	0.78	0.95	0.74	0.93	<b>0.81</b>
Herbst6p30	0.92	0.80	0.86	0.75	0.95	0.78	<b>0.96</b>	<b>0.80</b>	<b>0.96</b>	0.76	0.95	0.78	0.95	<b>0.84</b>	<b>0.96</b>	<b>0.80</b>
Herbst6p41	0.87	0.80	0.88	0.72	0.93	0.73	0.95	0.78	0.91	0.74	0.92	0.77	<b>0.99</b>	<b>0.83</b>	<b>1.00</b>	<b>0.84</b>
L1L	0.87	0.75	0.83	0.82	<b>1.00</b>	<b>0.78</b>	0.94	<b>0.85</b>	0.87	0.75	0.92	0.80	<b>1.00</b>	<b>0.78</b>	<b>1.00</b>	<b>0.85</b>
L2LO	0.98	0.85	0.89	<b>0.82</b>	<b>1.00</b>	0.80	0.98	0.77	<b>1.00</b>	0.82	<b>1.00</b>	0.73	<b>1.00</b>	<b>0.90</b>	0.98	0.80
L2LS	0.90	0.75	<b>0.96</b>	0.76	<b>0.98</b>	0.77	0.92	<b>0.80</b>	0.97	<b>0.80</b>	0.92	0.74	0.96	0.79	0.94	0.77

and the obtained is  $I(D) = \{\{C, E\}, \{F, B\}, \{G\}\}$ . The reason behind this phenomenon is that the position update operation will remove all the elements in  $\cup V_i^\alpha$  from the particle's position of last generation. For example, assume  $Pos = \{\{C, E\}, \{E, F, B\}\}$  and  $V^\alpha = \{\{C, E\}, \{G\}\}$ , then  $Pos + V^\alpha = \{\{C, E\}, \{F, B\}, \{G\}\}$ . It can be seen that the intersection of two sets is removed. The proposed algorithm suffers same problem on another event log.

Furthermore, the experiments were performed on 14 event logs with 10% noise. Experimental results are given in Table 5. From the results, we can find that PSOMiner achieves the best performance on most of the metrics of

these event logs. However, it cannot prove that the process models mined by PSOMiner are better than that of other algorithms, because a process model may have very good performance on three of the four metrics, but very poor on the last one. For fairness, we propose a new metric *fscore* which integrates four metrics together (see formula 15). The results of calculation is given in Fig. 8. It is easy to find that PSOMiner is obviously superior to other three algorithms on 8 event logs and is slightly superior to other three algorithms on 5 event logs. It only slightly inferior to Hybrid ILP Miner on the last event log (a.k.a. *L2LS*).

**Fig. 8** f-score of four algorithms on event logs with 10% noise



The results demonstrate that PSOMiner has good ability of anti-noise.

$$f_{score} = \frac{B_p \cdot B_r}{B_p + B_r} + \frac{S_p \cdot S_r}{S_p + S_r} \quad (15)$$

#### 5.4 Results on Real Data Sets

This section gives the results of experiments performed on four real event logs. Since the process models of these event logs are unknown, the metrics used in previous section are no longer applicable. Three alignment-based metrics, which are alignment-based fitness, alignment-based precision and their combination (i.e. fscore) [30], were employed to evaluate the performance of PSOMiner on real event logs. Moreover, for fairness, we refer to the results of benchmark given in [31]. The output of PSOMiner (a.k.a. causal matrix) was converted to petri nets based on the method in [7], since the tool only supports petri net as the input.

PSOMiner was run for 20 times on each event log. The results are shown in Table 6. Compared with the results of ETM in [31], it is easy to find that the quality of the process models mined by PSOMiner are superior to the quality of the

process models mined by ETM. Especially on BPIC2012, BPIC2013cp and BPIC2013inc, the f-score value obtained by PSOMiner is much better than that obtained by ETM. Moreover, compared with other state-of-art algorithms, the f-score obtained by PSOMiner on four event logs are also very competitive. These results prove that PSOMiner can perform well on real event logs.

## 6 Conclusion and Future Work

In this paper, we propose a novel PSO algorithm, named PSOMiner, to solve the problem of process mining. The most interesting thing is that a series of operators based on causal matrix are designed for position updating and velocity updating in PSOMiner. Besides that, a guided local mutation technique is embedded to address the problems of premature.

We validated the performance of the algorithm through a large number of experiments. Experimental results prove that (1) PSOMiner has strong search ability and fast convergence speed. (2) Compared with three state-of-the-art algorithms of process mining, PSOMiner can achieve the optimal results on 12 in 14 noise-free event logs and 13 in 14 noisy event logs. It demonstrates that PSOMiner has good performance and anti-noise ability. (3) PSOMiner achieved competitive results in the last experiment and it proves that PSOMiner can perform well on real world event logs.

Except for the step of evaluation, other steps of PSOMiner are easy to be parallelized, therefore we can employ modern parallel or distributed techniques, e.g. GPU or cluster, to further speed-up the algorithm. Next, the CI-based process mining can be further studied from two aspects. On the one hand, we can utilize current CI-based process mining techniques to solve deeper tasks in this field, such as cross-organization process mining,

**Table 6** The experimental results of performing PSOMiner on four real event logs

Event log	Fitness	Precision	F-score
BPIC2012	0.82 ± 0.04	0.77 ± 0.03	0.79 ± 0.03
BPIC2013 <sub>cp</sub>	0.98 ± 0.02	0.90 ± 0.09	0.94 ± 0.06
BPIC2013 <sub>inc</sub>	0.95 ± 0.05	0.74 ± 0.08	0.83 ± 0.07
Sepsis	0.90 ± 0.06	0.63 ± 0.05	0.74 ± 0.05

blockchain-oriented process mining, etc. On the other hand, we can study some recent CI techniques and propose new CI-based algorithms of process mining with higher performance.

**Author contributions** G-LL contributed to the algorithm design and performed the experiments. S-YJ helped perform the experiments; YS helped design the algorithm; BG is a director of this work.

**Funding** This work was supported in part by the National Natural Science Foundation of China under Grant No. 61772352; National Key Research and Development Project under Grant No. 2020YFB1711800 and 2020YFB1707900; the Science and Technology Project of Sichuan Province under Grant No. 2019YFG0400, 2021YFG0152, 2020YFG0479, 2020YFG0322, 2020GFW035, and the R&D Project of Chengdu City under Grant No. 2019-YF05-01790-GX.

**Availability of Data and Materials** The data sets used to support the findings of this paper can be downloaded from <https://svn.win.tue.nl/repos/prom/DataSets/GeneticMinerLogs/>.

## Declarations

**Conflict of interest** All authors declare that they have no competing interest.

**Open Access** This article is licensed under a Creative Commons Attribution 4.0 International License, which permits use, sharing, adaptation, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if changes were made. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by/4.0/>.

## References

- van der Aalst, W.M.P.: Process mining: data science in action (Second Edition). Springer-Verlag, Berlin (2016)
- Ruschel, E., Santos, E.A.P., Loures, E.D.F.R.: Establishment of maintenance inspection intervals: an application of process mining techniques in manufacturing. *J. Intell. Manuf.* **31**(1), 53–72 (2020)
- Weber, I., Xu, X.W., Riveret, R., et al.: Untrusted business process monitoring and execution using blockchain. In: Proceedings of the 14th International Conference on Business Process Management (BPM'16), 2016, pp. 329–347
- Mending, J., Weber, I., van der Aalst, W.M.P., et al.: Blockchains for business process management-challenges and opportunities. *ACM Trans. Manage. Inform. Syst.* **9**(1), 41–416 (2018)
- van der Aalst, W.M.P., ter Hofstede, A.H.M., Kiepuszewski, B., et al.: Workflow patterns. *Distrib. Parallel Database* **14**(1), 5–51 (2003)
- Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P., et al.: On the role of fitness, precision, generalization and simplicity in process discovery. In: Proceedings of On the Move to Meaningful Internet Systems Conference, 2012, pp. 305–322.
- Alves de Medeiros, A.K., Weijters, A.J.M.M., van der Aalst, W.M.P.: Genetic process mining: an experimental evaluation. *Data Min. Knowl. Disc.* **14**(2), 245–304 (2007)
- Buijs, J.C.A.M., van Dongen, B.F., van der Aalst, W.M.P.: A genetic algorithm for discovering process trees. In: Proceedings of IEEE Congress Evolutionary Computation (CEC'12), 2012, pp. 1–8.
- van Eck, M.L., Buijs, J.C.A.M., van Dongen, B.F.: Genetic process mining: alignment-based process model mutation. In: Proceedings of the Business Process Management Workshops, 2014, pp. 291–303.
- 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)
- Cook, J.E., Wolf, A.L.: Automating process discovery through event-data analysis. In: Proceedings of the 17th International Conference on Software Engineering (ICSE'95), 1995, pp. 73–82.
- Agrawal, R., Gunopulos, D., Leymann, F.: Mining process models from workflow logs. In: Proceedings of the International Conference on Extending Database Technology, 1998, pp. 467–483.
- van der Aalst, W.M.P., Weijters, A.J.M.M.: Workflow mining: discovering process model from event logs. *IEEE Trans. Knowl. Data Eng.* **16**(9), 1128–1142 (2004)
- Medeiros, A., Dongen, B., van der Aalst, W.M.P., et al.: Process mining for ubiquitous mobile systems: an overview and a concrete algorithm. In: Proceedings of the Ubiquitous Mobile Information and Collaboration Systems, 2004, pp. 156–170.
- Wen, L.J., Wang, J.M., van der Aalst, W.M.P., et al.: Mining process models with prime invisible tasks. *Data Knowl. Eng.* **69**(10), 999–1021 (2010)
- Wen, L.J., van der Aalst, W.M.P., Wang, J.M., et al.: Mining process models with non-free-choice constructs. *Data Min. Knowl. Disc.* **15**(2), 145–180 (2007)
- Weijters, A.J.M.M., van der Aalst, W.M.P.: Rediscovering workflow models from event-based data using little thumb. *Integr. Comput. Aided Eng.* **10**(2), 151–162 (2003)
- Weijters, A.J.M.M., Ribeiro, J.T.: Flexible heuristics miner (FHM). In: Proceedings of the IEEE Symposium on Computational Intelligence and Data Mining (CIDM'2011), 2011, pp. 310–317.
- Greco, G., Guzzo, A., Lupia, F.: Process discovery under precedence constraints. *ACM Trans. Knowl. Discov. Data* **9**(4), 321–3237 (2015)
- Bergenthum, R., Desel, J., Lorenz, R., et al.: Process mining based on regions of languages. In: Proceedings of the 5th International Conference on Business Process Management (BPM'07), 2007, 375–383.
- van der Werf, J., van Dongen, B., Hurkens, C., et al.: Process discovery using integer linear programming. *Fund. Inform.* **94**(3–4), 387–412 (2009)
- van Zelst, S.J., van Dongen, B.F., van der Aalst, W.M.P., et al.: Discovering workflow nets using integer linear programming. *Computing* **100**(5), 529–556 (2018)
- Leemans, S.J.J., Fahland, D., van der Aalst, W.M.P.: Scalable process discovery and conformance checking. *Softw. Syst. Model.* **17**(2), 599–631 (2018)
- Chen, W.N., Zhang, J., Chung, H.S.H., et al.: A novel set-based particle swarm optimization method for discrete optimization problems. *IEEE Trans. Evol. Comput.* **14**(2), 278–300 (2010)
- Chen, W.N., Tan, D.Z.: Set-based discrete particle swarm optimization and its applications: a survey. *Front. Comp. Sci.* **12**(2), 203–216 (2018)
- van der Aalst, W.M.P., Adriansyah, A., van Dongen, B.: Replaying history on process models for conformance checking and

- performance analysis. *WIREs Data Min. Knowl. Discov.* **2**(2), 182–192 (2012)
27. Adriansyah, A., van Dongen, B., van der Aalst, W.M.P.: Conformance checking using cost-based fitness analysis. In: *Proceedings of the 15th IEEE International Enterprise Distributed Object Computing Conference (EDOC'11)*, 2011, pp. 55–64.
  28. Alves de Medeiros, A.K.: *Genetic process mining*. Ph.D. thesis, Eindhoven University of Technology, 2006.
  29. Verbeek, E., Buijs, J.C.A.M., van Dongen, B.F., et al.: ProM 6: The Process Mining Toolkit. In: *Proceedings of the 8th International Conference on Business Process Management (BPM'10)*, 2010.
  30. Adriansyah, A., Munoz-Gama, J., Carmona, J., et al.: Alignment based precision checking. In: *Proceedings of the international workshop on business process intelligence*, 2012, pp. 137–149.
  31. Augusto, A., Conforti, R., Dumas, M., et al.: Automated discovery of process models from event logs: review and benchmark. *IEEE Trans. Knowl. Data Eng.* **31**(4), 686–705 (2019)

**Publisher's Note** Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.